A Study on Knowledge Discovery among Multiple Evolving Data Streams

Wei FAN

February 2011

ABSTRACT

Recently, data stream mining techniques become more and more important in many applications. For example, intrusion detection on the network flow data, outlier detection of sensor network data, and usage analysis on telecommunication data. In these applications, we utilize data stream mining algorithms to discover up-to-date patterns or associations hidden inside the continuous data. Beside the researches on single data source, in the era of information overload, it is also meaningful to mine correlations among cross-domain data sources in order to support people's decision making. For example, automatic analysis of news articles concerning to the financial market is helpful to generate profitable action signals (buy or sell stocks) accurately.

In this dissertation, we aim to discover interesting correlations among multiple evolving data streams. In terms of different streaming data sources, firstly, we categorize the correlations in the streaming data into two basic correlations: *discrete correlation* and *continuous correlation*. The discrete correlation corresponds to the applications assuming that the data samples are independent with each other. For example, in the market basket data, we assume that the records of customers' purchase are independent, thus the correlations among attributes (products) are discrete. Existing techniques of frequent itemset mining regard the frequently co-occurred sets of attributes as highly correlated attributes. However, in some cases, we may miss important patterns. For example, we may be interested in the knowledge of "what are the symptoms of the new and rare illness?" from medical records data, although the occurrence of the illness is rare. Correlation based association rules mining provides good solution to this kind of problem. However, corresponding algorithms of mining correlated patterns in static datasets exist, but no work has been done so as to complete the same task for data streams.

On the other hand, in some applications (i.e., sensor network, stock market), data samples in the whole collection of time series are correlated with each other in the alignment of time.

In this case, we define the cross-relationship among the multiple continuous time-series data streams as continuous correlation. Existing researches calculated the correlations between streams, and reported highly correlated pairs of streams. However, none of these algorithms manages to compactly and adaptively describe the key trends among the whole collection of streaming time series, although streams often are inherent correlated. The key trends are useful to reduce the massive numerical streams into just a handful of variables. Besides, several works are reported on applying clustering techniques to multiple data streams for discovering cross-relationships. However, the existence of data evolution in data streams leads to another important issue of supporting various clustering requirements at the same time, instead of the existing works on periodical way of checking cluster evolutions.

Consequently, this dissertation extends the study to mine complex correlations in cross-domain data sources by combining the investigations of these two kinds of basic correlations. Hence, in this dissertation, we do further explorations of correlation discovery in different applications of data streams. The key research challenges that arise in this dissertation include:

(I) correlated patterns mining in streaming transaction data;

(II) adaptive and flexible correlation mining among massive continuous time series; and

(III) cross-domain correlation analysis among multiple sources of data streams.

This dissertation makes a number of contributions toward the solutions of these tasks, including the following algorithms:

- *Quantifiable Correlated Patterns Mining*: This method achieves to mine correlated attributes from streaming transaction data. Additionally, in the applications of quantitative data, we also discover frequent ratio associations among the highly correlated attributes. To the best of our knowledge, this work is the first study achieving to mine both of correlations and ratio associations in streaming quantitative transaction data with only single scan of data and limited memory.

- *Correlated-Clusters Mining*: This algorithm reduces massive evolving streaming time-series data into just a handful of *hidden variables*, which summarize the key trends of massive evolving time-series data streams automatically, incrementally and adaptively. We prove that the discovered hidden variables can be used to detect concept drifts immediately, and do efficient forecasting in sensor network.

- *Flexible Timeline Clustering*: A framework is proposed to support various clustering requirements at any time during the whole collection of streaming time-series data. In the requests of clustering, the user specifies arbitrary interested time periods. An incremental time-series approximation method and statistic maintenance hierarchical structures are proposed to satisfy the demands of efficient retrieval with high accuracy.

- *Dynamic Prediction of Stock Prices Based on Analysis of News Articles*: As an example of correlation analysis among cross-domain data sources, we realize automatic analysis of the correlation between online news articles and stock prices. This work classifies the news articles into good news which are followed by a moving up trend in the company's stock market or bad news, reversely. In this problem, classification is the process of mining we defined discrete correlation, for the reason that we treat the collection of news articles as transaction data consisting of words, and the news articles are independent with each other. In order to improve the accuracy of prediction, we also take account of continuous correlations in this problem. On one hand, in the generation of news articles for learning, we abstract trends of stock prices, and then label the news articles according to corresponding trends in stock prices; on the other hand, we propose dynamic mechanism of choosing sliding windows to identify trends of stock prices according to the contents of consecutive news articles, taking account of the case that significant topics in consecutive news articles may influence the stock market sensitively.

Extensive experiments on both synthetic and real-life data demonstrate that our work is effective and practical. Furthermore, as the trial of investigating correlations between news articles and stock prices, the proposed correlation mining techniques can be used as the bases of another intelligent data analysis goal, information integration.

ACKNOWLEDGMENTS

I would like to express my thanks to my adviser, Prof. Toyohide Watanabe, Dr. Engg. for his academic supervision during my Ph.D. study at Nagoya University. I am very fortunate to be his student during the past four years. He has given me courage and confidence to do the research. Without his help, it would have been difficult to achieve my academic goal. More importantly, he gave me many invaluable advices to improve myself.

My sincerely thanks are extended to Prof. Yoshiharu Ishikawa, Dr. Engg., Prof. Shuichiro Yamamoto, Dr. Engg., and Associate Prof. Yusuke Hayashi, Dr. Engg. for serving as my supervisory committee. I greatly appreciate for their feedback, suggestions, and insightful perspective on my research.

I am also grateful to Associate Prof. Koichi Asakura, Dr. Engg. (School of Informatics, Daido University), Assistant Prof. Tomoko Kojiri, Dr. Engg. (Graduate School of Information Science, Nagoya University) for giving me significant advices and guidances on research and life.

Thanks also go to our team members in the Data Mining group and members in Watanabe lab, especially: Mr. Yusuke Koyanagi and Ms. Jie Zhang, for their assistance.

I am very thankful to my parents in China for their love, support and encouragement over these years.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

In recent years, with the advances in processing and communication techniques for facilitating the collection of data continuously, there are many emerging applications to deal with the rapid growth of data sets which are referred to as data streams: for example, network flow analysis [?], sensor network monitoring [?], telecommunication data management [?], financial data analysis [?] and scientific data processing [?]. Traditional systems for data analysis store the arriving data for later analysis, and allow developers to create algorithms that make multiple passes through the data. However, in the applications of data streams, the fact that massive amounts of data arrive at high rates, makes the traditional systems prohibitively slow, and challenges the traditional storage, computation and communication techniques.

Over the past few years, a considerable number of studies have been made on the topic of data streams [?, ?, ?, ?]. We can categorize the literature into two big research directions. Data Stream Management System (DSMS) techniques, which provide SQL-like supports for continuous query have been studied in many data stream systems [?, ?, ?]. On the other hand, data stream mining techniques extended from the traditional data mining techniques to the mining of complex models (e.g., decision trees, sets of rules) in data streams have also provoked a great deal of controversy [?, ?, ?, ?]. Belonging to the field of data stream mining, we aim at contributing to further explorations of correlations that may exist in various applications of data streams.

Our research background, objective and approach features are firstly introduced in this chapter. Then the contributions of this work are briefly summarized. Lastly the organization of this dissertation is depicted.

1.1 Research Background

1.1.1 Mining of Single Data Source

As mentioned in the above section, recently, data stream mining techniques become more and more important in many applications. For example, intrusion detection on the network flow data, monitoring and outlier detection of sensor network data, and usage analysis on telecommunication data. In these applications, we utilize data stream mining algorithms to discover up-to-date patterns or associations hidden inside of the continuous data.

