STOCK MARKET CLUSTERING: AN EXTENSIVE STUDY ON UNSUPERVISED TIME-SERIES REPRESENTATIONS

A dissertation submitted to the University of Manchester
for the degree of Master of Science in the Faculty of Science and Engineering

Submitted 2021

Author: Yuyang Liu
Student ID: 10660111
School of Engineering

Supervised by Dr. Xiaojun Zeng

Signature _____

Date _____ / _____ / _____

List of Tables

List of Figures

Abstract

Giving a short overview of the work in your project.

Declaration

I hereby declare that this dissertation is all my own work.

Copyright

Contents

Chapter 1

Introduction

1.1 Overview

As one of the most popular financial products, stock takes high risks for high rewards. Its price is affected by innumerous uncertain factors such as policies of local government, breaking of diseases, etc. [69]. To find strategies that can maximize profits and reduce risks, stock investors may need to extract the latent trading and price moving patterns of stock market [28]. However, due to the randomness, complexity and uncertainty existing in stock market, those patterns are elusive and hard to be discovered [33]. [7] states that even though researchers made massive efforts to teaching machine to recognize data patterns automatically in the past decades, humans are still the best pattern recognizers in most cases so far. Based on whether expert derived patterns exists (and being used) or not, pattern discovery methods can be classified into two categories: supervised methods and unsupervised methods [26]. In terms of stock price, supervised methods require fluctuation patterns pre-defined by experts, while unsupervised methods do not need prior knowledge of interesting structures. Consider that stock market is dynamic and rapidly developing and stock price series shows no periodic trajectory (at least in the short term) [62], the pre-defined patterns may need to be updated frequently (relies heavily on experts). Increasing stock price data volume makes manually defining patterns much more challenging [7]. These problems make stock researchers such as [71] focus more on unsupervised pattern discovery methods.

1.2 Motivation

The majority of existing stock-related researches aim to predict the stock price or moving trending. However, [62] already shown that stock pirce is unpredictable at least in the short term. Instead of directly predicting the stock pirce, we hope to retrieve the latent stock price moving patterns to help decision making. As mentioned in section 1.1 supervised methods require domain knowledge, therefore, we seek help from unsupervised methods. Most previous works such as [26, 71] mainly try to modify clustering algorithms to get better performance. They use the original data (or part of the data) as the input vector, ignore the usage of representation learning. In this project, we will adapt some novel time-series data representation methods, especially the path signature feature (PSF) used in [11, 14, 37], combined with traditional clustering algorithms to better reveal the hidden fluctuation patterns of stock pirce.

1.3 Aims and Objectives

The general goal of this project is to categorize different stock price trajectories into several classes. It can be detailed as follows:

1. Finding a proper representation method that can reduce the data dimension and better reveal the feature of a stock price trajectory.

2. Grouping those generated feature vectors to get the final set of hidden patterns based on clustering algorithms.

To achieve the goal, following works are need to be done:

1. Review the recent works related to time-series data representation and clustering algorithms.

2. Collect stock price data from open-source websites.

3. Preprocess data. In this process, we will apply the common numerical data processing pipeline to our data set.

4. Apply data representation algorithms to our data, analyze the effectiveness of those transformation in different aspects. This may require numerical analysis and visualization.

5. Apply multiple clustering algorithms to the generated representation.

6. Evaluate and compare the performance of those clustering algorithms based on certain metrics.

7. Visualize the patterns generated by clustering algorithms.

8. Integrate pattern matching algorithm into this project. Apply it to test set for the final evaluation.

1.4 Description of the Work

Project scope:

In this project, we will analyze some of the US-based stocks trading on the NYSE, NASDAQ, and NYSE. In our dataset, the ealierest records could date back to 1970s, and it's hard to analyze the whole series. Instead, we will split their historical series into smaller fragments. To get more stable patterns, we mainly focus on the long-term price fluctuation, meaning that each smaller fragment will cover more than 6 months data. Data compression and representation algorithms will be applied to each fragment to generate the feature vectors of them. Then those feature vectors will be grouped by clustering algorithms, the centroid of each group is regarded as a unique pattern. Finally, the value of the generated patterns will be examined by using pattern matching methods to checking whether similar patterns can be found in test data.

The final deliverables of this project will be:

1. The data set used in this project

2. The code implementation of this project

3. The dissertation with experimental results, analysis and visualization of this project

1.5 Report Structure

The remaining chapters are organised as follows: Chapter 2 briefly introduces works related to time-series data representation and clustering. Chapter 3 shows the detailed methodology used in this project. Chapter **??** introduces some metrics that can be used to evaluate the performance of the algorithms used in this project. Chapter **??** includes our progress and future work.

Chapter 2

Background and Related Works

This chapter introduces the basic concepts and relevant techniques involved in this project, including stock market, stock market data, time-series analysis, machine learning, data clustering, representation learning, nerual networks. In addition, representative works that are related to this project are reviewed.

2.1 Stock Market and Stock Market Data

A stock is a term used to describes the partial ownership certificates of any company, while a share represents the stock certificate of a particular company. Given the two terms, a stock market can be regarded as a place for the transfer, trading and circulation of issued shares. Because it is based on the issuance market, stock market is also called the secondary market. A stock exchange is a subset of a stock market, but they are usually used interchangeably. New York Stock Exchange (NYSE), Nasdaq, and the Chicago Board Options Exchange (CBOE) are three most well-known stock exchanges in the stock market of the U.S. For companies, stock markets are the main place to raise necessary capital from investors. For investors, they can get returns from their shares.

Every action in stock markets can be treated as stock market data, but in reality, merely the prices and trades information of shares are recorded. Those information are often studied by investors, traders and researchers for making better decision. Investors and traders make use of the data to decide the shares buying and selling, they try to predict the future movement of the prices to avoid risks and make more profit. Researchers use those data to make assumptions and test their hypothesis.

In the past, stock data were collected and recorded by financial data vendors. After information technological revolution, those data are manipulated and stored automatically by computers. Through the Internet, basic information such as the open price, closing price, exchange code are public accessible. Figure 2.1 shows an sample of stock market data from Internet.

Symbol	Company Name	Last Price	Change	% Change	Market Time	Volume	Avg Vol (3 month)	Market Cap
TCEHY	Tencent Holdings Limited	71.47	-1.16	-1.60%	3:59 PM EDT	2.47M	2.32M	696.54B
BABA	Alibaba Group Holding Limited	212.1	-2.66	-1.24%	4:00 PM EDT	11.01M	14.89M	584.93B
PDD	Pinduoduo Inc.	107.45	-3.29	-2.97%	4:00 PM EDT	7.08M	6.71M	134.67B
JD	JD.com, Inc.	75.59	-1.27	-1.65%	4:00 PM EDT	6.50M	11.16M	119.61B
NTES	NetEase, Inc.	112.1	-2.31	-2.02%	4:00 PM EDT	1.17M	2.00M	75.22B
BIDU	Baidu, Inc.	179.58	-5.94	-3.20%	4:00 PM EDT	3.73M	5.32M	62.40B
TME	Tencent Music Entertainment Group	11.7	-0.70	-5.65%	4:00 PM EDT	22.78M	16.60M	19.80B
WB	Weibo Corporation	60.29	-2.37	-3.78%	4:00 PM EDT	1.06M	1.58M	13.74B
VIPS	Vipshop Holdings Limited	18.58	-1.09	-5.54%	4:00 PM EDT	8.77M	12.35M	12.78B
IQ	iQIYI, Inc.	12.31	-0.42	-3.30%	4:00 PM EDT	6.65M	11.34M	9.72B
ATHM	Autohome Inc.	60.63	-2.11	-3.36%	4:00 PM EDT	301.47k	733.96k	7.58B

Figure 2.1: Sample of stock market data from Yahoo

Stock market data can be stored and transfered with various data model, in reality, they are mostly in the form of time-series data, meaning that there is always a time axis with it. This project mainly studies on the price sequences of stocks and therefore can be categorized into time-series analysis taks.

2.2 Time-Series Analysis for Stock Market Data

2.2.1 Time-series data

According to [20], time-series data are a collection of observations extracted repetitively at continuous specific time points. Given a small sampling interval, the dimension of time-series data could be extremely large. In addition, in time-series data, each data point is numerical.

These two facts show the basic properties of time-series data: numerical, continues and high-dimensional. Apart from stock market data, there are numerous data are in the form of time sequence, such as the voice data and video data.

Figure 2.2: Electrocardiogram is an real word example of time-series data. Figure source: Catholic Health

Generally, analysis of time-series data can be used in following four applications: (1) sequence data compression; (2) influential factor revelation; (3) pattern identification; (4) sequence prediction. Sequence data compression aims to find a way to compress data while preserve the most information, it can benefit the data storage. Influential factor revelation aims to find the for sequence change, it can be used in abnormality discovery. Pattern identification aims to extracting common phenomena existing in different sequences, it can be used in sequence comparison. Sequence prediction aims to predict the future trend of sequences based on the historical data, it can be used in decision making. Apart from those applications, researchers can seek help from time-series analysis to explain finacial phenomena without using sophisticated economic theories [27].

2.2.2 Stock data analysis

For decades, researchers attempted to reveal the rules behind stock price's changing. Their works can be categorized into two basic groups: (1) directly predicting stock price, including using traditional "technical analysis", neural network and fuzzy time-series, etc.; (2) grouping similar stocks to reveal the co-movement of stocks, including using clustering algorithms

and charts, etc. However, due to the intrinsic properties of stock market data, analysis of them still remains challenging. Those intrinsic properties come from the fact that stock price movement process is close to Brownian motion process. And this means the finacial sequences are highly unstable, show no periodical trends [58]. How to retrieve useful information from the random-walk-like sequence data is the core of the stock data analysis.

In terms of finacial data forecasting, [32] summarize two major theories: (1) technical analysis; (2) intrinsic value analysis. When analyze finacial data, these two theories can be used separately or jointly. Technical analysis assumes that the there is a finite set of stock patterns, and the trends of price behaviours can be recognized or matched in that set. It uses historical data such as the trading volume and price of share to learn the rules behind stock market behaviours, and predicts the future trends of that market based on the learned information. The main technical implementation of this theory in recent years is chart-based. In this implementation, stock data are represented by different charts, and these charts and then fed into machine learning models to produce task-specific results.

Intrinsic analysis (also called fundamental analysis) does not explicitly model the price behaviours of shares, to the contrary, it assumes that the price is affected by various factors and try to model on that factors. Those factors should be generally measurable (but may not in numerical), and they can be categorized into two groups: (1) internal factors; (2) external factors. In ternal factors indicate the business conditions such as the pecuniary condition and management methods. External factors indicate the general market conditions such as economic environment, political environment. Given those factors and the real numerical data of a stock, intrinsic analysis then will estimating the intrinsic value of that stock to decide whether to buy of sell the stock [32].

2.2.3 Practical methods for stock data analysis

Previous section introduces the intrinsic properties of stock market data, and the general approaches to deal with such data. As stated by [1], many researches attempting to predict stock trend fails, even though the historical data are available. At least right now, there is no model that can perfect fit stock price movement. This fact makes the most of recent works focus on stock pattern identification rather than stock price prediction. Clustering algorithms are well-known for its ability to group similar data and reveal hidden structures of data, and hence are often used to find patterns. Typical clustering algorithms are unsupervised, meaning that they do not require label information of data. For different clustering algorithms, they have different focus, but in general, the basic objective of them is to minimize the within-cluster scatters while maximize the between-cluster scatters. This objective is quite intuitive, since the generated clusters are expected to be discriminative enough.

Each clustering algorithms has its advantages and disadvantages, suitable data distributions and unsuitable data distortions. To apply traditional clustering algorithms to time series data, the data should be converted to static data first. In terms of time series clustering approaches, there are three major categorizes - model-based, feature-based and distance-based.

Model-based approaches presume a distribution/model for all clusters based on the raw data, and then try to fitting the parameters to each cluster by some optimization algorithms such as Expectation Maximization. Some common models used for time series clustering tasks are mainly statistical, include: (1) Mixture of Gaussian; (2)Autoregressive Markov model; (3) Hidden Markov model [43]. For such approaches, any probabilistic distance measures can be used when comparing two sequences.

Feature-based approaches try to extracting features from the original sequences, use the feature vectors to represent the raw data. Once the transformation is done, traditional clustering algorithms can be applied. In such an approach, the choice of appropriate features

becomes the key part. One feasible and commonly used scheme is that, compacting the long time series sequences with possibly multiple dimensions into one row vectors, and used traditional feature selection method such as zero norm optimization [13] to form the feature sets. One typical and well-known example of such scheme is Time Series Shapelets [64], which extracts a set of local features from the whole dataset, and then use the extracted features for further analysis. Another group of techniques also try to project the original sequences into vectors, but different from the previous scheme, the features produced by this group are no longer the subsets of the original data, insetad, they are drawn from the new feature space transformed from the original data space. Typical transformation techniques used for time series data include: (1) Fourier transformation; (2) statistical feature representation; (3) neural network based auto-encoder. For such approach, any distance metric that can measure the similarity between two sequences can be used.

Distance based approaches aim to find efficient distance functions that can precisely reveal the similarity between two sequences. With those distance functions, traditional clustering algorithms can be directly applied to raw time series sequences without any sequences transformation. Traditional distance metrics such as Euclidean distance and Cosin distance are sensitive to time distortions and small fluctuations, which commonly exist in time series data. This property makes traditional indexes inappropriate for time series sequences comparison. To overcome the problem, some distance metrics are proposed. Dynamic Time Warping (DTW) and its variations are the most well-known and successful indexes for sequence data comparison. Different from "lock-step" measures, DTW is elastic, which means that it allows comparing two sequences of different length. The combination of DTW and traditional clustering algorithms such as K-means was consider to be the most effective approach for time series clustering until few years ago, despite the computation of DTW is time consuming. The recent state-of-the-art time-series clustering algorithm is K-shape [46], a combination of Shape Based Distance (SBD) and K-means. It has been approved that SBD achieves similar results to cDTW (a variation of DTW) and DTW while being orders

of magnitude faster.

2.3 Related Researches

1. Pattern Discovery from Stock Time Series Using Self-Organizing Maps (2001) [26]:
   This work proposes a pattern discovery clustering algorithm for stock data. This algorithm works by segmenting stock sequences with sliding window first. Then those segments are fed into self-organizing map (SOM) to produce patterns. Compared with other clustering algorithm, SOM does not require explicit setting of cluster number k. It is a neural network based unsupervised algorithm, and it introduces redundancy to automatically control the number of output patterns. One problem of using SOM algorithm is that the execution time grows exponentially with the increment of the length of patterns. To reduce the computation resource and boost the algorithm, this work introduces perceptually important point (PIP) based sequences comparison algorithm.

2. Clustering stock market companies via chaotic map synchronization (2005) [6]:
   This work uses chaotic map clustering (CMC) algorithm to identify the temporal patterns of the stock prices. In CMC, each element is projected into a new feature space, and each data point is regarded as a site of a grid, hosting a chaotic map dynamics. Then the mutual information between maps is computed and used as a similar index to decide the clusters. This work test the effectiveness of CMC on stock market data, each company/stock is assigned to a map, and similarity between stocks/companies were measured by the coupling strengths between maps. This work proves that co-movement of stock price exists in the same industrial branch.

3. Mining stock category association and cluster on Taiwan stock market (2008) [34]:
   This work uses a two-phase data mining approach to help stock portfolio making in Taiwan stock market. In phase 1, Apriori algorithm is used to mine the patterns and rules behind stock market data. Those mined information are then used to identify association between stocks and stock category. In phase 2, K-means algorithm is used

to mine stock category cluster based on their price sequences. This work proves that similar stock price fluctuation can happen within geographic regions.