A data stream refers to a continuous flow of data generated at high-speed in dynamic, time-changing environments. The traditional data mining approaches cannot cope with this streaming setting. Hence, it is required to propose new data stream mining techniques for discovering interesting patterns or anomalies in real time. Several crucial challenges have to be taken into account in all applications when we develop data stream mining techniques. Such challenges are:

- **High data generation rate results in one pass requirement.** The volume of data streams is often far beyond our cognition. For example, in a single day *Wal-Mart* records 20 million sales transactions, *Google* handles 150 million searches, and *AT&T* produces 270 million call records. Scientific data collection (e.g., by earth sensing satellites or astronomical observations) routinely produces gigabytes of data per day. In this context, traditional data mining algorithms which make multiple passes over the data set cannot mine even a fraction of the streaming data in useful time. Therefore, it is important to design data stream mining algorithms which require only one pass of the arriving data and work with limited memory.

- **Evolving nature requires dynamic data handing.** The data samples in data streams may show concept shifts over time because of fundamental changes in the underlying phenomena. For effective decision making, stream mining must be adaptive to concept shifts. For example, when customer purchasing patterns change, marketing strategies based on out-dated transaction data must be modified in order to reflect current customer needs. Therefore, a straightforward adaption of one-pass mining algorithms may not be an effective solution to the task. Stream mining algorithms need to be carefully designed with a clear focus on the evolution of the underlying data.

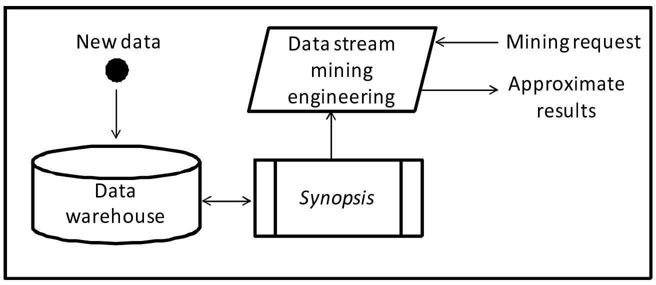


Figure 1.1: Approximate results of data stream mining.

- **Unboundedness entails approximate mining results.** Because data streams are unbounded, they have to be processed while new data arrives. A stream mining algorithm cannot wait until the end of a stream is reached before returning any data mining results. However, obtaining accurate and optimal results requires storing the entire history of the stream. As this requirement cannot be fulfilled in a streaming environment, stream mining algorithms aim to approximate results, as illustrated in Figure **??**. According to the single scan of streaming data, the algorithms store a summary of a data stream, often referred to as *synopsis*, which consumes less space than the stream itself. The tasks of data stream mining are executed on the *synopsis*. Some algorithms provide guaranteed error bounds for the quality of the approximate mining results.

A considerable number of studies have been made on proposing incremental and adaptive algorithms as the extension of traditional data mining algorithms to be applied to data stream environments [**?**, **?**, **?**].

1.1.2 Mining of Cross-domain Data Sources

Besides the needs for knowledge discovery from single data sources, in the era of information overload, it becomes more and more important to mine interesting correlations in streaming cross-domain data sources to support people's decision making.

For example, people read the news to understand what is happening and what might happen in the future, and may be interested in the stories which suggest why current economic performance is poor or predict an upturn in the economy in the coming months specially. Thus news releases influence human behavior, and so may indirectly affect the fluctuations in the time series of

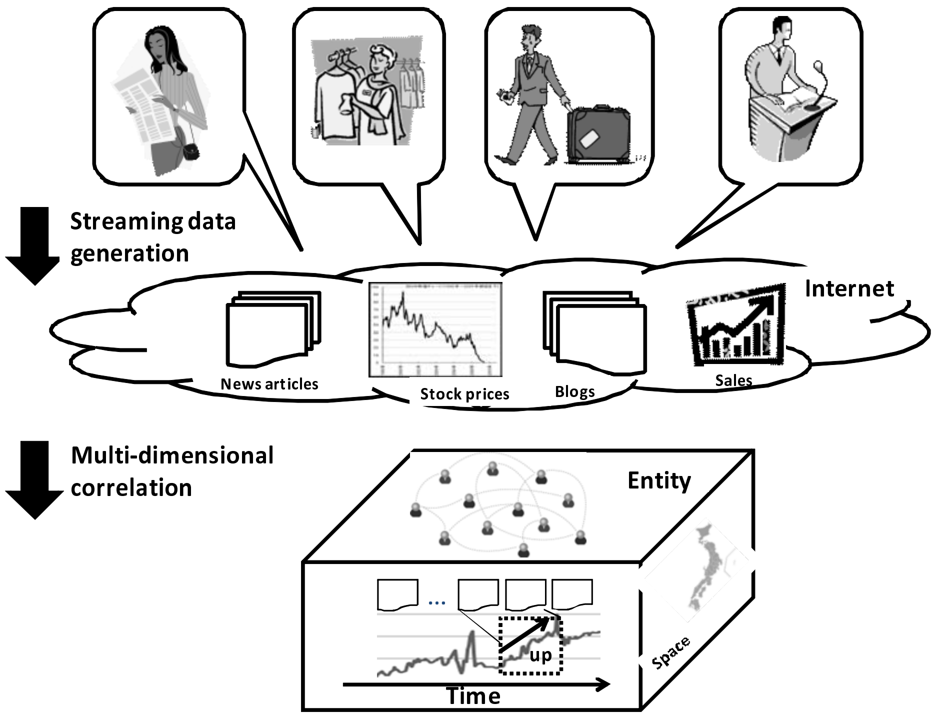


Figure 1.2: Multi-dimensional correlations among multiple streaming cross-domain data sources of social activities.

economic performance. Here we can see the automatic analysis of news articles concerning to the financial market is helpful to generate profitable action signals (buy or sell stocks) accurately.

Moreover, with the rapid proliferation of blogs, wikis and social networking sites, people are expressing their thoughts more freely, publicly, and frequently than ever before, and the demand of analyzing mass opinion by corporations, governments, and individuals is increasing dramatically. By mining user-generated comments, the product manager could quickly "take the pulse" of consumers in regard to certain products, or get the information why the sales dropped. Political parties need public opinion information to determine how their candidates are faring. A person running for president might need to track how she is doing on a state-by-state or even province-by-province level, while compiling similar numbers for her opponents. In the application of e-market, individuals may be prefer to accept the recommendations from people having the similar interests of themselves or belonging to the social network, where people are friends, coworkers, etc.

Therefore, we can see the correlations among social activities are complex and diversified. In order to investigate the complex correlations, the important issue is how to effectively organize the

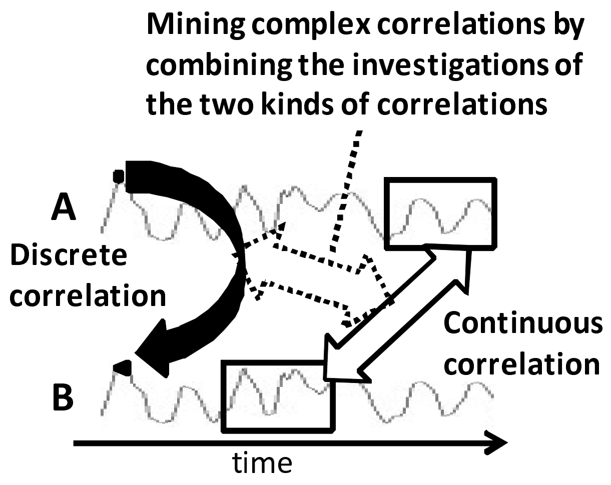

Mining complex correlations by combining the investigations of the two kinds of correlations
A
Discrete correlation
Continuous correlation
B
time

Figure 1.3: Examples of temporal correlations between two streaming variables.

massive streaming data. From the above examples, it would be possible to organize cross-domain data sources in the dimension of time (i.e., investigation of associations between news releases with time series of economic performance), space (i.e., comparison of the status of election state-by-state) as well as entity (i.e., providing recommendations based on the analysis of social network) as shown in Figure ??.

1.2 Research Objective and Contributions

In this dissertation, referring to the Figure ??, we aim at investigating correlations among multiple streaming cross-domain data sources in the dimension of time. It is also possible to combine the mining results of correlations in terms of different dimensions to discover high-level knowledge, but this issue goes beyond the discussion of this dissertation. We are willing to propose efficient methods to integrate multi-dimensional correlations in our further research work. In this dissertation, firstly, we analyze the possible correlations existing in a single data source, and then we extend the study to mine complex correlations in cross-domain data sources.

In terms of data streams in a single data source, we categorize the correlations into two basic correlations *discrete correlation* and *continuous correlation*. As shown in Figure ??, we give the example of two streaming variables A and B. The discrete correlation corresponds to the applications assuming that the two dimensional data samples (one dimension is A, and the other is B) at different time points are independent with each other. For example, in the market basket data, we assume that the records of customers' purchase are independent, thus the correlations among attributes (A and B) are discrete. Existing techniques of frequent itemset mining regard the frequently co-occurred sets of attributes as highly discretely correlated attributes. Taking the example as shown in Table **??**, the network flow data records the *IP* addresses of sources and destinations, duration of sessions, bytes as well as the protocol of transmissions. From these data, frequent itemset mining techniques can discover the discrete correlations, such as sets of *IP* addresses (sources and destinations) involved in more than 1000 sessions. Other techniques like clustering as well as classification are also applied to mining multi-dimensional streaming transaction data assuming data samples are independent with each other.

On the other hand, in some applications, as shown in Figure **??**, variables A and B are continuous evolving time-series data (i.e., the daily close price data of every brand in stock market, observation data from sensor network). In this case, it may be not appropriate to assume the data samples at different time points of variable A or B to be independent. Take the example of a coal mine, where thousands of sensors keep reporting the observed data of the temperature, the humidity, and the concentrations of oxygen and gas in pits. It is important to monitor the surveillance data streams. Due to the nature of sensors and the detection environment, the collected data often includes noise. In such a scenario, apparently, existing studies on clustering the multi-dimensional streaming data objects [**?**] (i.e., independent records of (temperature, humidity, oxygen concentration, gas concentration)) does not make good sense. These methods fail to

Table 1.1: An example of network flow data.

Source	Destination	Duration	Bytes	Protocol
10.1.0.2	16.2.3.7	12	20K	http
18.6.7.1	12.4.0.3	16	24K	http
13.9.4.3	11.6.8.2	15	20K	http
15.2.2.9	17.1.2.1	19	40K	http
12.4.3.8	14.7.8.4	26	58K	http
10.5.1.3	13.0.0.1	27	100K	ftp
11.1.0.6	10.3.4.5	32	300K	ftp
19.7.1.2	16.5.5.8	18	80K	ftp
...	...	...	...	...

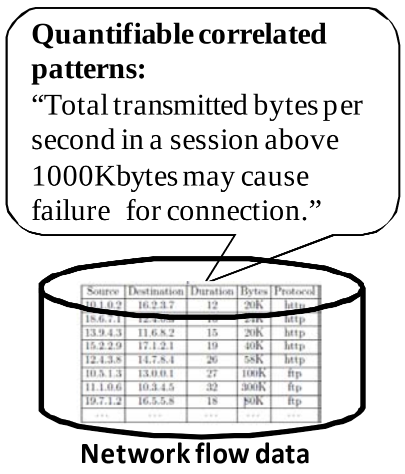
discover interesting knowledge as well as in high-dimensional applications, because all pairs of data objects tend to be almost equidistant from one another due to sparsity of data [?]. Instead it may be helpful to analysis the continuous data distribution of all the sensors. Therefore, in terms of the applications of time-series data streams, we define the continuous correlation to describe the cross-relationship between the variables A and B. According to the continuous correlation, if we know the behavior of variable B in a time period, then we may predict the future fluctuations of variable A.

In terms of the existing researches on frequent itemset mining, hidden discrete correlation among rare but important events (i.e., the knowledge of "what kind of transmission may cause failure in network connection?" in a medical records data) cannot be discovered by existing frequent itemsets mining algorithms. Thus, the definition of frequent co-occurrence in the existing work is insufficient to represent the discrete correlation among attributes. Those infrequent but significant patterns are too expensive to be obtained by the existing frequent itemset mining algorithms. Corresponding algorithms of mining correlated patterns in static datasets exist, but no work has been done so as to complete the same task for streaming transaction data. On the other hand, existing researches in [?, ?, ?] calculate the continuous correlation among time-series data streams, and report highly correlated pairs of streams. However, none of the algorithms can compactly and adaptively describe the key trends among the whole collection of streaming time series. Furthermore, several works are reported on applying clustering techniques to multiple data streams for discovering continuous correlations. However, the existence of data evolution in data streams leads to another important issue of various clustering requirements at the same time, instead of the existing works on periodical way of checking cluster evolutions [?, ?].

Finally, this dissertation extends the study to mine complex correlations in cross-domain data sources by combining the investigations of these two kinds of basic correlations. Therefore, the key research challenges that arise in this dissertation include

(I) Discrete correlated patterns mining in streaming transaction data [?, ?];

(II) Adaptive and flexible continuous correlation mining among massive time-series data streams [?, ?, ?, ?, ?, ?, ?]; and


Quantifiable correlated patterns:
"Total transmitted bytes per second in a session above 1000Kbytes may cause failure for connection."
Network flow data

Source	Destination	Duration	Bytes	Protocol
10.1.0.2	16.2.3.7	12	20K	http
18.6.7.1	[illegible]	[illegible]	24K	http
13.9.4.3	11.6.8.2	15	20K	http
15.2.2.9	17.1.2.1	19	40K	http
12.4.3.8	14.7.8.4	26	58K	http
10.5.1.3	13.0.0.1	27	100K	ftp
11.1.0.6	10.3.4.5	32	300K	ftp
19.7.1.2	16.5.5.8	18	80K	ftp
...	...	...	...	...

Figure 1.4: An example of quantifiable correlated pattern mined from network flow data.

(III) Cross-domain correlation analysis among multiple sources of data streams [?, ?, ?].

In the following subsections, we give summarized explanations of these challenges.

1.2.1 Discrete Correlation Mining in Streaming Quantitative Transaction Data

Frequent itemset mining as one of the most important issues of data mining has been applied to many applications of transaction data. Often, the number of resultant frequent itemsets in terms of specified statistic threshold is very large. Problems associated with excessively many meaningless itemsets have been studied in recent years. Mining correlated patterns is one of the effective solutions to address these problems. Correlated patterns mining is able to filter out the uncorrelated itemsets, and to identify the rarely occurring but completely correlated ones. Corresponding algorithms of mining correlated patterns in static datasets exist, but no work has been done so as to complete the same task for streaming transaction data. Due to the issue of large volume of data in data streams, mining correlated patterns raises a new challenge to solve the combinatorial explosion problem for calculation of correlation coefficients. It is impossible to scan transactions multiple times in terms of time and space constraints in stream environments.

Furthermore, in the applications of quantitative transaction data, we also consider quantitative associations among highly correlated itemsets. We propose algorithms to mine frequent ratio associations among highly correlated itemsets, called *quantifiable correlated patterns*, in a streaming transaction data. This kind of knowledge is more informative and applicable for users' decision-making. For example, as shown in Figure **??**, from the network flow data, we may detect the correlation among the failure of connection, duration of session, and the number of bytes transmitted, and the ratio association between the duration of session and the number of bytes is 1000 Kbytes per second. However, this problem cannot be handled easily. For the reason that ratios are real data; it is necessary to count supports of infinitely different values for mining frequent ratios. In this dissertation, we propose efficient algorithms to resolve these two combinatorial explosion problems with only one scan of transaction data.

1.2.2 Continuous Correlation Mining among Massive Time-series Data Streams

As discussed in the above section, we do further explorations of monitoring and discovering continuous correlations among massive time-series data streams. To this end, in this dissertation, we propose two novel approaches: *correlated-cluster mining* and *flexible timeline clustering*. Here, the former method targets to discover hidden variables for summarizing the whole streams collection taking into account of the data evolution; the later method supports various requirements of clustering multiple time-series data streams at the same time.

1.2.2.1 Correlated-cluster Mining

In some applications (i.e., sensor network data), time-series data streams are often inherent correlated, therefore, we consider to discover hidden variables for compactly and adaptively describing the key trends among the whole collection of streaming time series. This process can be used to reduce the massive numerical streams into just a handful data streams. Therefore, the hidden variables mining is useful for data compression and outlier detection. For example, in Figure **??**, in the application of sensor network, there are time series of two sensors. Each time series consists of three partitions along with the time. We can see that both of the two sensors show periodicity during phase 1 and phase 3, while during the phase 2, the observation data of sensor 1 is "stuck" due to some failure. The extra second hidden variable was discovered during the phase 2 captures the presence of the "abnormal" trend of sensor 1.