4. Clustering Indian stock market data for portfolio management (2010) [44]:
   This work extensively examines the perfomance of K-means, SOM and Fuzzy C-means clustering algorithms on stock data with 9 evaluation criteria. It experiments with one-year period data from Bombay Stock Exchange. Slightly different with other works, this work focus on returns, validation ratio, enterprise value rather than the exact share price. By comparing the three clustering algorithms with different indexes such as Silhouette Coefficient, Davies Bouldin, Dunn's index on different number of clusters, this work shows that the clusters generated by K-means are the most compact.

5. Clustering Stock Price Time Series Data to Generate Stock Trading Recommendations: An Empirical Study (2017) [42]:
   This work proposes a stock recommendation system that combines the existing works. This system adpots the same SOM algorithm used in [26] to group stocks. The biggest difference is that, to reduce the dimensionality of patterns, it uses regression trees rather than PIP-based compression.

This project adpots the common methods used in those works to set up experiments, including the stock data processing pipeline and the parameter settings of clustering algorithms. By reviewing those work, it can be found that most of stock clustering related works try to improve the quality of analysis by changing clustering models or changing analysis focus. However, as proved by many researchers such as [61, 64], the simple nearest neighbor algorithm with well-extracted features can output competitive results in most time series classification tasks. We believe that such observation can be found in time series clustering tasks. Hence the aim of this project is to invest the different representation methods for time series data and test their effectiveness in the context of stock grouping task.

2.4 Related Techniques

This section gives a brief introductions of techniques used in this project, including time-series data transformation, and clustering algorithms. More details of them can be found in corresponding sections.

2.4.1 Time-series data transformation

Stock price records are typical time-series data, and they have natures such as large data volume, high dimensionality, etc. Analyzing and storing the raw data may need huge computing resource, therefore, in most algorithms, the data are transformed to a shorter form. There are two types of such transformation for time-series data - compression and representation, and both of them are evaluated in this project. Followings are introductions and some related works of them:

1. Time-series compression: the aim of compression is reducing the number of data points while reserving the general shape of the sequence. This makes it different from typical representation, since the latter one often changes the visual appearance of the original data (note that compression can also be regarded a special case of representation). There are several methods to reduce the dimension of the original data, simple approaches include: (1) resampling [3], which selects n points from the start point of a time series with a fixed step (n is the dimension after compression) ; (2) piecewise aggregate approximation (PAA) [29, 65], which segments the time series into n sub trajectories, and uses the numerical mean of each trajectory to represent the whole time series. More promising methods aim to use the perceptually important points (PIP) to represent the time series [25]. The PIP identification algorithm is first proposed by [16] and then used wildly in time series data analysis. Given a time series $P = \{P_1, P_2, \cdots, P_k\}$, where $P_t \in P$ is a data point, PIP identification process will calculate the importance of all the data points, and the first n points with higher importance will be used to represent the whole series. This process works as follows: the

start point P_1 and the end point P_2 in P are the first two PIPs. The third point will be the one with maximum distance to the first two PIPs. The order of remaining PIPs will be decided by their vertical distance to the line crossing its two adjacent PIPs. This process continues until n PIPs are found (n is the reduced dimension) or all points in P are ordered. More detailed introduction and comparison of them can be seen in Section 5.3.

2. Time-series representation: similar to comparison, representation can also be used to reduce the dimensionality of the original data. However, as mentioned above, representation will usually transform the original data into a new feature space and hence will change the visual appearance of the data. The aim of such transformation is extracting useful features from time series that invariant to the distortion. Statistical features representation [45] is one typically method used to transform sequence data. It uses some common statistical features such as the mean value, skewness, kurtosis to represent the sequence. Those features contain the shape information and hence can be used to shape-based clustering tasks such as this project. Bag-of-Patterns representation [36] imports the Bag-of-Word method to time-series data, it splits the sequence into sub-sequences and assigns a string to each of them, then uses histogram of strings to represent that sequence. Path signature [14] uses a set of path integrals to represent sequence, each element in that set indicates a shape feature of that sequence. ROCKET [21] randomly generates massive 1-dimensional kernels and convolve them with a sequence to generate a feature map, and the feature map to represent that sequence. MVP [67] imports the word embedding training process used in Transfomer to train the sequence embedding without labels, and use the trained model to produce the representation of each sequence. More detailed introduction and comparison of them can be seen in Chapter 5.

2.4.2 Clustering algorithms

Clustering is a classical and important task in unsupervised learning. This project will use it to find the stock price moving patterns. Traditional clustering methods can be summarized into following 9 categorizes. Since the main focus of this project is not about clustering algorithms, their technical details are not discussed.

1. Partition based: clustering algorithms that assume that the center of data points is the center of the corresponding cluster. Typical algorithms include K-means [38] and K-medoids [47].

2. Hierarchy based: clustering algorithms that generate clusters based on the hierarchical relationship among data points, either in a bottom-up or top-down way. Typical algorithms include BIRCH [70].

3. Fuzzy theory based: clustering algorithms that use possibility to represent belonging relationship among objects rather than binary status 0 or 1. One data points could belongs to several clusters at the same time. Typical algorithms include FCM [9], FCS [18].

4. Distribution based: clustering algorithms that assume there were multiple distributions in the original data, and data points sampled from same distribution belong to a same cluster. Typical algorithms include GMM [49].

5. Density based: clustering algorithms that assume data points in high density region belong to the same cluster. Typical algorithms include DBSCAN [22] and Mean-shift [17].

6. Graph theory based: clustering algorithms that regard data points as nodes and their relationship as edges in a graph. Typical algorithms include CLICK [54].

7. Grid based: clustering algorithms that transform original data space into a grid structure with fixed size. Typical algorithms include STING [60].

8. Fractal theory based: clustering algorithms that attempt to divide data points into multiple groups that share some common characters with the original data. Typical algorithms include FC [5].

9. Model based: clustering algorithms that pre-define a model for each cluster and mathcing data points best fitting for that model. Typical algorithms include COBWEB] [24] and SOM [31].

Most previous works try to improve the quality of generate clusters by modifying clustering algorithms. However, we doubt that with a proper data representation method, a simple clustering algorithm can generate acceptable results. Therefore, this project tries to improve the clustering result by finding a better representation method of stock price records.

2.5 Other Related Concepts

This section introduces the related concepts of this project. They are not specific techniques, but the theories behind various methods.

2.5.1 Machine learning

Machine learning is a branch of artificial intelligence, it can be define as a set of algorithms that can learn useful information from experience without human participation. Here the word "experience" standards for the available historical information, which are usually in the form of electronic data. With those algorithms, computers are able to solve various problems such as stock grouping, pattern discovery without explicit programming [41].

Machine learning techniques powers many of the services we use today, they are wildly used in various fields and industries. Some well-known applications build upon machine learning algorithms include the search engines (Baidu, Google), voice assistants (Siri), recommendation systems(YouTube), etc. Other more recent advanced applications include automatical programming (OpenAI Copilot), protein structure predicting (AlphaFold2). All such appli-

cations show a revolution brought by machine learning.

Machine learning has various branches, each branch adpots a unique learning paradigm. Common branches include: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, instance-based learning, etc. Supervised learning requires full label information to explicitly tell the machine what the patterns it should look like, it is usually applied in regression and classification tasks [56]. Unsupervised learning does not need ground truth information of the data, each unsupervised learning algorithm uses its own rule to extract and display interesting structures in the data. Those structures are preserved and used as the learned feature tp recognize the class of the new data. Unsupervised learning is mainly used for data clustering and feature reduction tasks. [39]. Semi-supervised learning is a combination of supervised learning and unsupervised learning, it requires some labelled data as the guideline and then can learn from unlabelled data. Typical semi-supervised learners are trained on labelled data, and then they are adjusted on the the validation of unlabelled data. Such method reduces the requirement of tedious labelling process [72]. Reinforcement learning studies on how an software agent can take actions in an environment to maximize a numerical rewards signal. Essentially, the learning process of agents is a closed-loop problem, since the the action taken by agents will influent the later input. In addition, in reinforcement learning, agents are not told to take which actions, they have to find the actions yielding the most rewards by searching in action space. This method is usually applied in decision making tasks. [57]. Instance-based learning does not require presuming a explicit model to describe data. Insetad, it makes predication purely on the similarity of the query (i.e. the new data) to its its nearest neighbor(s) in the training set. It works by simply storing all the data, computing and sorting the similarity scores, and using the score to assign class to new data when at query time [39].

2.5.2 Neural Networks

Artificial neural networks (ANNs or simply NNs) are computing systems that endeavor to mimic the animal brain operates to get the ability of recognizing underlying relationships in a set of data. The term "network" describesi that an ANN is a collection of artificial neurons organized in a specific way. The basic artificial neuron is the perceptron, its structure can be seen in Figure 2.3.

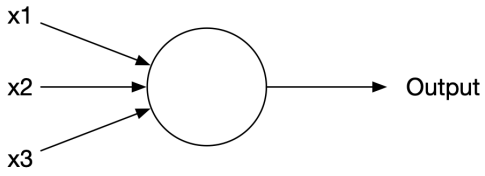


Figure 2.3: Perceptron

A simple perceptron loosely models a neuron in a biological brain, it works by taking several binary inputs $x_1, x_2, \cdots, x_n$ and producing a binary output [50]. The mathematical rule of computing the output is designed as the weighted sum of input. For each input x_i, it has a corresponding weight w_i expressing the importance of it to the output. Given the weighted sum of input $\sum_i w_j x_i = W^T X$, the perceptron determinates whether output 1 or 0 by a simple step function:

$$output = \begin{cases} 0 & \text{if } W^T X \leq threshold \\ 1 & \text{if } W^T X > threshold \end{cases} \tag{2.1}$$

If $W^T X$ is greater than the threshold value, the perceptron will output 1, otherwise 0. Obviously, such decision plane is quite naive, but it shows how a simple perceptron makes decision, and it makes sense to integrate more neurons to form a complex network. Figure 2.4 shows an simple feedforward network using perceptron as the neuron unit.

As can be seen, this NN has three columns (more formally, layers). The numerous in the first layer make three simple decisions by the formula listed above. Then, those decisions are also

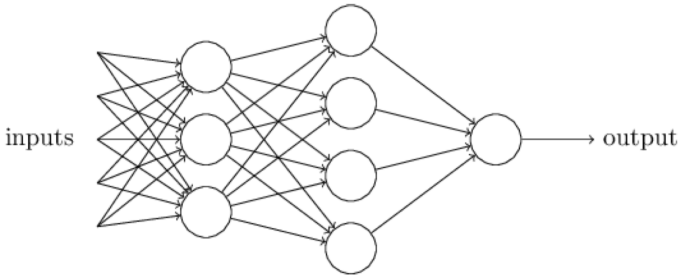


Figure 2.4: Simple feedforward network (source: [50])

weighted and accumulated together by the second layer. In this way, the second layer are said able to make decisions at a more complex and abstract level. The last layer is an output layer, it receives output from the second layer, and makes more complex decisions based on them. This is a quite simple example, the modern ANNs usually adpot more sophisticated neuron units and model structures, but their basic theories remain the same.

In previous example, the input X and the structure of the model are given, the only unknown are the model parameters (the weights) W. The training process of an ANN is therefore can be regarded as the process of find weights W that makes the best decisions. To guide how to find the best W, cost functions are introduced, they use mathematical equations to encode the real word target. One simple and intuitive example is regression problem, where the target is approximate the real distribution of the input data. For this problem, the cost function is usually defined as the distance (Euclidean, absolute value, etc.) between the target points and prediction. Given the cost functions are differentiable, the training of ANNs can be done by backpropagation [50]. Technically, backpropagation computes the partial derivative of the cost function with respect to the weights. The gradients guide the updating direction of parameters. Due to the complexity of the searching space, the training of NNs is inclined to reach local optimal rather than global optimal. But for real world applications, there may be no need of global optimal, most requirements can be achieved in a local optimal, therefore, ANNs are wildly used in recent applications.

2.6 Chapter Summary

This chapter introduces the background of this project, including the basic research problem, some important concepts, related literature, the research gap, and the main techniques used in this project. As the research subject of the project, stock market data are detailed introduced at the beginning, definitions of stocks, shares, stock market are given as complement to describe them. Real world examples are provided to show that the stock market data are usually in the form of time sequences, meaning that stock market data could be high-dimensional and continuous. Given the properties of stock market data, two major analysis methods are introduced. Directly forecasting the stock price trend is attractive but also challenging. A more promising method is extracting the universal patterns existing in various stock price sequences and making decisions upon them. Clustering is a typical method of finding patterns, therefore, researches applying clustering algorithms to analyze stock market data are reviewed. Those works mainly focus on clustering algorithms, none of them study on the effect of using different representations of stock sequences, and this motivates the project to experiment with different representations. Related techniques of transforming time-series data and clustering data are introduced. Finally, machine learning and neural networks are briefly introduced since they are highly relevant to this project.

Chapter 3

Research Methodology

This chapter introduces the research methodology selected in this project, including both high-level design and low-level tool for implementation.

3.1 Problem Definition

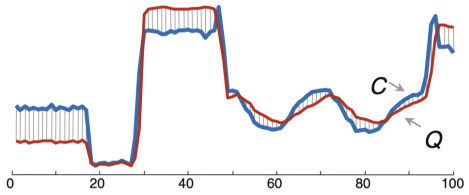
The main task of this project is to find an appropriate method to group stock market data. In this project, the stock market data are defined as $X = (X_1, \cdots, X_N) \in \mathbb{R}^{N \times T \times D}$, where $X_m = (x_{m_1}, \cdots, x_{m_T}) \in \mathbb{R}^{T \times D}$ represents all historical series of a single stock, N is the number of stocks, T is the number of time slices, D is the dimension of a single data. This task is further divided into 4 sub-tasks: (1) time-series data preprocessing; (2) time-series data transformation; (3) time-series data clustering; (4) experiments and evaluation.

3.2 Guideline for this project

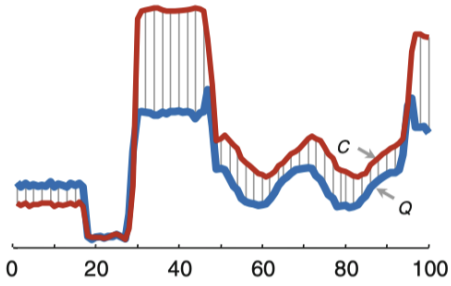
This project aims to group stocks by their annual trends, however, according to [8], sequences data are often distorted in some way in different domain, directly compare distorted sequences could be meaningless. To make the clustering results more informative, make the general trend more representative, this project follows a sequence comparison guideline. The guideline chosen here inherits from the known time-series invariances summarized by [46]. This project follows it to process stock data, find an appropriate method to group stock data and choose an appropriate metric to evaluate the clustering results. Followings are the details of the guideline.

1. Scaling and offset invariance: due to the different amplitudes (or scaling) and offset (or

translation), two sequences may not match well despite they have similar shape and trend. For example, given a sequence $\vec{x}$ and its variation $\vec{x'} = a\vec{x} + b$, where a and b are constants, their similarity to other sequences should be the same. This property is important especially for the comparison of two stocks with different price range and hence is required. Figure 3.1 shows an example of scaling and offset invariance, in these two subfigures, blue lines are the same, while red lines have different scaling. Both the red lines should be treated similar to the blue line, however, directly applying distance measure to the bottom example may not able to reflect that.