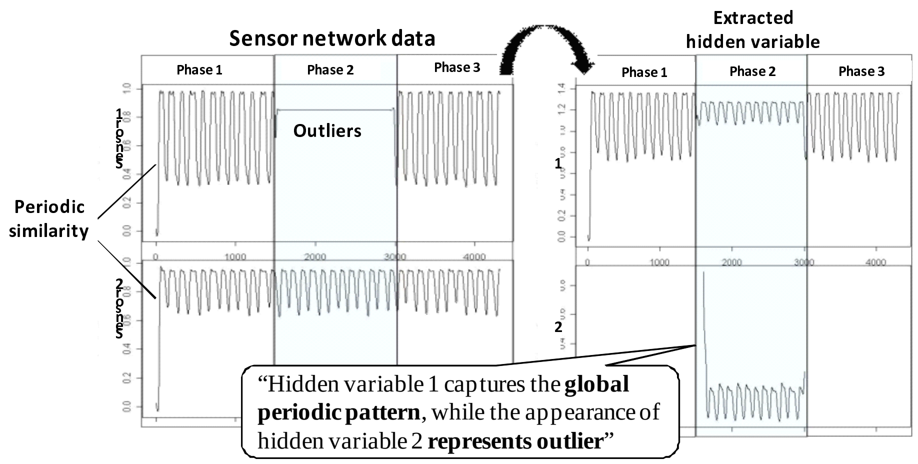


Figure 1.5: An example of mining hidden variable for immediate detection of outlier.

In the case of static data, the Karhunen-Loeve transformation (KLT) or Principal Component Analysis (PCA), is the prominent approach for discovering hidden variables among multiple time series. PCA projects the data set from the original n to a d-dimensional space, where $d \ll n$, and each new dimension is a linear combination of the original dimensions. While, in terms of characteristics and crucial challenges of data streams, the traditional PCA technique is no longer applicable to mine correlations in data stream environment. Firstly, due to the time and space constraints, it is impossible to store all of the unlimited and continuous data, and re-perform the eigen-analysis on all of the data including the newly arrived data sample. Therefore, an incremental process for mining correlations with one single scan of streaming data is required. Secondly, due to the data evolution in data streams, data distribution and underlying correlations may be subject to continuous changes. For example, in the case of a 2-dimensional data set as illustrated in Figure **??**, we can use a single dimension (the first principle component (PC)) to capture the variance of the data. It is possible to represent original data samples by their projection on the single dimension, eliminating other dimensions (in this case the second PC) with little loss of information. However, in practice, continuous data within data streams may change at any time. In such cases, traditional PCA technique is unable to discover new appearing correlation pattern, but sensitive to the effect of previous ones (see Figure **??**). Therefore, it is required to monitor the change of correlation in data streams, and distinguish the emerging correlation with stale ones. Here, we focus on local correlations in each correlated-cluster corresponding to the

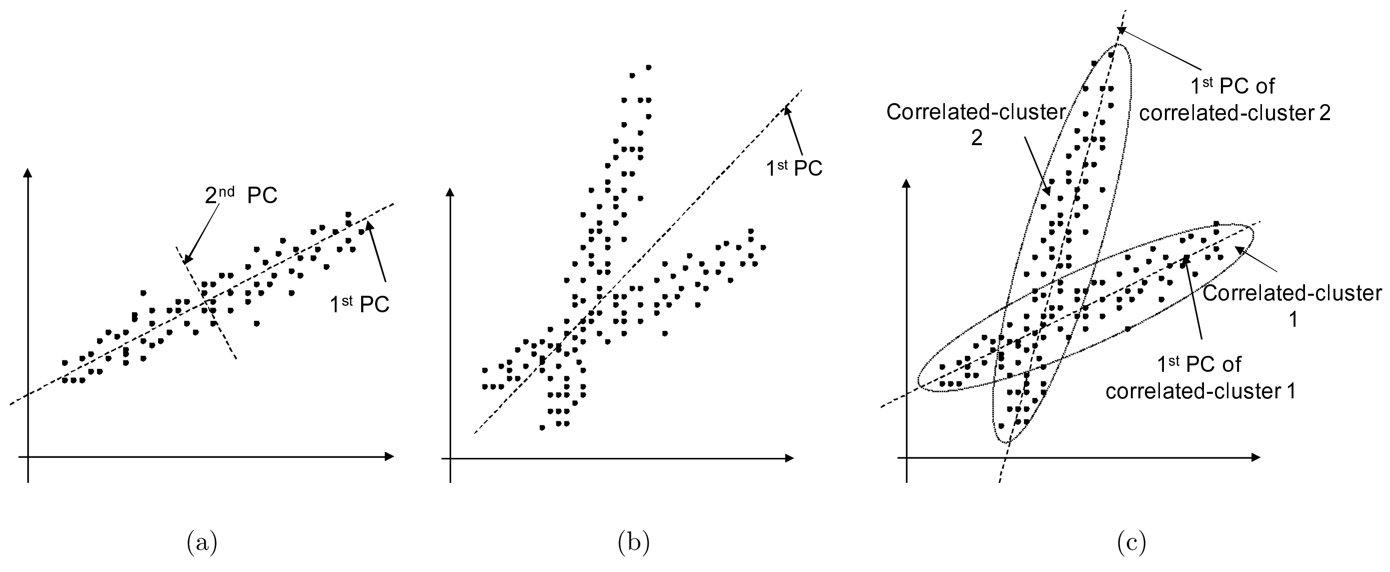


Figure 1.6: Principal Component Analysis and correlated-clusters (a) PCA on currently correlated data reduces 2-dimensional to 1-dimensional (b) PCA on continuous changing correlated data (c) PCA on correlated-clusters in the same data as in (b)

data evolution (*e.g.*, the data in Figure **??**). Obviously, both of the data samples and correlated dimensions are different from one correlated-cluster to another.

Therefore, in this dissertation, we propose an online incremental and scalable method to mine correlated-clusters in multiple evolving data streams. The resultant relevant subspaces (hidden variables) summarize the multiple data streams. Additionally, considering the concept drifts, we guarantee the effectiveness of the proposed algorithms to automate the change detection and maintenance of underlying correlation in the streaming data.

1.2.2.2 Flexible Timeline Clustering

Several existing works are reported on applying clustering techniques to discovering cross-relationship among multiple data streams. Consider the application of stock market data analysis. Assume that the clustering request is unknown when the data is collected and processed. After the time period of clustering request is given, according to clusters provided, investors are able to choose a combination of several groups of stocks to reduce the risk of investment. For example, some users are interested in the short-term behavior of clusters, such as the daily or hourly

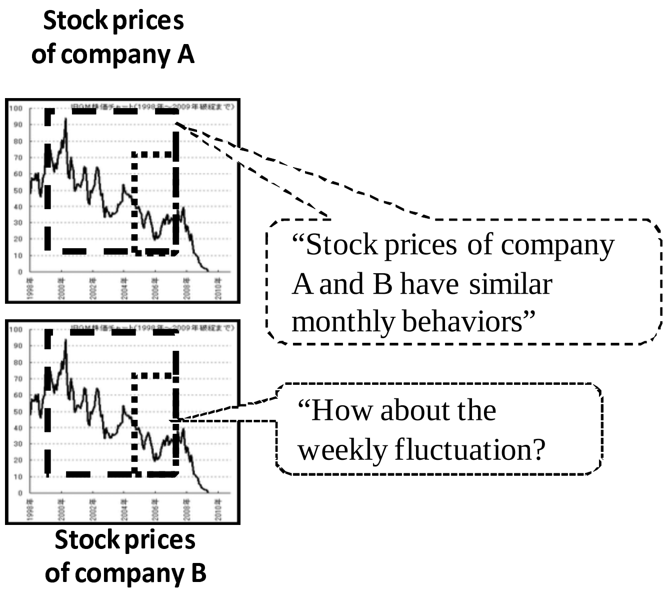
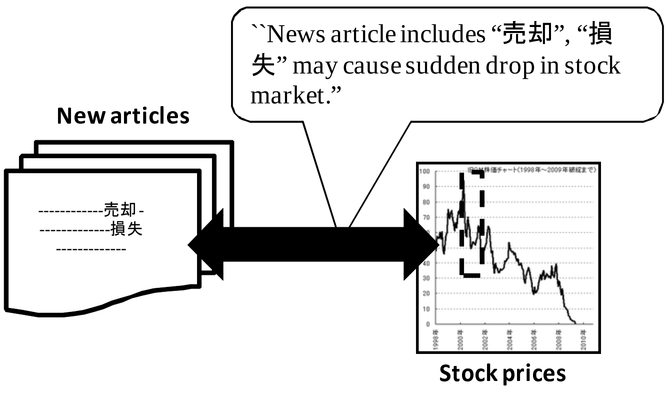

Stock prices of company A
"Stock prices of company A and B have similar monthly behaviors"
"How about the weekly fluctuation?
Stock prices of company B

Figure 1.7: An example of supporting flexible timeline clustering of two time series of stock prices.

behavior. Therefore, we would like to provide daily clusters for one week or hourly cluster for a few hours. On the other hand, for users interested in the long-term behavior of clusters, such as the monthly or annual behavior, we would like to provide monthly clusters for this year or annual clusters for last five years. This leads to an important application of supporting various clustering requirements at the same time. In the users' requests, users specify the interested time periods. To this end, in the context of data streams, how to summarize the huge number of data resources for online calculation of similarity, and how to efficiently retrieve abstractions of the streaming data during a certain period in response to users' requests are important issues. In this dissertation, an online time-series approximation method and statistic maintenance hierarchical structures are proposed to address the computation and space constraints in data stream environments.

For example, in Figure **??**, in the application of stock prices data, we satisfy clustering massive stock series data over arbitrary time period flexibly. For instance, "the monthly stock prices of company A and company B behavior similarly". Meanwhile, we also support requests to observing different time periods, such as the relationship in terms of weekly fluctuation.


``News article includes "売却", "損失" may cause sudden drop in stock market."
New articles
------------売却-
-------------損失

Stock prices

Figure 1.8: An example of bad news article which cause a sudden drop in the stock market.

1.2.3 Cross-domain Correlation Investigation over Multiple Data Sources

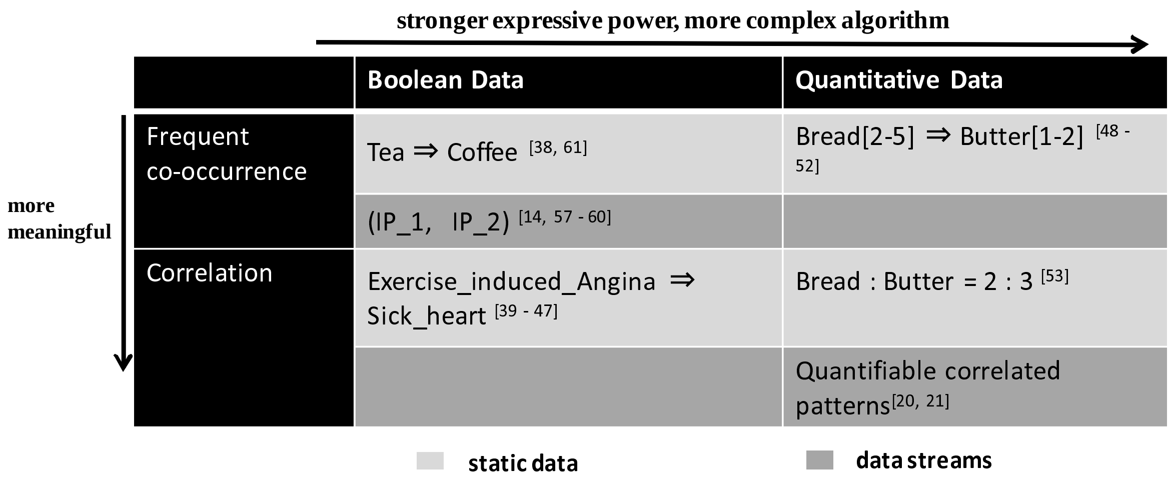
In real applications, data sources from multiple domains may be semantically related with each other, but has different data type or different data distribution. However, this kind of semantic relationship is often the critical knowledge for decision-making or explanation of events. Cross-domain correlation mining studies how to investigate the semantic relationship among the different data sources. In this dissertation, we investigate the association between stock price fluctuation and released news articles as an example. Given a news article, we decide whether it is a piece of good news that is followed by a moving up trend in stock market or a piece of bad news reversely. Additionally, we predict how much the fluctuation of stock price influenced by the news articles is. Additionally, we also forecast how much would be the fluctuation of stock prices influenced by the news article. For example, in Figure **??**, we discover the correlations between news articles and stock prices, like the released news articles which report the information about "売却" and " 損失" may cause a sudden drop in the stock market (i.e., the black dashed part in the stock price data).

1.2.4 Contributions

In this dissertation, we aim to discover knowledge among multiple evolving data streams. We propose efficient algorithms to mine interesting patterns or correlations in the collection of streaming data. Firstly, we address the problem in a single data source. In terms of different applications, we define two kinds of basic correlations existing in the streaming data. One is discrete correlation, and the other is continuous correlation. For example, in the case of market basket data, we analyze continuous records of customers' purchase. This records are independent with each other. Therefore, when we want to discover the patterns of customers' purchase, we should mine the discrete correlations among the attributes in the records. Meanwhile, in terms of the applications handling time-series data streams, data samples are correlated with previous ones on the alignment of time. Thus we define the cross-relationship among time series as the continuous correlation. Secondly, this dissertation extends the study to mine complex correlations in cross-domain data sources by combining the investigations of these two kinds of basic correlations. We take the example of investigating the correlation between online news article and stock prices in order to realize prediction of stock market. Particularly, this dissertation introduces novel and efficient algorithms as follows:

- *Quantifiable Correlated Pattens Mining*: This method achieve to mine discrete correlations from streaming quantitative transaction data and show that this is a more effective approach than mining associations to discover useful patterns. We propose the novel notion of Quantifiable Correlated Pattern (QCP), which is founded on two formal concepts, correlation and frequent ratio association. On one hand, we apply the min-hashing technique to the problem of QCP mining to capture the dependency between the attributes; on the other hand, we further ensure the dependency between the attributes with specific quantitative ratios.

  As shown in Figure **??**, we enumerate the related work on knowledge discovery of static as well as streaming transaction data. The first research work [**?**, **?**] proposed to mine frequent itemsets in static binary data. For example, it discover the association rule like "if a customer buys tea, then he would buy coffee together". However, as pointed out in the studies of correlated patterns mining [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**], the frequent co-occurrence could not describe correlation sufficiently. On the other hand, some studies discovered quantitative associations. Compared with the patterns proposed in [**?**, **?**, **?**, **?**, **?**] in the form of quantitative intervals, the ratio rules in [**?**] were useful to estimate missing data and do prediction. While in the case of data streams, some studies detected approximate frequent patterns in [**?**, **?**, **?**, **?**, **?**]. Unfortunately, there is no study on correlation mining


stronger expressive power, more complex algorithm
more meaningful
░ static data
▓ data streams

	Boolean Data	Quantitative Data
Frequent co-occurrence	Tea ⇒ Coffee [38, 61]	Bread[2-5] ⇒ Butter[1-2] [48 - 52]
	(IP_1, IP_2) [14, 57 - 60]	
Correlation	Exercise_induced_Angina ⇒ Sick_heart [39 - 47]	Bread : Butter = 2 : 3 [53]
		Quantifiable correlated patterns[20, 21]

Figure 1.9: Studies on knowledge discovery in transaction data

or quantitative association mining from streaming transaction data. Our proposed method in [?, ?] firstly achieves to mine both of the correlations and quantitative associations in data stream environment with only single scan of data and limited memory. Due to the issue of large volume of data in data streams, mining correlated patterns raises a new challenge to solve the combinatorial explosion problem for calculation of correlation coefficients. It is impossible to scan transactions multiple times in terms of time and space constraints in stream environments. Therefore, existing methods for correlation mining in static data cannot be extended to data streams environment easily.