(a)



(b)

Figure 3.1: Example of scaling and offset invariance (Source: [8])

2. Local scaling invariance: given two sequences with same length, if they are aligned globally or locally, they should be considered as a similar pair. The term "global alignment" means the general trend of the two sequences are similar but differ in some subsequences. To the contrary, "local alignment" means there are some regions of two sequences match well while other parts do not. Such invariance is crucial for some trajectories with different speed (such as the movement of different people). Figure 3.2 shows an example of such invariances, these two sequences are regarded as similar. In terms of stock sequences, such invariance can help to group stocks with similar trend

but different trend duration together and hence is required.

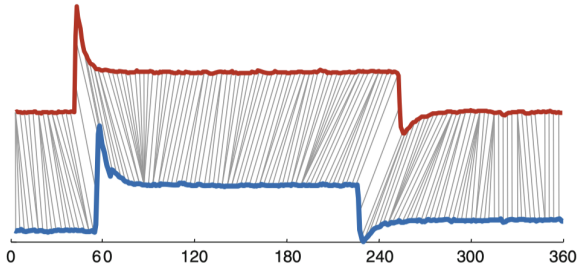


Figure 3.2: Example of local scaling invariance (Source: [8])

3. Global scaling invariance: different from local scaling invariance, global scaling invariance is about length-varying sequences. If two sequences have a similar shape but different length, they should also be grouped together. Figure 3.3 shows an example of global scaling invariance, CDC is a full gene expression and it matches poorly to the prefix of a related gene sequence, CDC15. However, if it is rescaled by a factor of 1.41, it can matches CDC15 more closely. In terms of stock sequences, due to numerous factors, the length of each records can be different, hence such invariance should be considered.

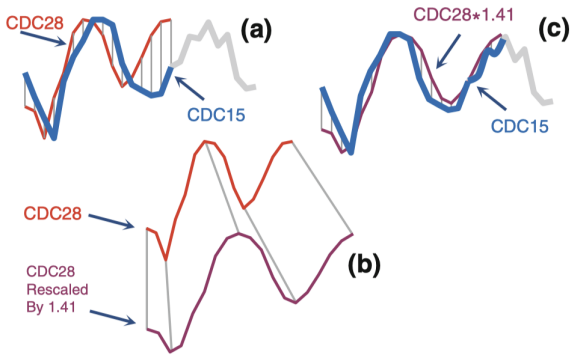


Figure 3.3: Example of global scaling invariance (Source: [8])

4. Complexity invariance: if two sequences have similar shape but different frequencies (complexity), they should be grouped together. This invariance is crucial for stock prices analysis, since the stock sequences contains massive short term small fluctuations that do not change the general shape but affect the comparison of stocks.

Those invariances are basic standards of how this project measures the similarity of two stocks. Following sections explain how to apply them in programming and experiments.

3.3 Stock Data Preprocessing

According to [46], scaling and offset invariance and complexity invariance can be achieved by data processing while other invariances can not. Therefore, this project adds a data preprocessing phase before futher analysis. The main steps include:

1. Data denoising: this step is highly associated with complexity invariance. According to the guideline, the small fluctuations on stocks should not impair the similarity measure. Commonly used distance measures such as Euclidean distance and Cosin distance can not deal with such fluctuations. This project uses data denoising (or called data smoothing) to alleviate the issue. The detailed method and its effect can be seen in Section 4.2.1

2. Data normalization: this step is highly associated with scaling and offset invariance. After normalization, sequence $\vec{x'} = a\vec{x} + b$ will be treated as the same with sequence x. Theoretically, both Z-Score and Min-Max normalization can achieve this, and their effect and comparison can be seen in Section 4.2.2

Note that based on the guideline, the final experiments and visualization are conducted on the pre-processed data rather than the raw data. For those invariances that can not be achieved by simple processing methods, this project seeks help from data representation algorithm and evaluation metrics.

3.4 Stock Data Representation

Feature engineering attempts to find the latent features of original data that can improve the performance of machine learning algorithms. By research, there are various traditional and novel representation algorithms for time-series data, each of them achieves certain invariances.

This project tries to apply them to this project to test whether they are suitable for trend-based time-series clustering tasks. The introduction of different representations can be found in Chapter 5.

3.5 Stock Data Clustering

In this project, the representative trends of stocks are produced by clustering algorithms. Different from other works that mainly test the performance of various clustering algorithms with the original sequences, our interest is that whether those traditional and novel representations with a simple cluster algorithm can produce a good result. Hence this project mainly uses one clustering algorithm KMeans. To get more universal observations ans conclusions, this project may use other clustering algorithms, but not in all experiments.

3.6 Evaluation

The evaluation of this project is conducted on each phase. This decision benefits the ablation study. For different phase, there could be different metrics and criteria, but in generally, they follow the above guideline. In preprocessing phase, both visual and numerical comparison are used to evaluate different methods since their effects are easy to understand. In representation phase, mainly the visual comparison is used, since there is no direct metric to compare different representation algorithms. The numerical comparison of different representations hence is conducted in the clustering phase. In clustering phase, both visual and numerical comparison are used, since the data doesn't have labels, indexes that does not require ground truth information are chosen as the metrics.

3.7 Tools for Implementation

This project is developed by Python programming language, since there are various out-of-box libraries for data analysis. The main libraries used to build this project include: Numpy,

Matplotlib, Scikit-learn, Tslearn, Seglearn, Pyts, Pywt and Pytorch. Numpy provides the basic operations for manipulating vectors and matrixes. Matplotlib provides numerous functions to display the data and the experimental results. Scikit-learn provides well-optimized implementation of basic machine learning algorithms, including the clustering algorithms such as KMeans, DBSCAN and some evaluation metrics. Tslearn, Seglearn and Pyts are libraries for time-series data preprocessing, they provide functions such as segmenting data, scaling the data and extracting statistical features. Pywt provides various sequence decomposition methods, and it is used to denoise the data. Pytorch provide functions to build, train and store ANNs, it can use GPU to boost the computation. Other functions are purely implemented by Python.

3.8 Chapter Summary

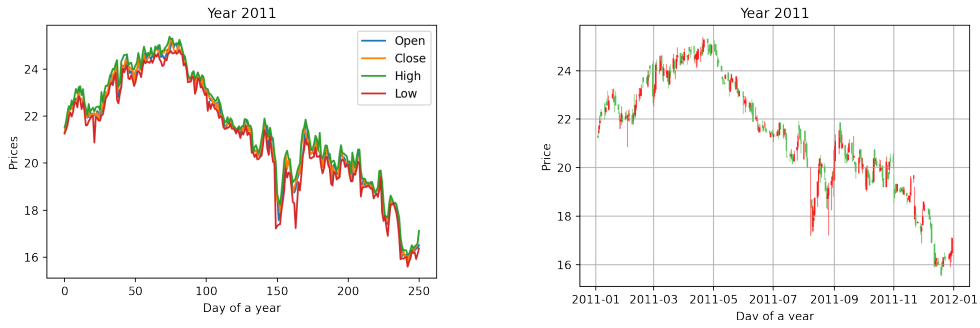
This chapter introduces the research methodology of this project. The basic research guideline is proposed based on the invariances of stock market data. This guideline indicates what kind of stock price sequences should be treated as similar. Then, according to the guideline, the preprocessing steps, representation methods, clustering algorithms and the evaluation metrics are discussed and selected. Finally, implementation tools of this project are introduced.

Chapter 4

Data Description and Preprocessing

4.1 Data description

This project, mainly investigates the stock market in America, since the relevant data are relatively complete and easy to obtain. In particular, the dataset used in this project is Huge Stock Market Dataset (accessed in May 8th 2021). This dataset contains full price and volume records of 7195 different stocks from 1977 to 2017 for all stocks trading on the NYSE, NASDAQ, and NYSE MKT. Each record has 6 attributes: **Date:** the date of that record; **Open:** the opening price of the stock at that day; **High:** the highest stock price of that day; **Low:** the lowest stock price of that day; **Close:** the closing price of the stock of that day; **Volume:** the total number of shares traded of that day. In the original dataset, there is an additional attribute "OpenInt", however, since there is no actual record of it, this attribute is discarded. In addition, the "Volume" attribute is not directly associated with stock price, therefore, it is also discarded in this project. The final representation of a single stock can be regarded as a 2-dimensional tensor with its first dimension corresponding to the time axis and the second dimension representing features. Note that due to some factors such as the regular suspension of stock market, there could be no records in some periods. Consider that mining the whole dataset could be time-consuming, merely 5 years records from Jan 1st 2011 to Dec 31th 2015 of 100 companies are used in this project. Figure 4.1(a) and 4.1(b) show the price curve and candlestick chart of 'IBA' stock in 2011 respectively, x-axis represents the trading days while y-axis represents the prices.



(a) Curve chart (b) Candlestick chart

Figure 4.1: Sample data

4.2 Time-series preprocessing

As mentioned in Section 3.2, the stock market data are expected to reformed with some invariant properties before fedding into clustering algorithms. According to [46], some of those invariances can be achieved by preprocessing the data. This section describes some common approaches to preprocess time series data and what methods are used to eliminate the certain distortions.

4.2.1 Time-series denoising

Sequence denoising is closely related to the **complexity invariance** mentioned in Section 3.2. Due to numerous factors, stock price sequences contains many noises (i.e. short-term volatility). Such noises could make some distance measurement inaccurate, for example, in terms of dynamic time warping (DTW), the unsmoothed curve with many fluctuations could make the alignment process broken. Therefore, noisy sequences may have an obvious impact on the performance of clustering algorithms [71]. For time series clustering problem, the continuous trend characteristics of data are hoped to be stable, and this can be achieved by filtering short-term noises. Since the time series data are non-linear and have high Signal to Noise Ratio, using traditional filters such as Gaussian filter and median filter could change the general shape of the sequences. In practical, the denoising of stock sequences is most done by Wavelet transform. Similar to fourier transform, Wavelet transform gives the frequency rep-

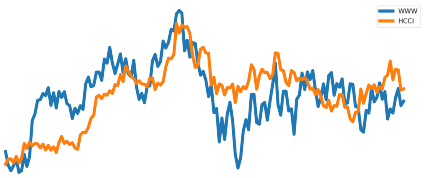
resentation of a signal. The main difference is that the standard Fourier transform is merely localized in frequency while wavelets are localized in both time and frequency [55], which means that with this transform, the low frequency band could have a lower time resolution and higher frequency resolution, while the higher frequency band could have a higher time resolution and lower frequency resolution [63]. The properties of Wavelet transform makes it appropriate for time series data decomposition.

Generally, the wavelet-based denoising process is done by **Discrete Wavelet Transforms (DWT)**. DWT decomposes the signal into mutually orthogonal set of wavelets, each set describes the time evolution of the signal in the corresponding frequency band [2]. Once the coefficients of each set is computed, they can be used to reconstruct the signal. Theoretically, noises are more sparse among all frequencies, the coefficients of important information are higher than that of noises, which makes the denoising possible.

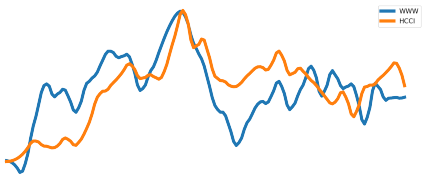
Figure 4.2 shows the general pipeline of DWT-based denoising process. The main parameters of using such approach are the wavelet function and decomposition layers. To verify the effectiveness of the denoising process, the original sequences are compared with the denoised version in terms of the similarity/distance measurement. Figure 4.3 visually proves that the denoising process improves the accuracy of sequences matching. In this set of figures, the records of two stocks with abbreviation "WWW" and "HCCI" in 2011 are used as an example. In general, these two stocks share a similar trend and are expected to have a small distance between them. As can be seen in Figure 4.3(a), the original sequences have many fluctuations, which makes the DTW alignment of two sequences inaccurate and disordered (see Figure 4.3(b)). After applying the denoising algorithm to them, these two sequences are smoothed while the general shapes are preserved, the number of points in each denoised sequence is also the same with the undenoised one. Moreover, it's obviously that the alignment result is improved. As mentioned before, the serration could make DTW alignment inaccurate, and the results in Figure 4.3(c) and Figure 4.3(d) show that smoothing the curve

can alleviate the problem. Numerically, the distance between the denoised sequences is 3.85, while that between the original sequences is 5.50, roughly 43% larger. Similar improvement can be found in use of other distance metrics. Table 4.1 shows the results of denoising process conducted on 20 similar sequences. 4 different distance metrics are used to computed the distance between each pair, where the "Cosine" and "Correlation" are $1 - \frac{X \cdot Y}{\|X\|\|Y\|}$ and $1 - \frac{(X-\bar{X})\cdot(Y-\bar{Y})}{\|X-\bar{X}\|_2\|Y-\bar{Y}\|_2}$. The values in the table are the arithmetical mean of 190 pairs. Note that the sequences used in this experiment are already normalised and compressed. As can be seen, all the four metrics are shortened, which proves that the denoising process can contribute to complexity invariance. **Same with [63], db8 with threshold parameter of 0.04 are chosen as the wavelet function.**

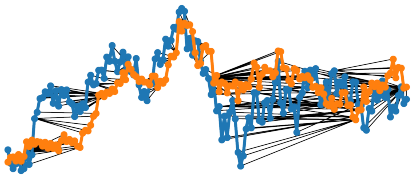
Figure 4.2: DWT-based denoising process



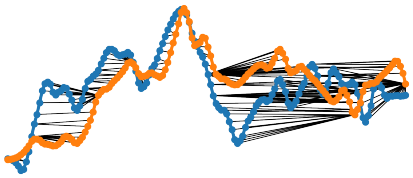
(a) Original sequences



(b) Denoised sequences



(c) DTW Alignment of original sequences



(d) DTW Alignment of denoised sequences

Figure 4.3: Sequences denoising results

	DWT	Cosine	Euclidean	Correlation
Original	3.442	0.168	6.743	0.168
Denoised	3.382	0.161	6.541	0.161

Table 4.1: Sequences denoising results

4.2.2 Time-serie normalization

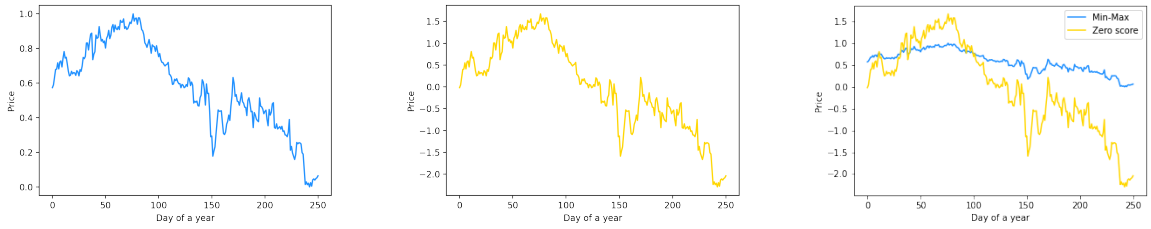
Sequence normalization is the most simple and effective way to ensure **scaling and offset invariance** [8]. Each stock could have a unique movement range of its price, and such range could vary during different time periods. To eliminate the influence of the variance of vertical scaling and offset, data normalization is required. This project mainly experiments with two most common normalization approaches used in machine learning – Min-Max normalization and Z-Score normalization. Min-Max normalization could scale the data into range [0,1] while preserving the general shape. The equation of this approach is listed in 4.1. It guarantees that all features will have the exact same scale. The main drawback of this normalization is that it can not handle outliers well. Assuming that there are 99 data points locating in value range [0,40] and one point has value 100, then the 99 points will be transformed into value range [0,0.4], leading to a squashed distribution. To the contrary, Z-Score normalization can avoid outlier issue. Its formula is listed in 4.2, where μ stands for the mean value of the feature and σ stands for the standard deviation of the feature. different from Min-Max normalization, this normalization could result in negative values: values greater than μ will be postive numbers while values less than μ will be negative numbers. The main drawback of Z-Score normalization is that it can not guarantee the transformed data have the exact same scale. Different from other data, for time series data, the normalization is usually applied to each individual.

Figure 4.4 shows the results of using these two approaches. As can be seen, both of them preserve the shape of original sequence (see Figure 4.1) in different scale. Mathematically, both of them can guarantee scaling and offset invariance. But in experiments, it can be found that using the exact same scaling could be inappropriate. Figure 4.5 shows an example of that. Sequence 1 (colored with blue) ranges between 3.0 and 5.5 while sequence 2 (colored with orange) ranges from 20.0 to 30.0. They shares a roughly similar shape when examine them individually (as shown in the top middel and top right). However, as you may find in Figure 4.5(a), if they are examined together, the difference is significant. This indicates that

scaling matters in time series tasks. After Min-max normalization, they locate in a same range (see bottom left), and hence they are categorized into a same cluster. When compare two stocks, such result is incorrect. **To take the scaling into consideration, this project chooses the Z-Score normalization as the normalization strategy.**

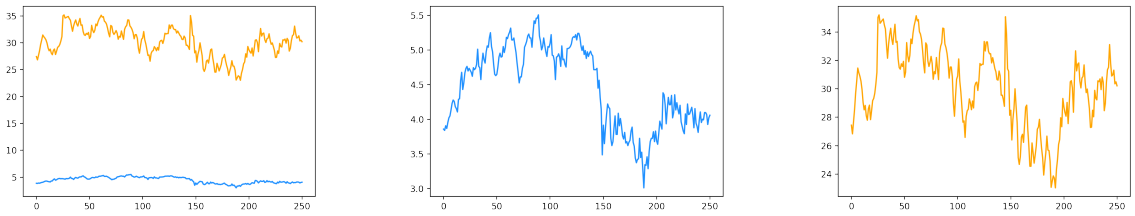
$$x'_i = \frac{x_i - \min_{i \le j \le n}\{x_j\}}{\max_{i \le j \le n}\{x_j\} - \min_{i \le j \le n}\{x_j\}} \tag{4.1}$$

$$x'_i = \frac{x_i - \mu}{\sigma} \tag{4.2}$$

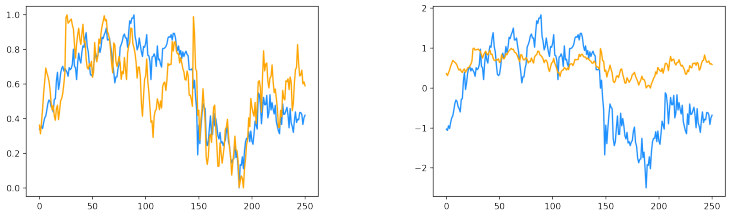


(a) Min-Max normalization (b) Z-Score normalization (c) Comparison

Figure 4.4: Results of normalization



(a) Original sequences (b) Sequence 1 (c) Sequence 2



(d) Min-Max normalization (e) Z-Score normalization

Figure 4.5: The effect of scaling

4.3 Chapter Summary

This chapter introduces the source of the stock market data used in the project and the data pre-processing steps. The data are stored in the form of tables with each table represents a unique stock and each line of a table represents a record of a stock at a certain time. Since the original data volume are quite large, merely 5 years records from Jan 1st 2011 to Dec 31th 2015 of 100 companies are used in this project. Note that the data have no label information, limiting the choice of evaluation indexes of clustering results. Since the stock market data are typical time-series sequences, according to the guideline, they can be denoised and normalised to achieve complexity invariance and scaling and offset invariance respectively. In terms of denoising, db8 with threshold parameter of 0.04 are chosen as the wavelet function. In terms of normalization, both Min-Max normalization and Z-Score normalization can guarantee scaling and offset invariance, however, Min-Max normalization makes the values of all stocks locate in an exact same range, which is inappropriate when compare different stocks.

Chapter 5

Time-Series Representations

As mentioned in Chapter 2, in many time-series classification tasks, a simple KNN with well-extracted features is able to produce a competitive results with other more sophisticated models. In recent years, various novel time-series representation algorithms are proposed, most of them outperform the previous baseline in the context of classification tasks. However, **there are rare studies about the effectiveness of them in the time-series clustering tasks**. By research, there are some representation methods can achieve certain invariances, and they are used in this project to test their effectiveness in terms of time-series clustering task.

Generally, time-series data representations can be categorized into three groups. As seen in Figure 5.1, there are three major ways to fed the preprocessed data into clustering models. The left group of approaches are mainly statistics-based, they generally require two steps: (1) extracting static features from the original sequences either globally or locally. Here "globally" means using the information of a whole sequence to form a feature vector, while "locally" requires segmenting the whole sequence into subsequences first, extracting the features individually and then concatenating the features to form a feature vector; (2) selecting features, filtering out features that may decrease the perfomance of clustering algorithms and reserving the most discriminative features. The right group of approaches represent clustering algorithms that directly take the processed time-series data as input, their perfomance is highly associated with the similarity measure. The middel block represents various representation schemes that try to project the original sequences into discriminative feature space, extract the hidden structure of the data. In this chapter, mainly some traditional and novel algorithms of the left and the middel blocks are introduced.

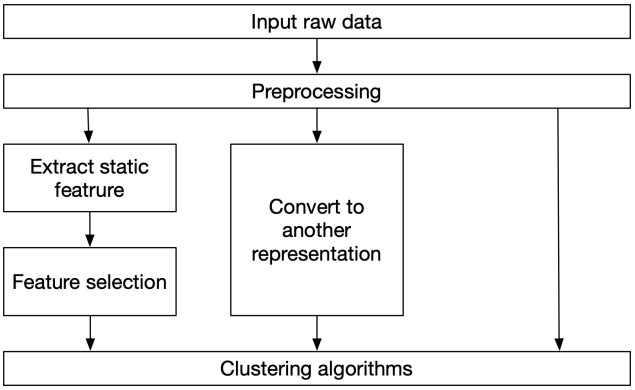


Figure 5.1: Approaches for time-series data clustering

5.1 Static Feature Representations

5.1.1 Statistical features representation

Statistical features are the most common and well-known features of time-series data. Compared with raw data, those statistics features are less sensitive to noise and distortion. Compared with other representations, this representation is easy to obtain. Due to these two attributes, statistical features representation is regarded as the baseline of time-series representation problems. [45] firstly examine this representation in time-series classification tasks, then, such representation is applied to clustering tasks.

According to [45], statistical features can be divided into two groups: (1) first-order features, which can be computed directly from the actual value of data; (2) second-order features, which are computed from the difference of nearby values. For time-series data, the second-order features are computed from a transformed time series $y_t\prime = y_{t+D} - y_t$ $(1 \leq t \leq n - D)$, where D is a user-defined gap between the points in the time series being compare. Some common statistical features include: (1) the mean value μ; (2) the standard deviation σ; (3) the skewness $SKEW$; (4) the kurtosis $KURT$. The latter two features indicate the shape of the sequence's value distribution. Skewness representation the magnitude of asymmetry (a sequence is symmetric if it looks the same to the left and right of the center point). Kurtosis indicates the degree of peakedness of a distribution relative to a normal distribution (a

distortion is flat if it has more outliers). Their equation are:

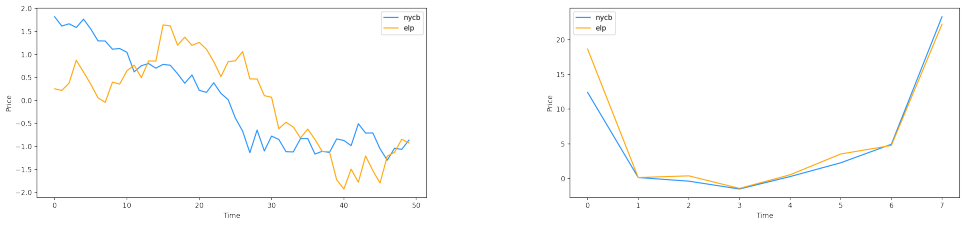
$$\mu = \frac{\sum_{t=1}^{n} y_t}{n} \tag{5.1}$$

$$\sigma = \sqrt{\frac{\sum_{t=1}^{n}(y_t - \mu)^2}{n}} \tag{5.2}$$

$$SKEW = \frac{\sum_{t=1}^{n}(y_t - \mu)^3}{n\sigma^3} \tag{5.3}$$

$$KURT = \frac{\sum_{t=1}^{n}(y_t - \mu)^4}{n\sigma^4} - 3 \tag{5.4}$$

Extracting the four features from y_t and $y_t\prime$ forms the first-order and second-order representation respectively. The first-order features solely can represent the original data, but it can also be concatenated with the second-order features to form a better representation. Figure 5.2 shows an example of this representation transformation. The samples used here are the records of stock "NCYB" and "ELP" in 2011. Both first and second order features are used, and the gap D is set to 5. Note that the features include the mean and standard deviation, and hence the samples here are not normalised.



(a) Original sequences (b) Statistical features representation

Figure 5.2: Example of statistical features representation

5.1.2 Bag-of-patterns representation

Bag-of-words (BOW) is one typical and traditional static feature representation method. It was wildly used in natural languages processing and image processing until more powerful representation learning algorithms came out. Similarly, time-series sequences can be represented by pattern histograms, and such representation is call **Bag-of-Patterns** [36].

Generally, BOW requires computing a p-by-q matrix X. In the context of document representation, p denotes the the number of unique terms in the whole corpus, q is the number of documents, and the element $X_{i,j}$ represents the frequency of the word i occurring in the document j. Document j then can be represented by the jth column of X. In the context of time-series representation, [36] define p as the number of unique SAX strings and q as the number of sequences. Different from sentences, sequences are consecutive and have no "delimiters", therefore, there is no obvious definition of "words". To solve this problem, [36] use sliding window to separate the original sequence into subsequences, and apply Symbolic Aggregate approXimation (SAX) to each frame. The intuition of using SAX here is that SAX is a symbolic representation for time-series and can convert each frame into a SAX word (a string), which follows the definition of BOW. The details of SAX are described in [35]. To be brief, SAX is developed on the PAA representation discussed before, it takes two parameters from users: (1) α: the size of the alphabet to use; (2) w: the size of the words to produce. Similar to PAA, it first splits the original sequences into w frames, and calculate the mean value within each frame. Each mean value (or called coefficient) is then mapped to specific a symbol based on a distribution that are divided into α equiprobable regions. Note that when the sequences are normalised, the distribution is more likely to be a Gaussian. Figure 5.3 shows an example of SAX transformation with α set to 5.

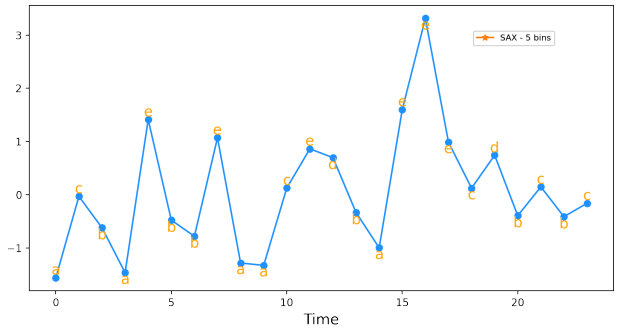


Figure 5.3: Example of SAX transformation

After the transformation, each frame is assigned with a string, and the "term-document" ma-

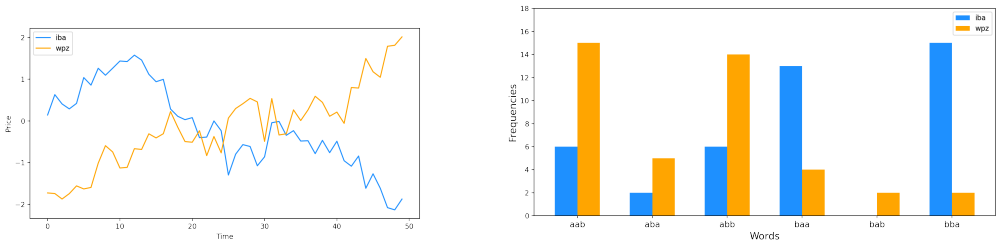
trix can be constructed. It should be noted that in some regions (especially smooth regions), neighbouring subsequences could be assigned with the same string, and this may result in over-counting issue. One feasible way to solving the problem is called Numerosity Reduction, which merely records the starting point of a set of consecutive subsequences that are mapped with the same string. One example of Numerosity Reduction provided by [35] is that, given a SAX transformed sequence S as follows:

$$S = aac\ acc\ abc\ abb\ abb\ abb\ abb\ bac\ baa$$

Its reduced form is defined as:

$$S\prime = aac_1\ abc_3\ abb_4\ bac_8\ baa_9$$

As stated by [36], the effect of this issue highly depends on the data. In terms of sequences that a generally smooth, dealing with the over-counting problem is required. For other sequences, that may not necessary. Figure 5.4 shows an toy example of Bag-of-Patterns transformation, for better comparison, the result is expressed in a bar chart. The samples used here are the processed sequences of "IBA" and "WPZ" in 2011. Their BOP representations are quite different, which is comply with their real trend.



(a) Original sequences (b) BOP representation

Figure 5.4: Example of Bag-of-Patterns representation

5.1.3 Path signature representation

Path signature [14] is a relatively novel representation method for non-smooth paths, it is the core part in rough path theory. The term "rough path" denotes a sequence that is continuous but highly oscillatory and therefore has no derivatives of all orders. One typical example is a stock price stream, which can be regarded as a result of Brownian movement. In the rough path theory, each rough path can be decomposed into a set of infinite path integrals, and that set is called the signature of the path. A brief mathematical definition could be: given a sequence (continuous or discrete) $P = (P_1, P_2, \cdots, P_T) \in \mathbb{R}^{T \times D}$, where T is number of points in the sequence, D is the dimension of each data point, let P_t^i denotes the i-th attribute of point P_t $(1 \leq t \leq T, 1 \leq i \leq D)$, the whole sequence can be represented by n-fold iterated integral over the sequence (n could be infinite). The simplest case is that when $D = 1$, the 1-fold representation of P is a real value defined as:

$$S(P)^1_{1,T} = \int_{1<t\leq T} dP_t^1 = P_T^1 - P_1^1 \tag{5.5}$$

Similarly, the 2-fold representation is also a real value:

$$S(P)^{11}_{1,T} = \int_{1<t\leq T} S(P)^1_{1,T} dP_t^1 = \frac{1}{2}(P_T^1 - P_1^1)^2 \tag{5.6}$$

The k-fold representation is:

$$S(P)^{11\cdots 1}_{1,T} = \frac{1}{k!}(P_T^1 - P_1^1)^k \tag{5.7}$$

The final signature representation of sequence P is the vector $(S(P)^1_{1,T}, S(P)^{11}_{1,T} S(P)^{11\cdots 1}_{1,T}) \in \mathbb{R}^k$, whose dimension k is the number of fold wanted by users.