In our proposed method we devise an efficient method to address the combinatorial explosion problem using *min-hashing* technique proposed in [?] to find possibly correlated attributes without calculating correlation coefficients. In [?], the min-hashing function was used to identify association rules with high confidence. Though related, confidence for association rules and correlation coefficients are two different quantities. In particular, the fact that the correlation coefficient of a pair is above a certain threshold does not mean that the confidence of the corresponding association rule is above that threshold. We use the min-hashing function in a differnet way. We show a strong connection between the Jaccard distance and the correlation coefficient of the attribute sets. In particular, we observe that, if the attribute sets has a large correlation coefficient, then its Jaccard distance must be small. We use the min-hashing function to estimate the Jaccard distance and only select attribute sets with small distance to the candidate sets. Therefore, we can get the candidate sets of correlated attributes efficiently with single data scan. We remark that our proposed pruning process may produce false negative, i.e., sets with high correlations might be pruned. However, the false-negative probability can be controlled according to the parameter of false-negative tolerance which can be set arbitrarily small.

Furthermore, in order to mine quantitative ratio associations between attributes, for the reason that ratios are real data; it is necessary to count supports of infinitely different values for mining frequent ratios. In order to address this problem, we propose the ideas of *ratio range* and *downward closure property* to reduce the search space for discovering frequent ratio associations between attributes.

- *Correlated-Clusters Mining*: This algorithm reduces massive evolving streaming time-series data into just a handful of *hidden variables*, which summarize the key trends of massive evolving streaming time-series data automatically, incrementally and adaptively. We prove that the discovered patterns can be used to detect concept drifts immediately, and to do efficient forecasting in sensor network.

  Several related works in [?,?,?] realized mining of pair-wise correlations. In order to discover key trends in the whole collection of data streams, as discussed in Section 1.2.2.1, the traditional batch process of Principal Component Analysis is not appropriate in the environment of data streams due to the large volume of streaming data as well as the existence of data evolution. Other related works for incremental Principal Component Analysis elaborated in Section 2.3.2 are sensitive to data evolution. In our propose method, we propose two ideas to address this problem. One is a extended multi-variate regression measure which is used for online detection of data evolution. The other is an incremental process of Principal Component Analysis motivated from the principal of adaptive filter in [?] used for signal processing.

- *Flexible Timeline Clustering*: A framework is proposed to support various clustering requirements at any time during the whole collection of streaming time-series data. In the requests of clustering, the user specifies arbitrary interested period of time. An incremental time-series approximation method and statistic maintenance hierarchical structures are proposed to satisfy efficient retrieval and high accuracy requirements. Results of clustering can be analyzed in different ways according to the specific applications.

Only our proposed method and the method proposed in [?] support flexible mining cross-relations between pairs of stream data. In [?], the authors proposed calculation of pair-wise correlations based on Discrete Fourier Transformation (DFT) statistics of original data. As discussed in [?], Piecewise Linear Approximation (PLA) which approximates series data samples into linear segments is widely used because of its simplicity and is applicable to online approximation. In our work, we propose a segmentation criterion called *tolerant slope interval* for the PLA of time series. Therefore, our proposed method has the advantages of smooth approximation and low computational complexity. Moreover, the proposed method gives more accurate solutions to recent data, but coarser solution for older data, realizing more efficient utilization of space resource.

- *Dynamic Prediction of Stock Prices Based on Analysis of News Articles*: This method realizes automatic and dynamic classification of news articles for predicting the forthcoming trends of stock prices. Given a news article referring to one company, we decide whether it is a piece of good news that is followed by a moving up trend in the company's stock market or a piece of bad news reversely. The novelty of our proposed framework is the achievement of dynamic analysis of the complex correlation between online news articles and stock price series. Existing research work did not support flexible identification of the trends in stock price series, or take account of the case that temporal consecutive news articles may influence the stock market sensitively. In our proposed framework, we combine the investigations of the discrete correlation as well as the continuous correlation.

  In this problem, the process of classification achieves to mine discrete correlation as shown in the blue part in Figure ??, for the reason that we treat the collection of news articles as transaction data consisting of words, and the news articles are independent with each other. While, in order to improve the accuracy of prediction, we also take account of continuous

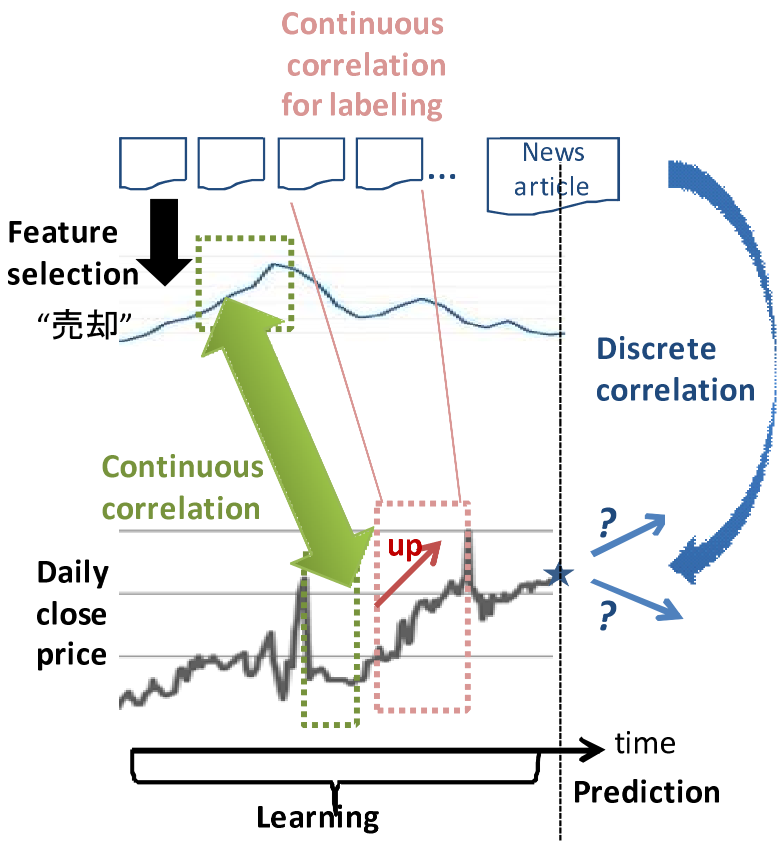


Figure 1.10: Combination of investigations of discrete correlation and continuous correlation in the application of stock price prediction based on analysis of online news articles

correlations in this problem. As shown in Figure ??, on one hand, in the generation of news articles for learning, we abstract trends of stock prices, and then label the news articles according to corresponding trends in stock prices as shown in the red part of Figure ??. On the other hand, taking account of the evolving social environment, we can see that people may be more interested in the contents of consecutive news articles which may influence the stock market sensitively. Therefore, it is prefer to identify the trends in stock market dynamically. We propose dynamic mechanism of choosing sliding windows to identify trends of stock prices according to the contents of consecutive news articles as shown in the green part of this figure. In order to detect the sensitive topic in consecutive news articles, we observe continuous changes of the words' occurrences.

Table 1.2: Research themes.

Theme	Approach	Basic Technology
Discrete correlation mining	Mining quantifiable correlated pattern in streaming transaction data	Mining false-tolerant correlated attributes; Mining frequent quantitative ratio relationships among attributes.
Continuous correlation mining	Mining correlated-clusters among multiple time-series data streams	Multi-variable regression measure for change detection; Incremental update of local hidden variables.
	Flexible timeline clustering of multiple continuous time series	Online summarization of time series; Hierarchical structures for maintenance of summarization statistics.
Correlations mining in cross-domain data sources	Combination of the investigation of discrete correlation and that of continuous correlation	Dynamic identification of trends in stock price series; Classification of news articles.

1.3 Organization

The remaining chapters will give the explanations of our proposed methods in detail. This dissertation consists of three themes discussing discrete correlation mining, continuous correlation mining, as well as the combination of investigations of these two kinds of correlations among multiple evolving data streams, respectively. Table **??** lists the themes, approaches and basic technologies of our work. Figure **??** shows the relationship among these chapters.

The remaining of this dissertation is organized as follows. Chapter 2 overviews the related work and research horizon. Firstly, models of processing data streams used in this dissertation for correlation mining are discussed. Then, we review existing work of frequent itemsets mining in data streams, and then shed the lights on the importance of correlation and new challenges for the task of quantifiable correlated patterns mining in streaming transaction data. In terms of the second theme, targeting to mining continuous correlation among time-series data streams, we focus on the problems of discovering global correlations and cross-relationships based on clustering techniques, respectively. We review the related work for addressing these two tasks. Finally, existing work on stock prediction based on news articles is also introduced.

Chapter 3 concerns on the first theme: mining quantifiable correlated pattern in streaming transaction data. In order to address the key challenges of two combinatorial explosion problems for mining correlated attributes as well as frequent ratio relations among these correlated attributes, we devise an efficient method using *min-hashing* technique to find possible correlated

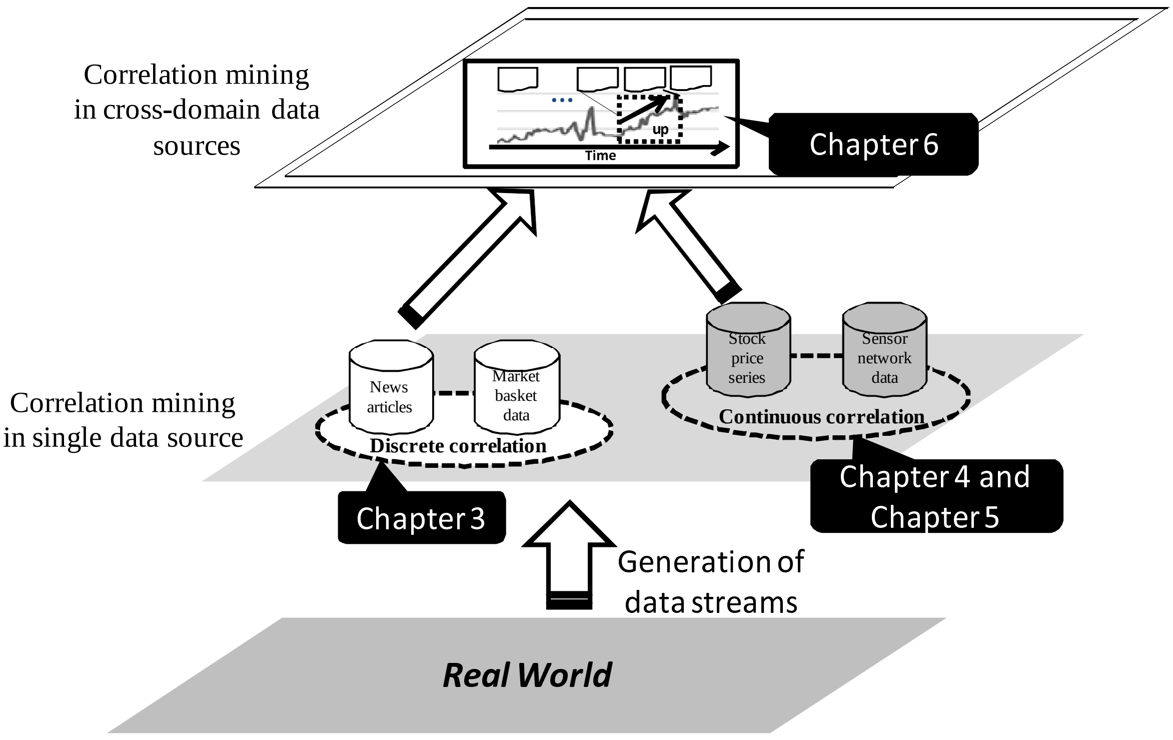


Figure 1.11: Organization of this dissertation

attributes without calculating correlation coefficients, and reduct the infinite search space of ratio patterns based on the ideas of ratio range and downward closure property in order to mine frequent ratio relations.

Chapter 4 and 5 concern on the second theme: mining continuous correlation among multiple time-series data streams. Chapter 4 elaborates our proposed algorithms for incremental discovery of hidden variables to summarize the key trends in continuous time-series. Considering the existence of data evolution, we propose a multi-variable regression measure to cluster data samples. Additionally, in each cluster of data, an incremental method for Principal Component Analysis is proposed to update the local correlation or hidden variables. Meanwhile, clustering techniques also can be used for discovering cross-relationships among time series. Chapter 5 proposes an online time-series approximation method and statistic maintenance hierarchical structures for achieving flexible timeline clustering. The framework consists of two phases: the online maintenance which provides an efficient mechanism to maintain summary hierarchies of data streams; and on the side of the offline clustering phase retrieves approximations of desired subsequences from summary hierarchies according to clustering requests.

In the first and second theme, we discuss the methods for mining discrete correlation as well as continuous correlation in different applications of single data source, respectively. In the third theme, we consider the problem of mining correlation in multiple cross-domain data sources. In Chapter 6, we take the example of investigating the correlation between online news article and stock price series. Given a news article referring to one company, we decide whether it is a piece of good news that is followed by a moving up trend in the company's stock market or a piece of bad news reversely. In this problem, the process of classification achieves to mine discrete correlation, for the reason that we treat the collection of news articles as transaction data consisting of words, and the news articles are independent with each other. While, in order to improve the accuracy of prediction, we also take account of continuous correlations in this problem. On one hand, in the generation of news articles for learning, we abstract trends of stock prices, and then label the news articles according to corresponding trends in stock prices. On the other hand, taking account of the evolving social environment, we can see that people may be more interested in the contents of consecutive news articles which may may influence the stock market sensitively. Therefore, it is prefer to identify the trends in stock market dynamically. We propose dynamic mechanism of choosing sliding windows to identify trends of stock prices according to the contents of consecutive news articles. In order to detect the sensitive topics in consecutive news articles, we observe continuous changes of the words' occurrences.

In Chapter 6, we take the individual company as the research objective. The news articles referring to the company are useful to predict the stock prices of the company itself. In fact, the proposed methods introduced in Chapter 4 is also appliable to the framework proposed in Chapter 6, and the results are useful to explain the financial market involving multiple companies. For example, firstly, we can discover the abonormal behaviors of stocks comparing to the key trends in the whole market according to the method proposed in Chapter 4, and we can explain how it happened by analysis the corresponding news articles.

Comparing the processes of identifying patterns or trends in time series data discussedn in Chapter 5 and Chapter 6, we can see that Chapater 5 provides mechanism to store historical data as well as the flexibility of multi-solution clustering. Nevertheless, in Chapter 6, we foucus on dynamically choosing sliding windows for abstracting trends now and predict future accurately. Thus, we scan the streaming stock price data samples as them arrive and then drop them immediately, without storing. Certainly, it is also possible to apply the methods proposed in Chapter 5 to discover similar patterns of different stocks and explain why these patterns happened.

Finally, Chapter 7 summarizes the work presented in this dissertation. It contains a brief description of the new algorithms introduced in this dissertation, and points out a few important directions for future research.

Chapter 2

Related Work and Research Horizon

2.1 Data Processing Model

Because data streams come continuously and unboundedly, the analysis results of data streams often keep changing as well. The important issue in data processing model, is finding a way to extract data for knowledge discovery from the overall data streams. According to the research in [?], there are three stream data processing models:

Landmark Window Model In a landmark window, all stream objects that have been observed up to the current time are contained in the window. All objects contribute with equal weight to the result, regardless of their time stamp. A landmark window increases over time as a new stream object arrives. An example is illustrated in Figure **??**.

Examples of algorithms for frequent itemset mining and clustering using a landmark window model are proposed by Manku and Motwani in [?] and Guha et al. in [?], respectively. The benefit of the landmark window model is that stream objects are simply accumulated over time, as there is no need to remove objects from the window.