When D = 2, 1-fold representation has 2 elements defined as follows:

$$\begin{aligned} S(P)^1_{1,T} &= \int_{1<t\leq T} dP_t^1 = P_T^1 - P_1^1 \\ S(P)^2_{1,T} &= \int_{1<t\leq T} dP_t^1 = P_T^2 - P_1^2 \end{aligned} \tag{5.8}$$

2-fold representation has $D^2 = 4$ elements:

$$\begin{aligned} S(P)^{11}_{1,T} &= \int_{1<t\leq T} S(P)^1_{1,T} dP^1_t = \frac{1}{2}(P^1_T - P^1_1)^2 \\ S(P)^{22}_{1,T} &= \int_{1<t\leq T} S(P)^1_{1,T} dP^1_t = \frac{1}{2}(P^2_T - P^2_1)^2 \\ S(P)^{12}_{1,T} &= \int_{1<t_1\leq T} \int_{1<t_2\leq T} dP_{t_1} dP_{t_2} \\ S(P)^{21}_{1,T} &= \int_{1<t_2\leq T} \int_{1<t_1\leq T} dP_{t_2} dP_{t_1} \end{aligned} \tag{5.9}$$

In general, when D = d, the i-th element of k-fold representation can be seen in eqaution 5.10, where $(n_1, n_2, \cdots, n_k) \in \{1, \cdots, k\}$:

$$S(P)^{n_1,n_2,\cdots,n_k}_{1,T} = \int_{1<t_k\leq T} \cdots \int_{1<t_3\leq t_4} \int_{1<t_2\leq t_3} dP^{n_1}_{t_1} dP^{n_2}_{t_2} \cdots dP^{n_k}_{t_k} \tag{5.10}$$

The final signature of the whole path P is the collection of all elements in every fold defined as follows:

$$\begin{aligned} S(P)_{1,T} = (1, &S(P)^1_{1,T}, \cdots, S(P)^D_{1,T}), \\ &S(P)^{1,1}_{1,T}, \cdots, S(P)^{2,1}_{1,T}, \cdots, S(P)^{D,1}_{1,T}, \cdots, S(P)^{D,D}_{1,T}, \\ &S(P)^{1,1,\cdots,1}_{1,T}, \cdots, S(P)^{n_1,n_2,\cdots,n_k}_{1,T}, \cdots, S(P)^{D,D,\cdots,D}_{1,T}, \cdots) \end{aligned} \tag{5.11}$$

Conventionally, the first term is set to 1. With the increment of fold k, the collection could be infinite. However, in practice, there is no need of using a large k, users can decide the k based the the dimension formula of the final vector: $\omega(D, k) = (D^{k+1} - 1)(D - 1)^{-1}$. The truncated elements are regarded as the path signature feature of the original sequence, and then can be used in further analysis. A promising property of path signature representation is that each term of it has a physical meaning and can reveal the shape information of the path. For example, the first order terms $S(P)^i_{1,T} = P^i_T - P^i_1$ $(i = 1, \cdots, k)$ denotes the increments of the path. Higher order terms can be interperted as generalised polynomials of paths [15].

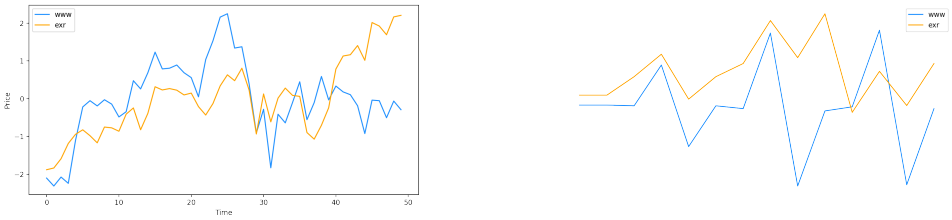
It's worth noting that path signature should be applied to multi-variable sequences since from the eqaution, higher order term $S(P)_{1,T}^{1_1,\cdots,1_k} = \frac{1}{k!}(P_T^1 - P_1^1)^k$ is merely an function of the increment $P_T^1 - P_1^1$. To apply this representation method to univariate sequences, one feasible way is adding at least one new feature. This new feature could be the time axis, or the features generated by lead-leg transformation. The equations below show the process of lead-leg transformation. As stated by [15], such transformation makes path signature able to capture the quadratic variation of sequences, which is crucial in finacial applications.

$$P_j^{lead} = \begin{cases} P_{t_i} & \text{if j=2i} \\ P_{t_i} & \text{if j=2i-1} \end{cases} \tag{5.12}$$

$$P_j^{lag} = \begin{cases} P_{t_i} & \text{if j=2i} \\ P_{t_i} & \text{if j=2i+1} \end{cases} \tag{5.13}$$

$$P^{lead-lag} = (P_j^{lead}, P_j^{lag})_{j=0}^{2T} \tag{5.14}$$

Figure 5.5 shows an example of path signature representation with lead-lag transformation. The sample used here are the records of stocks "WWW" and "EXR" in 2011.



(a) Original sequences (b) Path signature representation

Figure 5.5: Example of path signature representation

5.2 Dynamic Feature Representations

In contrast to static feature representation methods that produce identical results each execution, the middel block transforms sequences mainly by randomization and optimization.

This section introduces two major and novel dynamic feature representation methods for time-series data.

5.2.1 Random convolutional kernels based sequence representation

Convolutional neural networks are typically used in computer vision tasks, their success implicits that convolutional kernels may also be effective for extracting the features from time-series data, since in essence, time-series data have the same topology as images [23]. Different from transforming sequences with pre-defined patterns, convolutional neural networks can automatically detect patterns by learning. In addition, as stated by [21], the learned kernels could also overlap with many types of features found in other methods, one supporting example is that they find that some of the learned kernels are similar to the extracted Shapelet patterns.

For a convolutional kernel, the most important parameters are its weights, and the process of updating weights can be treated as the extraction of features. Once the weights are determined, those kernels can be used to transform data. This process is done by using sliding dot product operation on the whole data, to produce a feature vector. For time-series data, a kernel is usually a 1-dimensional vector, after the convolution operation between the kernel and the input sequence, an additional bias term will be added. Apart from the two parameters weights and bias that are usually learned by updating, there are some user-defined parameters: (1) "Size" decides the length of the kernel; (2) "Dilation" decides the sampling gap of convolution operation; (3) "Padding" determinates whether appending values to the start and end the input sequence. These three parameters together decide the dimension of the produced vectors.

Typically, the weights updating process requires labelled data, hence convolutional kernel based methods are mainly used in classification tasks. However, researchers such as [48, 52]

prove that random generated kernels can also be effective. This observation extends the use of convolutional kernels to unsupervised representation tasks. One possible way to make unsupervised kernels competitive with supervised kernels is increasing the number of random kernels. Due to the fact that those kernels' weights are not learned, their computation efficiency can be relatively high, which makes it possible to use a large number of kernels. One state-of-the-art time-series representation algorithm inheriting from such observation and idea is **ROCKET (RandOm Convolutional KErnel Transform)** [21]. It transforms time-series sequences by using a large set of kernels with random parameters (include all the 5 parameters mentioned above). The wide variety of kernels (especially kernels with different dilation and size) is said able to capturing patterns at different frequencies and scales. Similar to CNNs, ROCKET uses max pooling to sample features, besides, it uses an additional indexe - the proportion of positive values, to show whether a given pattern can be found within a time-series. For ROCKET, there is merely one user-defined parameter: the number of kernels k. Figure 5.6 shows an example of random kernels generated by ROCKET.

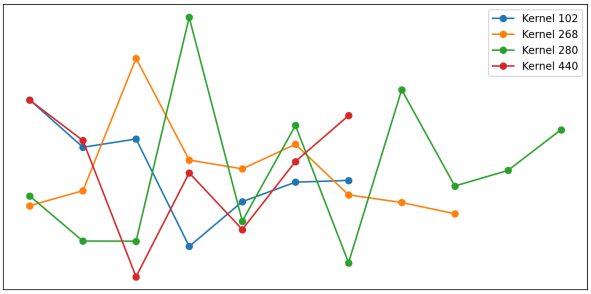
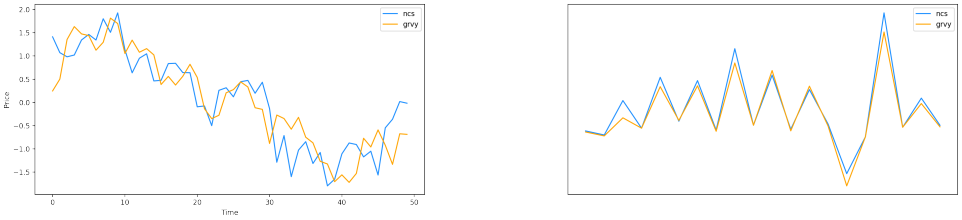


Figure 5.6: Example of random convolutional kernels

Once the random kernels are generated, they can be used to transform the original sequences by convolution operation. Given a 1-d kernel ω with dilation d and size m, and a sequence $X = (x_1, \cdots, x_T) \in \mathbb{R}^T$, the convolution operation from x_i is defined as:

$$x_i * \omega = \sum_{j=0}^{m-1} x_{i+(j\times d)} \times \omega_j \tag{5.15}$$

Each transformed feature map is then aggregated into two real value features: (1) the maximum value of it; (2) the proportion of positive value of it. And the final dimension of the transformed sequence is $2k$. One sample result of applying ROCKET to the records of "NCS" and "GRVY" in 2011 can be seen in Figure 5.7.



(a) Original sequences (b) ROCKET representation

Figure 5.7: Example of ROCKET representation

5.2.2 Self-supervised time-series representation learning

Autoencoders are the core of various unsupervised learning tasks, they aim to extract the hidden structure of the input data, and transform the data into another representation with the least possible amount of distortion [4]. A typical autoencoder has two components: (1) an encoder that transform the original data into a new feature sapce; (2) an decoder that transforms the encoded sequences into output vectors. The traditional objective of an autoencoder is to minimize the difference of the output data and input data, and this is called reconstruction loss. Researchers such as [40] and [10] adpot this scheme to transform time-series data, except that they use more sophisticated neural networks as the encoder and decoder.

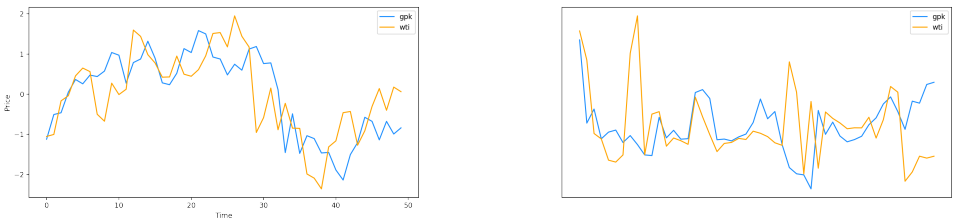
Recently, due to the success of transformer architecture in unsupervised pre-training tasks, a relatively novel auto encoding objective **Masked Value Prediction (MVP)** is proposed by [67]. This objective is analogue to Masked Language Modeling (MLM) of BERT, the main difference is that it is designed for time-series data. Literally, MVP works by setting some regions of the input sequence to 0 and ask the model to predict the masked values. Given an

arbitrary sequence $X_m = (x_{m_1}, \cdots, x_{m_T}) \in \mathbb{R}^{T \times D}$ as defined in Chapter 3, MVP will produce an independent binary noise mask $M_m \in \mathbb{R}^{T \times D}$ for it. The mask has the same size with the input sequence, and the masking process is done by elementwise multiplication. A proportion r is set to control the number of element (data point including all its features) to be maksed. It is worth noting that the authors do not simply use a Bernoulli distribution with parameter r to randomly mask the whole sequence, since such approaches could not guaranteed masked subsequences are long enough. They state that very short masked sequences can decrease the robustness of the model. Therefore, instead of using a Bernoulli distribution, they choose the state transition probabilities to decide the elements to be maksed. The length of each maksed segment is sampled from a geometric distribution with mean l_m, and each masked segment is succeeded by an unmasked segment of mean length $l_u = \frac{1-r}{r} l_m$ [67]. With MVP, the head or the output layer can be a simple linear layer with parameters $W_o \in \mathbb{R}^{T \times N}$, $b_o \in \mathbb{R}^N$, where N is the dimension of the final sequence representation (the output of the last layer) $z_t \in \mathbb{R}^N$. For any sequence X, the model will output its estimate X'. However, merely the prediction on the maksed regions will be used to compute the Mean Squared Error loss. Let $M = \{(t, i) : m_{t,i} = 0\}$ denotes the set of marked data points, the equation of MVP objective can be seen as follows:

$$X' = W_o z_t + b_o \tag{5.16}$$

$$\mathcal{L}_{MSE} = \frac{1}{|M|} \sum_{t,i \in M} \sum (X\prime_{t,i} - X_{t,i})^2 \tag{5.17}$$

After the training, each sequence can be represented by its corresponding z_t. Theoretically, MVP can be applied to almost all neural networks. For better comparison, the Time Series Transformer (TST) that used by the authors are chosen as the backbone. Figure 5.8 shows an example of this transformation. The samples used here are the processed records of stock "GPK" and "WTI" in 2011. The relationship (both global and local) of the new representations is roughly comply with the original representations, except that the weights of different regions are uneven.



(a) Original sequences (b) TST transformation

Figure 5.8: Example of self-supervised representation learning

5.3 Shape-Preserved Representations

All the representation algorithms introduced above transform the sequences into a new feature space and hence are hard to explain in visual. Therefore, another type of representation methods called shape-reversed representation are introduced. Such methods are usually compression-based, they are built on the observation that human can recognize a sequence even if that sequence are truncated or shortened. Note that compression-based methods are indeed can be categorized into Statistical Feature Representation, an independent section are added here because of their specialness. Section 2.4.1 briefly introduces the three main sequence reduction algorithms. The first and simplest approach is resampling, which alters the frequency of the original data by interpolation. There are two types of resampling: upsampling and downsampling. The former one will increase the length of samples while the latter one will decrease the frequency of samples. This project experiment with the **Linear Downsampling** approach, which simply choose n data points with a fixed step from the original data. This transformation doesn't consider the structure of the original sequence and hence could lose some important information, but due to its simpleness, it is wildly used in many applications such as large sequence data storing. Note that the linear downsampling approach will not alter the original value.

One more advanced approach is Perceptually **Important Points (PIP) based sampling**. PIP based approaches are developed on the observation that human can recognize a sequence even when merely a small set of points of that sequence are given. [66] stated that the process

of identifying PIPs is similar to the process of human cognition, where the general shpae of the data will be first recognized and the the details. In the context of data mining, PIPs are mainly used for dimension compression or sequence segmentation. The general process of this algorithm is introduced in Section 2.4.1, here mainly present the equations to find salient points. In PIP identification process, three distance metrics are used: the Euclidean distance d_E, the perpendicular distance d_P and the vertical distance d_V. In a simple example, let $X = \{x_1, x_2, \cdots, x_l\}$ denotes a 1-d sequence of length l, $x_t = (t, x_t)$ and $x_{t+T} = (t+T, x_{t+T})$ denote two adjacent PIPs. The Euclidean distance d_E of an intermediate point $x_i = (i, x_i)$, $t < i < t+T$, is defined in Equation 5.18.

$$d_E(x_i, x_t, x_{t+T}) = d_E(x_i, x_t) + d_E(x_i, x_{t+T}) \tag{5.18}$$

The vertical distance and perpendicular distance are computed from the line across the two PIPs. Assuming that (i, z_i) is a point on that line, $z_i = si + c$, the slope of that line is $s = \frac{x_{t+T} - x_t}{T}$ and the constant term is $c = x_t - st$, the perpendicular distance d_P and the vertical distance d_V between the intermediate points and the two PIPs are defined in Equation 5.19 and 5.20 respectively. The next PIP is select as the intermediate point $x_i^* = (i^*, x_{i^*})$ that maximizes the three distances. The equation can be seen in Equation 5.21.

$$d_P(x_i, x_t, x_{t+T}) = \frac{|si + c - x_i|}{\sqrt{s^2 + 1}} \tag{5.19}$$

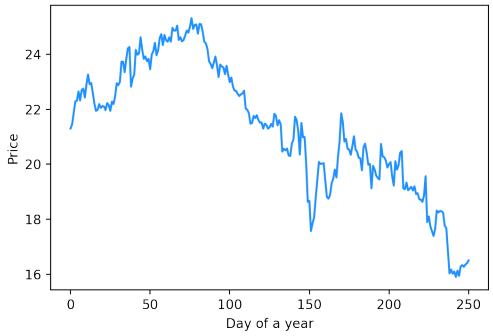
$$d_V(x_i, x_t, x_{t+T}) = |si + c - x_i| \tag{5.20}$$

$$i^* = arg\max_i(d(x_i, x_t, x_{t+T})) \tag{5.21}$$

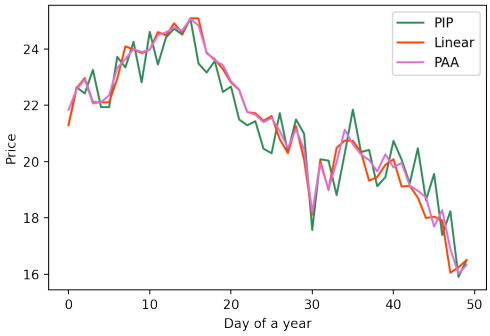
According to [66], PIP-based compression approaches could alleviate the effect of missing data as long as the main points to form the sequence exist. This property can help to reach global scaling invariance. The main drawback of this approach is that the distribution of salient points could be uneven, which may change the shape of certain sequences. Note that similar to linear downsampling, this approach will not change the original value.

The third compression approach is **Piecewise Aggregate Approximation (PAA)**. As introduced in Section 2.4.1, this approach will divide the data into n equal-sized segments, and use a vector of mean values of segments to represent the original data. The compressed data can be represented as $P\prime = p_1\prime, \cdots, p_n\prime$, where the i_{th} element of $P\prime$ can be calculated by Equation 5.22, where N is the length of original sequence.

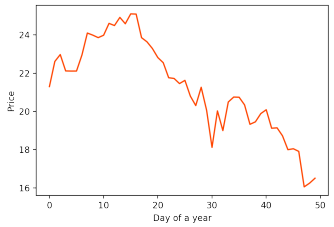
$$p_i\prime = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} p_j \tag{5.22}$$



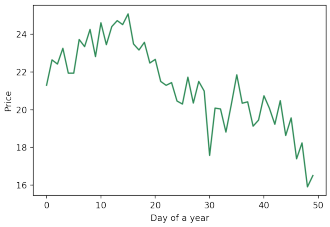
(a) Closing price of IBA in 2011



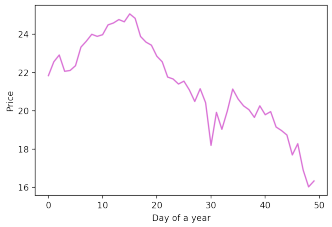
(b) Comparison of 3 approaches



(c) Result of linear downsampling



(d) Result of PIP



(e) Result of PAA

Figure 5.9: Compression results (n=50)

Figure 5.9 shows the results of using these approaches. Similarly, the sample used here is the un-modified closing price sequence of IBA in 2011 (Figure 5.9(a)). The general comparison of the three approaches can be seen at the top right, where the green line represents the PIP-compressed sequence, orange line represents the linear-downsampled sequence and the purple line is the PAA-compressed sequence. The original sequence contains 251 data points

while that of the compressed sequence is set to 50, the compression ratio $r = n/N$ is roughly 20%. The figures in the second row are the results of the three approaches respectively. From the second row, we can find that all the three approaches can produce recognizable shapes of the original sequence. Although the global shapes of the three modified sequences are similar, the local shapes can vary - PIP reserves more fluctuations and turning points while the others does not. As shown in the top right figure, linear downsampling and PAA based compression overlap at most regions, especially in the region between 0 and 30. Both of them smooth the region between 15 and 20 while PIP reserves some points of that region. As mentioned before, downsampling has no idea of which point is more important, hence it can miss some turning points, and this is proved in region 35 to 50. PAA can be treated as an mean filter in terms of denoising, and the drawbacks are discussed in Section 4.2.1.

To compare the three method objectively, more experiments are conducted based on the similar between sequences. The results can be seen in Table 5.1. Same with the experiment conducted in Section 4.2.1, 20 similar stocks from our dataset are randomly chosen as the samples, and use the mean values pair-wise distances as the metrics. One difference is that in this experiment, data are not normalised. The experimental results shows that there is no significant difference among the three approaches. One interesting observation is that the simplest downsampling method gets the highest markers on 3 indexes, especially on DWT, which is thought be the most appropriate index under the guideline. It's worth noting that

	DWT	Cosine	Euclidean	Correlation
Downsampling	177.077	0.020	180.243	0.121
PIP	185.685	0.020	189.590	0.193
PAA	188.722	0.020	191.760	0.137

Table 5.1: Sequences compression results

the choosing of the sequence length "c" is a classical tradeoff between the fidelity and dimensionality. According to [30], the "best" compression rate highly depends on the internal structure of the data and the application. The effect of different sequence length is examined in clustering results.

5.4 Chapter Summary

This chapter summarizes three basic categories of time-series data representation approaches and introduces the representatives of them. Those three categories can be further classified into two groups: (1) Shape-Changed representations; (2) Shape-Preserved representations. Here the terms *Shape-Changed* and *Shape-Preserved* denote whether the transformed sequences have the different or similar visual look with the original sequences respectively. Based on whether the representations are reproducible and fixed, they can also be classified into (1) Static feature representations and (2) Dynamic feature representations. Here *Static* means the features are the intrinsic properties of the original data, and the computation of them are fixed (i.e. their values won't changed in different run). To the contrary, *Dynamic* means the features are learned or generated in random, meaning that their values could change in different run. These four different groups have overlaps, however, as far as we know, most Shape-Preserved representations are static and therefore, merely 3 categories are listed.

In terms of static feature representations, three approaches are introduced: (1) Statistical Feature Representation, which uses some common statistical features such as the mean value and standard deviation of a sequence to represent it; (2) Bag-of-Patterns Representation, which transforms a sequence into a string and uses word histogram to represent the sequence; (3) Path Signature Representation, which decomposes a sequences based on rough path theory, and uses a subset of decomposed features to represent the original sequence.

For dynamic feature representations, two most advanced representatives are introduced: (1) ROCKET, which randomly generates various convolutional kernels and applies them to a sequence to form a feature vector; (2) MVP, which is an self-supervised learning paradigm for time-series data, it forces the model to predict the maksed values based on the context and uses the intermediate vector (i.e. the output of the encoder) as the feature vector of the

original sequence.

For shape-preserved representations, three common time-series compression approaches are introduced: (1) Linear downsampling, which samples a sequence with a fixed interval; (2) PAA, which separates a sequence by sliding window, and used the average mean of each segment to represent the sequence; (3) PIP, which samples a sequence by the rank of the importance of data points.

It is worth mentioning that most of the mentioned approaches have been approved effective in time-series classification tasks, but there is rare studies about their effectiveness in time-series clustering tasks.

Chapter 6

Semi-Shape-Preserved Clustering Representation

This chapter introduces the proposed novel representation method that are temporarily named as **Semi-Shape-Preserved Clustering Representation (SSPCR)** and how it is designed according to the guideline of the project. *Semi-Shape-Preserve* means that the generated feature vector contains the shape information of the original sequence while does not have the same visual appearance with it. *Clustering* means that this method is designed for clustering tasks.

6.1 Preliminaries

6.1.1 Reformulation of within-cluster-scatter

The main objective of traditional clustering algorithms such as K-means is to minimize the total squared error between the data points and their representative center, this objective is termed as **within-cluster-scatter**. Let x_i denotes a m-dimensional data vector, $i = 1, \cdots, n$, $X = [x_1, \cdots, x_n]$ denotes the $m \times n$ data matrix, Π denotes the partition results, k denotes the number of cluster, the objective function of K-means can be defined as below:

$$\mathcal{L}_{clustering} = ss(\Pi) = \sum_{i=1}^{k} \sum_{s=1}^{s_i} \left\| x_s^{(i)} - m_i \right\|^2 \tag{6.1}$$

$$m_i = \frac{\sum_{s=1}^{s_i} x_s^{(i)}}{s_i} \tag{6.2}$$

$x_s^{(i)}$ represents the sth data vector in cluster i, and m_i represents the mean vector of cluster i. According to [68], the minimization of $ss(\Pi)$ can be reformulated as a maximization of the trace of the Gram matrix X^TX with spectral relaxation. Given the cluster indicator matrix $F \in \mathbb{R}^{n\times k}$, the Equation 6.1 can be re-written as:

$$\mathcal{L}_{clustering} = ss(\Pi) = tr(X^TX) - tr(F^TX^TXF) \tag{6.3}$$

tr represents the function to compute the trace of a matrix. If the data matrix X is fixed, the minimization of $ss(\Pi)$ can be relaxed to finding a F with the constraints: (1) F is an orthogonal matrix; (2) $F^TF = I$; (3) the trace of matrix F^TX^TXF if maximized. The mathematical equation is defined as:

$$\max_F \ tr(F^TX^TXF), s.t. F^TF = I \tag{6.4}$$

This equation shows that the F has a closed-form solution. Based on the Ky Fan theorem [68], there is an optimal F that can be obtained by SVD decomposition of the data matrix X and composing the first k singular vectors. It's worth mentioning that this reformulation can lead to a global optimal solution, which may not be obtained with traditional approaches.

6.1.2 Autoencoders

As mentioned before, an autoencoder is a special neural network designed for dimension reduction/extension and feature extraction. A typical autoencoder consists of two components: (1) the encoder, which is in charge of transforming the original data vector into a new feature space; (2) the decoder, which takes the encoded vector and tries to reconstruct the original vector. Figure 6.1 shows the structure of traditional autoencoders. Some traditional autoencoders adpot a symmetric structure, where the encoder and decoder are mirror neural networks and share the same weights, but this is not required for all autoencoders. Generally, autoencoders generate two types of representations based on the dimensionality settings: (1) under-complete, when the dimension of the transformed vector is lower than that of the orig-

inal vector; (2) over-complete, when the dimension of the transformed vector is higher than that of the original vector. First type can be used for data compression while the second type may capture intrinsic structures underlying data. The learning process of the representation is done by minimizing the reconstruction loss, given x_i is the original vector and $\bar{x_i}$ is the transformed vector, the equation is defined as:

$$\mathcal{L}_{reconstruction}(X) = \frac{1}{2|X|} \sum_{i=1}^{|X|} \|x_i - \bar{x_i}\|^2 \tag{6.5}$$

There are more sophisticated autoencoders with other constraints, but in general, they share the basic paradigm.

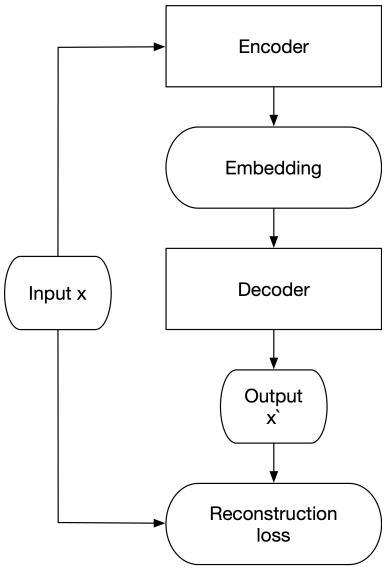


Figure 6.1: Structure of traditional autoencoders

6.2 Algorithm Description

In previous chapter, some shape-preserved and shape-changed representations are introduced, all those approaches are not designed for clustering tasks. As discussed before, shape-preserved representations are mainly static, and they indeed do not transform the new vectors into a new feature space. To the contrary, shape-changed representations have the ability to reveal the intrinsic properties of the original data, however, the transformed

vectors could have quite different visual appearances with the raw sequences, leading the loss of shape information. Both of the two categories of representation approaches have their own advantages and disadvantages, and this motivates the proposition of SSPCR. The design guideline behind the proposed learning paradigm is to make the transformed vectors more distinguishable while preserving the most shape information.

When design SSPCR, the first question is how to force the representation to be encoded with the shape information. Since the dimension of the transformed vectors are usually different with that of the raw data vectors, directly approximating the original sequences are not feasible. Inspired by [45], the general shape of a time-series sequence can be represented by a set of statistical features, this gives a hint that **if a transformed vector has the similar statistical features with that of its original vector, it can be said to preserve some of the shape information**. To make the learning paradigm appropriate for clustering tasks, the reformulation of within-cluster-scatter (see Section 6.1.1) is introduced. This reformulation proves that given the data matrix, there is an optimal partition leading to the smallest within cluster scatter (i.e. the mostcompact clusters). More important, it shows how to get the best clustering result in terms of that objective function. This gives a hint that **if the transformed vectors have the smallest within cluster scatter computed by the reformulated equation, they can be said to be appropriate for clustering**. Having these two hints, the last question is how to get the representation. There may be a global optimal solution, but it could be hard to compute. Instead, **SSPCR treats finding the best representation as an optimization problem and adpots autoencoders with gradient descent to find a local optimal solution**.

Figure 6.2 illustrates the how SSPCR works. SSPCR adpots the basic structure of the traditional autoencoders, expect that there are two more constraints: the shape loss and the clustering loss. In the training process, given a batch input vectors X, SSPCR will first extract their statistical features S_X, and then fed the raw sequences into the encoder. Then

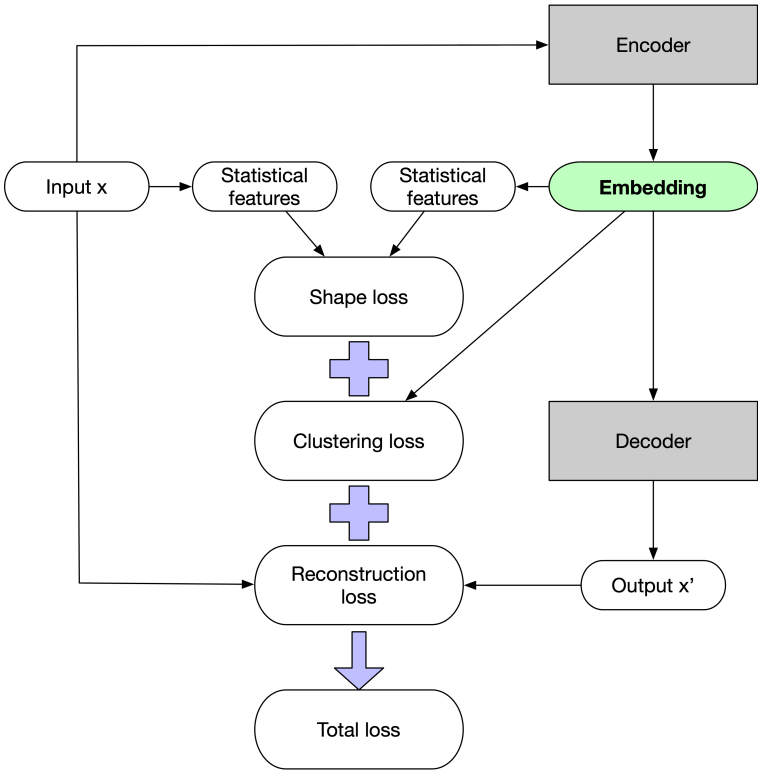


Figure 6.2: General structure of SSPCR

the encoder will map the original input sequences into a new feature space to get the new representations/embeddings of that data. The statistical feature set S_E of the embeddings will be extracted and used to compute the shape loss. In the meanwhile, the clustering loss of that data batch X will be computed with the Equation 6.3. Follow the steps of common autoencoders, the embeddings will be fed into the decoder, where they will be used to reconstruct the original sequences. The reconstructed output X' combined with the original sequences X will then be used to compute the reconstruction loss based on the Equation 6.4. Finally, those three loss will be summed together as the objective function for the training process, the equation is defined as:

$$\mathcal{L}_{SSPCR} = \alpha\mathcal{L}_{shape} + \beta\mathcal{L}_{reconstruction} + \gamma\mathcal{L}_{clustering} \tag{6.6}$$

Where α, β and γ are the weights assigned to each loss. To be specific, $\mathcal{L}_{shape}$ encourages the encoder to produce embeddings encoded with most shape information of the input data, $\mathcal{L}_{reconstruction}$ forces the embeddings to preserve the most intrinsic properties of the input

and makes them reconstructable, $\mathcal{L}_{clustering}$ encourages the embeddings to be distinguishable for clustering. It's worth mentioning that in the equation of $\mathcal{L}_{clustering}$, the cluster indicator matrix F can be computed only when the data matrix X is fixed, however, in SSPCR, X is the embedding matrix that are learned. To make the reformulation applicable, SSPCR adpots Expectation Maximization (EM) approach to update the weights. Here the E step is fixing the F and updating the X, and the M step is fixing the X and updating the F, at the begining, F is randomly initialized as an orthogonal matrix. Note that the updating of F does not require backpropagation, it can be directly computed by SVD decomposition. In addition, F should not be update at each training epoch, since that will make the training unstable. In terms of the shape loss, SSPCR also adpot the mean squared error to represent it.

Currently, the statistical features selected are the same with those used in [45]: the mean value, standard deviation, skewness and kurtosis, their equations can be found in Section 5.1.1. Let E denotes the embeddings, the shape loss computed with statistical features is defined as:

$$\begin{aligned}\mathcal{L}_{shape\_statistical}(X,E) = \frac{1}{2|X|}\sum_{i=1}^{|X|} &\|\mu(x_i)-\mu(e_i)\|^2 + \|+\sigma(x_i)-\sigma(e_i)\|^2 \\ &+ \|SKEW(x_i)-SKEW(e_i)\|^2 \\ &+ \|KURT(x_i)-KURT(e_i)\|^2\end{aligned} \tag{6.7}$$

The choice of statistical shape features are nontrivial and need human participation. The feature set selected here is preliminary, more features will be tested in the future work. To exclude the huamn participation, make the algorithm have less hyperparameters, another shape loss is proposed. This shape loss is inspired by the shape-preserved representations – one can control the dimension of the input data by compression algorithms. This gives a hint that **the representation can be forced to approximate the compressed vector**. Let $\hat{X}$ denotes the compressed sequences, the compression-based shape loss is similar to the

reconstruction loss, defined as:

$$\mathcal{L}_{shape\_compression}(\hat{X}, E) = \frac{1}{2|\hat{X}|} \sum_{i=1}^{|X|} \|\hat{x}_i - e_i\|^2 \tag{6.8}$$

Downsampling is selected as the compression method since it produces the best results in the experiment. According to experimental results, the second shape loss produces slightly better partitions than the first one, but it has an obvious limitation that the dimension of the representation can not be greater than the original sequence.

Given the raw dataset D, the expected number of clusters K, the maximal training epochs, and the updating interval of F T, the pseudocode of the SSPCR learning paradigm can be seen in Algorithm 1.

The core part of SSPCR is the objective function, and there is no special requirement of the structure of the encoder and decoder. **For better comparison, the encoder of Time-series Transformer [67] is chosen as the encoder of SSPCR in this project, while the decoder is a simple fully-connected layer**. The general model structure of Time-series Transformer is similar with the original Transformer (see Figure 6.3), expect the decoder is replaced by other customized neural networks. More detailed description of the original Transformer model can be found in [59].

Given a pre-processed data sample X of length w and m variables (m could be 1), Time-series Transformer first projects the data onto a d-dimensional feature space, where d is a pre-defined dimension of the sequence representation in the model. Usually, such projection is done by a fully-connected layer. Since the model is a simple feed-forward neural network, it can overlook the ordering of input, which is quite important for time-series data such as stock sequences. To preserve the ordering information, a positional encoding PE is added to the projected data U before being fed into the encoder. Note that the positional encodings used in the Time-series Transformer is not the deterministic, sinusoidal encodings proposed

Algorithm 1: SSPCR training method

Input: D: Dataset $D = \{d_1, d_2 \dots d_n\}$
K: number of clusters
$Epoch$: the number of maximum iterations
T: the time interval of updating the cluster indicator matrix F
Output: E: the learned embeddings of D

Pre-process D
Initialise an arbitrary orthogonal matrix F
for $i = 1$ *to* $Epoch$ **do**
  for $X \in D_{batches}$ **do**
    Compute $E = encoder(X)$
    Compute $\mathcal{L}_{shape}$ based on Equation 6.7 or 6.8
    Compute $\mathcal{L}_{clustering}(E)$ based on Equation 6.3
    Compute $X' = decoder(E)$
    Compute $\mathcal{L}_{reconstruction}(X, X')$ based on Equation 6.5
    Compute the total loss $\mathcal{L}_{SSPCR}$
    Update parameters θ of the neural network and the embeddings E by back propagation
    if $i\%T = 0$ **then**
      SVD decomposite E
      Update F by composing the first K singular vectors
    end
    if $early\_stop(\mathcal{L}_{SSPCR})$ **then**
      Stop iteration
    end
  end
end

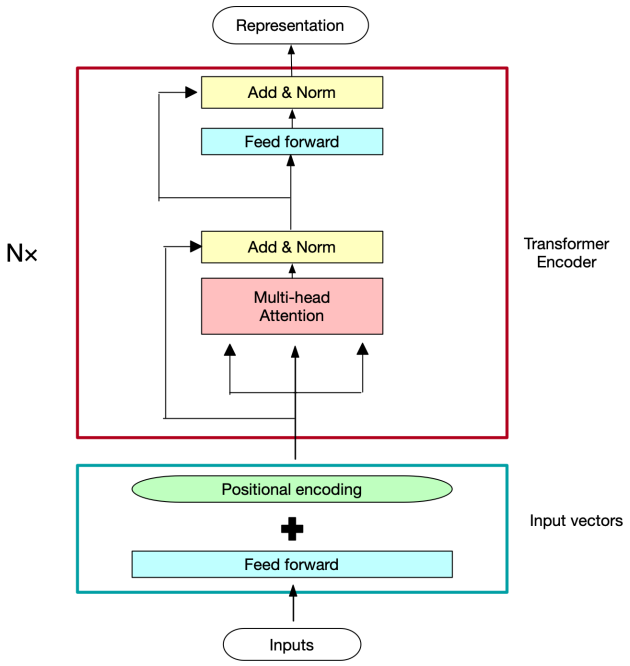
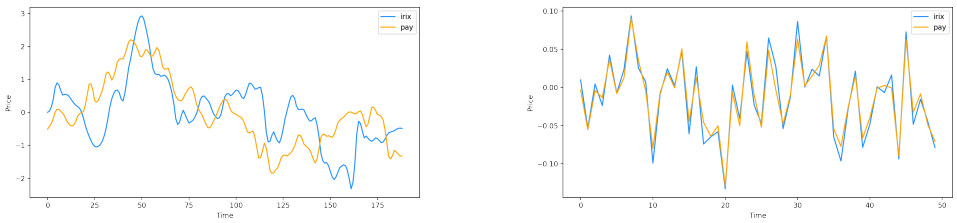


Figure 6.3: The structure of Transformer's encoder

in the original Transformer, instead, they are learnable weights. The encoder is formed by N self-attention blocks, where each block takes the output from previous layer, transforms them into the query, key and value pairs. Then those pairs are used to compute the attention scores of the input. Residual and normalization layers are added to avoid vanishing gradients problem, guaranteeing the success of training. One different setting of the encoder is that it uses batch normalization rather than layer normalization, since the time-series sequence could have outliers that can rarely be found in natural language data. To handle with the problem that time-series sequences may have considerable variation in length, shorter sequences are padded with arbitrary values while longer sequences are truncated. To force the model to ignore the padded positions, an additional padding mask with large negative values on the padded positions are added to the attention scores before feeding them to the softmax layer of self-attention blocks. Figure 6.4 shows an example of applying SSPCR with Time-series Transformer to stock records of "IRIX" and "Pay" in 2011.



(a) Original sequences (b) SSPCR transformation

Figure 6.4: Example of SSPCR representation

6.3 Chapter Summary

This chapter introduces a novel representation learning algorithm designed for shape/trend-based time-series clustering tasks named as Semi-Shape-Preserved Clustering Representation (SSPCR), its pseudocode and the structure of the autoencoder used in the project. SSPCR are designed to take the advantages of both Shape-Preserved and Shape-Changed representations and discard their disadvantages. Semi-Shape-Preserved representations are expected to contain the most useful shape information while can unveil the intrinsic properties of the original data. Since the new vectors usually have different length, one can not force the representations to directly approximate the raw data. To solve this problem, two approaches are proposed: (1) using statistical features to represent the shape information, which doesn't refer to the length of sequences; (2) compressing the raw data to let them approximatable. These two approaches lead to two different shape loss functions. To make the representation suitable for clustering tasks, the objective function of traditional clustering algorithms within-cluster-scatter is added to the loss function. Due to the reformulation of within-cluster-scatter, its computation can be done by simple matrix production, and hence it can be optimized by gradient descent. The autoencoder used to test the effectiveness of SSPCR is the Time-series Transformer, but other autoencoders are also applicable.

Chapter 7

Experiment and Evaluation

7.1 Experimental Settings

This section describes the general experimental settings applied to all the following experiments, for different experiments, their additional settings will be described in corresponding sections. For the fairness, all the evaluation are eventually conducted on the pre-processed "raw" data without any futher transformation. This decision is made because of the facts: (1) different represent methods generate varying number of features, and the number of features will affect the computation of indexes listed in Section 7.1.1; (2) the separation quality could be good on the transformed data, but may not also acceptable on the original data, in term of the trend analysis, the latter one is more important.

7.1.1 Numerical Metrics

Generally, there are two indexes to evaluate the quality of data groups. The first one is the external index, it requires additional ground truth information to compare the generated groups with a reference model. Our dataset dose not contain such information, and hence the metric used here is the **internal index**. Literally, internal index reveals the intrinsic attributes of the groups, and does not require reference model. The common internal indexes are:

1. Silhouette Coefficient (SC) [51]: for each data point, its silhouette coefficient is composed of two scores:

    - **a:** the average distance between a data point and all other data points in the same cluster

- **b:** the average distance between a data point and all other data points in the next nearest cluster

The Silhouette Coefficient s for a single data point is defined as follows, where s ranges from -1 to +1:

$$s = \frac{b - a}{\max(a, b)} \tag{7.1}$$

The final Silhouette Coefficient is defined as the mean of the Silhouette Coefficient of each data point. **Higher Silhouette Coefficient Index means better clustering results**.

2. Calinski-Harabasz (CH) Index [12]: given a data set E of size n_E, and the number of grouped clusters k, the Calinski-Harabasz score s is defined as follows:

$$s = \frac{tr(B_k)}{tr_(W_k)} \times \frac{n_E - k}{k - 1} \tag{7.2}$$

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T \tag{7.3}$$

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E) c_q - c_E)^T \tag{7.4}$$

Where $tr(B_k)$ and $tr(W_k)$ are trace of the between group dispersion matrix and within-cluster dispersion matrix respectively, C_q represents the points in cluster q, c_q represents the centre of q, c_E represents the centre of E and n_q represents the number of points in q. **Higher Calinski-Harabasz Index means better clustering results**.

3. Davies-Bouldin (DB) Index [19]: given two cluster C_i and C_j generated from the same data set without overlapping, their similarity R_{ij} is defined as follows:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{7.5}$$

Where s_i is the average distance between each point in cluster i and the centroid of i,

d_{ij} is the distance between centroids of clusters i and j. The Davies-Bouldin index is defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k}\max_{i\neq j} R_{ij} \tag{7.6}$$

k is the number of clusters. **Lower Davies-Bouldin index means better clustering results**.

In the context of stock grouping analysis, analysts usually expect compact groups, especially when there is no explicit class information (the labels), because compact groups can reveal more general trends. In this aspect, these three indexes is relatively informative. Note that since they focus on different aspects, their trends may not comply with each other, but in general, if a grouping result gets high mark on all the three indexes, it can be regarded as a good separation.

7.1.2 Measures to counteract the effect of randomness

One well-known fact about clustering algorithms is that their outputs.

7.1.3 The choice of group centers

The generated groups could be mess, and it is necessary to highlight their centers when analyse their trends. One simplest way to extract the representative in a group is computing the arithmetic mean of that group. This method is called Euclidean Barycenter, it minimize the summed euclidean distance of that group. However, this method may be inappropriate in this project, since this project will visualize the results with the original sequences, and the original data are not aligned. This is illustrated in Figure 7.1, where the red lines represent the computed centers and black lines represent the data points in the group. To better reveal the general trend of a group, the Dynamic Time Warping (DTW) based methods are used to compute the barycenter. As mentioned before, DTW is a distance measure that can be used to compare two un-aligned sequences. Since it uses the dynamic programming, its time complexity is $O(mn)$, where m and n are the length of that two sequences respectively. To

reduce the time complexity, its variation "Soft-DTW" are used to compute the center. The computation details of Soft-DTW can be found in [53]. The second row of the figure shows the center computed by Soft-DTW with γ set to 1, compared with the Euclidean barycenter, this center is visually more representative.

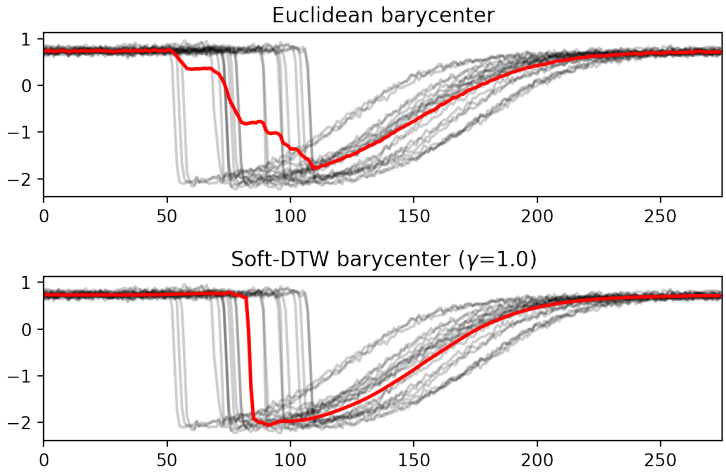


Figure 7.1: Barycenter

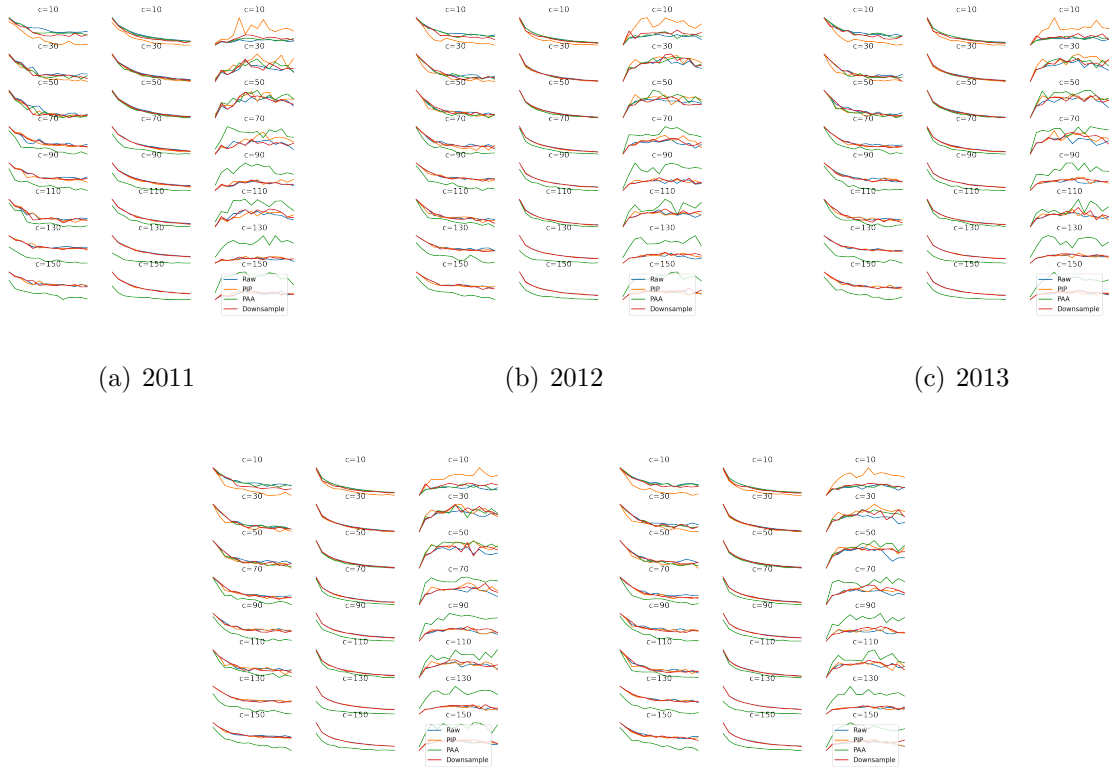
7.2 Comparison of PIP, PAA and Down Sampling

In Section 5.3, different similarity measures with visually similar stocks are used to evaluate the perfomance of three different sequence compression methods, and find that these three methods show no significant difference in terms of such measurement. However, as mentioned before, some compression methods can preserve key points while others can not, we believe this functionality does affect the final perfomance of some applications. Therefore, this section is added to evaluate and verify the effectiveness of these three compression algorithms in the context of time series clustering task. The test data used here are the records of all stocks in our dataset. The number of clusters is limited up to 15 to tradeoff of the ease of visualization and the fairness of comparison.

For all these three methods, one parameter required is the compression ratio or the sequence length c. As mentioned before, there is no standard criterion to numerically evaluate

the effect of compression ratio. The choice of sequence length c depends highly on the given task, and it mostly done by trail and error. Here the most appropriate c is chosen by using KMeans algorithms with the three indexes listed above. Figure 7.2 shows the experimental results. In all sub-figures, the three columns represent the SC score CH score and DB score respectively, while the blue, orange, green, red lines represent the scores of original version , PIP-compressed version, PAA-compressed version and down sampled version respectively. Again, higher SC and CH indicate better separation result, while higher DB means bad result. All 5 years data are used in the experiments, and the results are quite similar: **(1) using data compression algorithms does not improve the perfomance of KMeans algorithm with respect to the indexes used here; (2) Down Sampling gets the highest average score in all experiments with respect to the indexes used here;** (3) when c is small (less than 50), PIP approache gets the worst perfomance with respect to the indexes used here; (4) when c is large, PAA approache gets the worst perfomance with respect to the indexes used here; (5) within the given range, change c does not significantly affect the final scores of Down Sampling; (6) the best "c" locates in range [50,70] for all approaches. To exclude the effect of algorithm itself, other clustering algorithms such agglomerative clustering algorithm are used, and get the similar observation. Observation (1) and (2) are against our assumptions, experimental results prove that down sampling indeed preserve the most shape information of the original sequence. Another interesting observations is that, in all 5 years, the best number of clusters is 2, and the overall score decreases when the number of cluster increases.

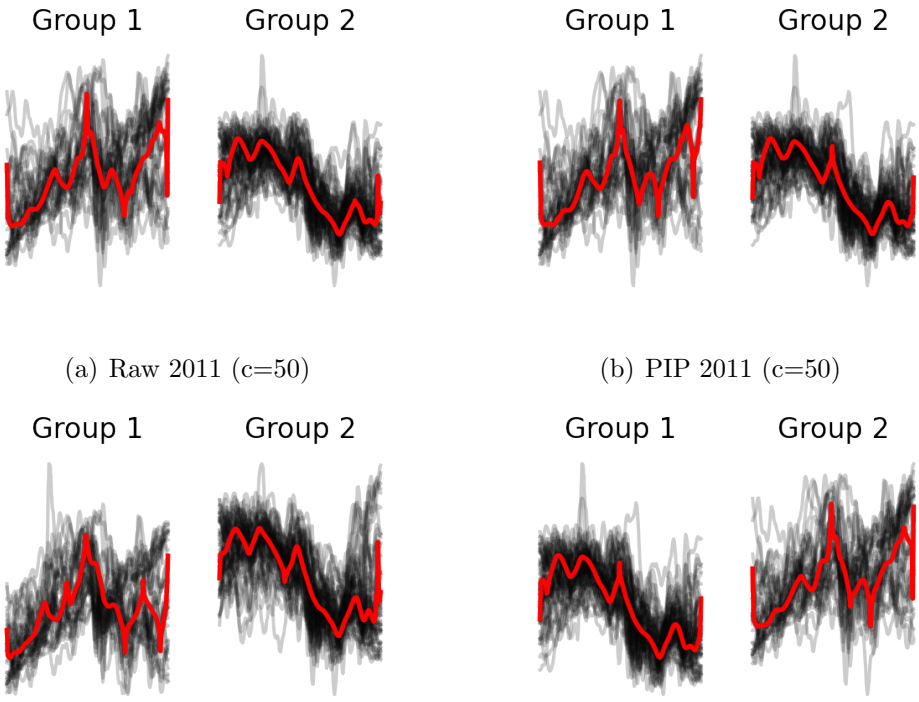
From those observations, it can be conclude that there is no improvement of using compression in terms of these three indexes. However, take the execution time and the storage cost into consideration, the compressed version is at least linearly faster than original version while use less memory, which emphasizes the importance of compression in time-series clustering tasks.



(a) 2011 (b) 2012 (c) 2013

(d) 2014 (e) 2015

Figure 7.2: Different sequence length



(a) Raw 2011 (c=50)

(b) PIP 2011 (c=50)

(c) PAA 2011 (c=50)

(d) Downsampling 2011 (c=50)

Figure 7.3: Visual comparison

Bibliography

[1] Alexander, C. Market models. *A Guide to Financial Data Analysis 1* (2001).

[2] Altunkaynak, A., and Ozger, M. Comparison of discrete and continuous wavelet–multilayer perceptron methods for daily precipitation prediction. *Journal of Hydrologic Engineering 21*, 7 (2016), 04016014.

[3] Åström, K. J. On the choice of sampling rates in parametric identification of time series. *Information Sciences 1*, 3 (1969), 273–278.

[4] Baldi, P. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning* (2012), JMLR Workshop and Conference Proceedings, pp. 37–49.

[5] Barbará, D., and Chen, P. Using the fractal dimension to cluster datasets. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (2000), pp. 260–264.

[6] Basalto, N., Bellotti, R., De Carlo, F., Facchi, P., and Pascazio, S. Clustering stock market companies via chaotic map synchronization. *Physica A: Statistical Mechanics and its Applications 345*, 1-2 (2005), 196–206.

[7] Basu, J. K., Bhattacharyya, D., and Kim, T.-h. Use of artificial neural network in pattern recognition. *International journal of software engineering and its applications 4*, 2 (2010).

[8] Batista, G. E., Keogh, E. J., Tataw, O. M., and De Souza, V. M. Cid: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery 28*, 3 (2014), 634–669.

[9] Bezdek, J. C., Ehrlich, R., and Full, W. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences 10*, 2-3 (1984), 191–203.

[10] Bianchi, F. M., Livi, L., Mikalsen, K. Ø., Kampffmeyer, M., and Jenssen, R. Learning representations of multivariate time series with missing data. *Pattern Recognition 96* (2019), 106973.

[11] Boedihardjo, H., Geng, X., Lyons, T., and Yang, D. The signature of a rough path: uniqueness. *Advances in Mathematics 293* (2016), 720–737.

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

[13] Chakraborty, B. Feature selection and classification techniques for multivariate time series. In *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)* (2007), Ieee, pp. 42–42.

[14] Chen, K.-T. Integration of paths–a faithful representation of paths by noncommutative formal power series. *Transactions of the American Mathematical Society 89*, 2 (1958), 395–407.

[15] Chevyrev, I., and Kormilitzin, A. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788* (2016).

[16] Chung, F.-L., Fu, T.-C., Luk, R., Ng, V., et al. Flexible time series pattern matching based on perceptually important points.

[17] Comaniciu, D., and Meer, P. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence 24*, 5 (2002), 603–619.

[18] Dave, R. N., and Bhaswan, K. Adaptive fuzzy c-shells clustering and detection of ellipses. *IEEE Transactions on Neural Networks 3*, 5 (1992), 643–662.

[19] Davies, D. L., and Bouldin, D. W. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, 2 (1979), 224–227.

[20] Davis, R. A. Introduction to statistical analysis of time series, 2014.

[21] Dempster, A., Petitjean, F., and Webb, G. I. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery 34*, 5 (2020), 1454–1495.

[22] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.

[23] Fawaz, H. I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., and Petitjean, F. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery 34*, 6 (2020), 1936–1962.

[24] Fisher, D. H. Knowledge acquisition via incremental conceptual clustering. *Machine learning 2*, 2 (1987), 139–172.

[25] Fu, T.-c. A review on time series data mining. *Engineering Applications of Artificial Intelligence 24*, 1 (2011), 164–181.

[26] Fu, T.-c., Chung, F.-l., Ng, V., and Luk, R. Pattern discovery from stock time series using self-organizing maps. In *Workshop Notes of KDD2001 Workshop on Temporal Data Mining* (2001), vol. 1, Citeseer.

[27] Han, J., Dong, G., and Yin, Y. Efficient mining of partial periodic patterns in time series database. In *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)* (1999), IEEE, pp. 106–115.

[28] Hu, Y., Feng, B., Zhang, X., Ngai, E., and Liu, M. Stock trading rule discovery with an evolutionary trend following model. *Expert Systems with Applications 42*, 1 (2015), 212–222.

[29] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems 3*, 3 (2001), 263–286.

[30] Keogh, E. J., and Pazzani, M. J. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (2000), pp. 285–289.

[31] Kohonen, T. The self-organizing map. *Proceedings of the IEEE 78*, 9 (1990), 1464–1480.

[32] Kumar, N. B., and Mohapatra, S. *The use of technical and fundamental analysis in the stock market in emerging and developed economies*. Emerald Group Publishing, 2015.

[33] Li, H., Shen, Y., and Zhu, Y. Stock price prediction using attention-based multi-input lstm. In *Asian Conference on Machine Learning* (2018), PMLR, pp. 454–469.

[34] Liao, S.-H., Ho, H.-h., and Lin, H.-w. Mining stock category association and cluster on taiwan stock market. *Expert Systems with Applications 35*, 1-2 (2008), 19–29.

[35] Lin, J., Keogh, E., Wei, L., and Lonardi, S. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery 15*, 2 (2007), 107–144.

[36] Lin, J., Khade, R., and Li, Y. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems 39*, 2 (2012), 287–315.

[37] Lyons, T. Rough paths, signatures and the modelling of functions on streams. *arXiv preprint arXiv:1405.4537* (2014).

[38] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA, pp. 281–297.

[39] Mahesh, B. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet] 9* (2020), 381–386.

[40] Malhotra, P., TV, V., Vig, L., Agarwal, P., and Shroff, G. Timenet: Pre-trained deep recurrent neural network for time series classification. *arXiv preprint arXiv:1706.08838* (2017).

[41] Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT press, 2018.

[42] Nair, B. B., Kumar, P. S., Sakthivel, N., and Vipin, U. Clustering stock price time series data to generate stock trading recommendations: An empirical study. *Expert Systems with Applications 70* (2017), 20–36.

[43] Nakano, K., and Chakraborty, B. Effect of data representation for time series classification—a comparative study and a new proposal. *Machine Learning and Knowledge Extraction 1*, 4 (2019), 1100–1120.

[44] Nanda, S., Mahanty, B., and Tiwari, M. Clustering indian stock market data for portfolio management. *Expert Systems with Applications 37*, 12 (2010), 8793–8798.

[45] Nanopoulos, A., Alcock, R., and Manolopoulos, Y. Feature-based classification of time-series data. *International Journal of Computer Research 10*, 3 (2001), 49–61.

[46] Paparrizos, J., and Gravano, L. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (2015), pp. 1855–1870.

[47] Park, H.-S., and Jun, C.-H. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications 36*, 2 (2009), 3336–3341.

[48] Pinto, N., Doukhan, D., DiCarlo, J. J., and Cox, D. D. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput Biol 5*, 11 (2009), e1000579.

[49] Rasmussen, C. E., et al. The infinite gaussian mixture model. In *NIPS* (1999), vol. 12, pp. 554–560.

[50] Rojas, R. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[51] Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics 20* (1987), 53–65.

[52] Saxe, A. M., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., and Ng, A. Y. On random weights and unsupervised feature learning. In *Icml* (2011).

[53] Schultz, D., and Jain, B. Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces. *Pattern Recognition 74* (2018), 340–358.

[54] Sharan, R., and Shamir, R. Click: a clustering algorithm with applications to gene expression analysis. In *Proc Int Conf Intell Syst Mol Biol* (2000), vol. 8, p. 16.

[55] Shensa, M. J., et al. The discrete wavelet transform: wedding the a trous and mallat algorithms. *IEEE Transactions on signal processing 40*, 10 (1992), 2464–2482.

[56] Singh, A., Thakur, N., and Sharma, A. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (2016), Ieee, pp. 1310–1315.

[57] Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[58] Tsay, R. S. *Analysis of financial time series*, vol. 543. John wiley & sons, 2005.

[59] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008.

[60] Wang, W., Yang, J., Muntz, R., et al. Sting: A statistical information grid approach to spatial data mining. In *VLDB* (1997), vol. 97, pp. 186–195.

[61] Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery 26*, 2 (2013), 275–309.

[62] Wei, J., and Huang, J. An exotic long-term pattern in stock price dynamics. *PLoS One 7*, 12 (2012), e51666.

[63] Wu, D., Wang, X., and Wu, S. A hybrid method based on extreme learning machine and wavelet transform denoising for stock prediction. *Entropy 23*, 4 (2021), 440.

[64] Ye, L., and Keogh, E. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), pp. 947–956.

[65] Yi, B.-K., and Faloutsos, C. Fast time sequence indexing for arbitrary lp norms.

[66] Zaib, G., Ahmed, U., and Ali, A. Pattern recognition through perceptually important points in financial time series. *WIT Transactions on Modelling and Simulation 38* (2004).

[67] Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., and Eickhoff, C. A transformer-based framework for multivariate time series representation learning. *arXiv preprint arXiv:2010.02803* (2020).

[68] Zha, H., He, X., Ding, C., Gu, M., and Simon, H. D. Spectral relaxation for k-means clustering. In *Advances in neural information processing systems* (2001), pp. 1057–1064.

[69] Zhang, L., Aggarwal, C., and Qi, G.-J. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (2017), pp. 2141–2149.

[70] Zhang, T., Ramakrishnan, R., and Livny, M. Birch: an efficient data clustering method for very large databases. *ACM sigmod record 25*, 2 (1996), 103–114.

[71] Zhang, X., Liu, J., Du, Y., and Lv, T. A novel clustering method on time series data. *Expert Systems with Applications 38*, 9 (2011), 11891–11900.

[72] Zhu, X., and Goldberg, A. B. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning 3*, 1 (2009), 1–130.