Damped Window Model In a damped window, the influence of stream objects on the stream mining result fades over time according to a user-defined fading function, which is monotonically decreasing. Objects in the damped window contribute to the mining result with decreasing weight as their age. Older data samples contribute less weight toward the pattern emerging in recent data. An example is illustrated in Figure **??**.

Examples of algorithms for frequent itemset mining and clustering using a damped window model are presented by Chang et al. in [?] and Aggarwal et al. in [?], respectively. The benefit

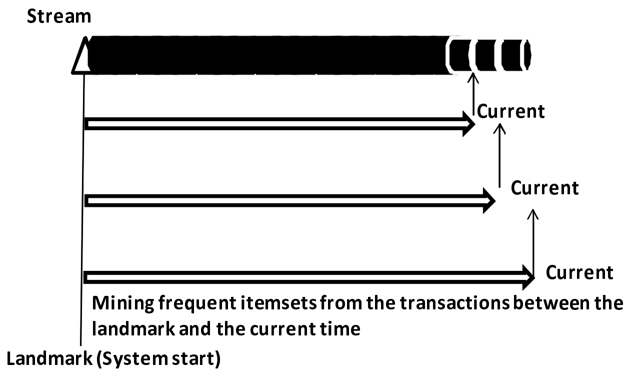

Stream
Current
Current
Current
Mining frequent itemsets from the transactions between the landmark and the current time
Landmark (System start)

Figure 2.1: Landmark Window Model.

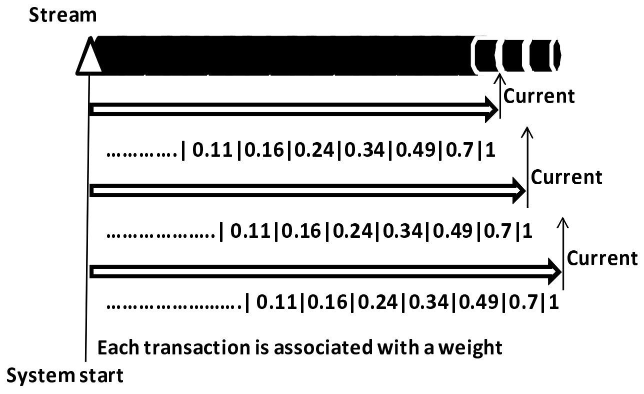


Figure 2.2: Damped Window Model.

of a damped window is that this model considers different weights for new and old data samples. This is suitable for application in which data has an effect on the mining results, but the effect decreases as time goes on.

Sliding Window Model A sliding window has a fixed size, and it slides over the data while new stream objects arrive in order to contain the most recent objects. The size of sliding window may be decided according to applications and system resources. The mining result of the sliding window method totally depends on recently generated data samples in the range of the window; all the data samples in the window need to be maintained in order to remove their effects on the current mining results when they are out of range of the sliding window. An example is illustrated in Figure **??**.

Examples of algorithms for frequent itemset mining using a sliding window model are proposed

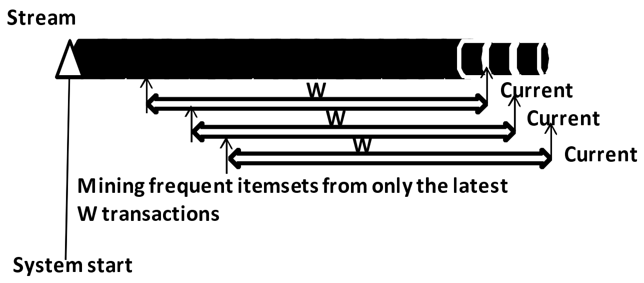

Stream
W
Current
W
Current
W
Current
Mining frequent itemsets from only the latest W transactions
System start

Figure 2.3: Sliding Window Model.

in [?,?,?]. The benefit of a sliding window is that because only recent stream objects are considered when computing a data mining result, changes in the properties of stream objects will be reflected in the mining results much faster than with a landmark window model.

All these three models have been used in current researches on data stream mining. Choosing which kind of data process models to use, largely depends on the needs associated with the specific application. In this dissertation, in Chapter 3, we mine quantifiable correlated patterns mining over streaming sliding windows, for the reason that we target to applications of mining transaction records data in a certain period of time, and we assume that the transactions in the referred period have equal importance to the results, such as analysis of market basket data of customers' purchases. In Chapter 4 and Chapter 5, we basically adopt the landmark process model to monitor and analyze the entire data streams for discovering underlying correlations. We propose a generalized "goodness" measurement for demanding a better model of correlation mining as well as for discovering of concept drifts in Chapter 4. On the other hand, in Chapter 5, the proposed summarization maintenance hierarchy is useful to satisfy end-users' demands for recent-biased data analysis. We can apply our methods to the online streams such as stock tickers, sensor data and network measurements. In the finally proposed methods for news articles analysis-based prediction of stock prices, elaborated in Chapter 6, we utilize the sliding window models for identifying trends of stock price movement.

2.2 Correlated Patterns Mining in Streaming Transaction Data

2.2.1 Knowledge Discovery in Static Transaction Data

In the 1990s, Aggrawal and Srikant developed association rule mining algorithms from static transaction databases [?]. An association rule tends to capture a certain type of correlation among items. Taking the market basket database for example, let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items (products), and DB be the set of purchase transactions. Each transaction T consists of a set of items such that $T \subseteq I$. Let X be a set of items, referred to as an itemset. A transaction T is said to contain X if and only if $X \subseteq T$. The *support* of an itemset X in DB, denoted as $sup(X)$, is the number of transactions in DB containing X. Then the association rule is often in the form of

$$X \Rightarrow Y, \tag{2.1}$$

where Y is another itemset, and there is no overlap between X and Y. In [?], the association rule in expression (**??**) is significant, if and only if

1. $sup(x \Rightarrow Y) = sup(XY)$, the number of transactions in DB containing both X and Y, is larger than a minimum support s;

2. $conf(x \Rightarrow Y) = \frac{sup(XY)}{sup(X)}$, the number of transactions containing X also contain Y, is larger than a minimum confidence c.

Here, we can see that the bigger the value of support statistic is, the more attentions would be paid on the association rule. Meanwhile, the confidence statistic specifies the degree of correlation among the itemset X and Y.

A successful example is that *Wal-Mart* mined association rules on their market basket data to reveal that "A customer purchasing *nappies* often also purchases *beer*" (i.e., *nappies* $\Rightarrow$ *beer*). Check of the store loyalty card details highlighted that there were young fathers who were probably sent out of the house by their partners to purchase the nappies but in the meantime decided to treat themselves to some beer. *Wal-Mart* took the obvious action and placed beer at the ends of the baby section aisles and increased their profits.

Table 2.1: Contingency table of the purchase of *Tea* and *Coffee* [?]

	Tea = Y	Tea = N	Row Sum
Coffee = Y	20	70	90
Coffee = N	5	5	10
Col. Sum	25	75	100

However, there are some problems with the above *support-confidence* framework for identifying the significance of association rules. If the threshold of minimum support s is low, the mining results may contain too many weakly-related *cross-support* patterns [?]. Here, the cross-support patterns denote the spurious patterns involving items with substantially different support levels. For example, as illustrated in Table **??**, the numbers represent percentages of purchases of *Tea* and *Coffee* in market basket data. In this table, columns "$Tea = Y$" and "$Tea = N$" correspond to the purchases of *Tea* and *non-Tea*, respectively. Similarly, rows "*Coffee* $= Y$" and "*Coffee* $= N$" correspond to *Coffee* and *non-Coffee*, respectively. According to the *support-confidence* framework [?], association rule

$$Tea \Rightarrow \mathit{Coffee} \tag{2.2}$$

has a 20% support and an 80% confidence. With fairly high support and confidence, we may consider it as a valid rule and believe that customers who buy tea will also buy coffee. However, considering the fact that the support of *Tea* is much lower than the support of *Coffee*, and the priori probability that a customer buys coffee is 90%, then we can see that the association rule is incomplete information. In fact, this association rule is misleading as pointed in [?]. Because the ratio

$$\frac{P\{[Tea = Y] \wedge [\text{Coffee} = Y]\}}{P\{[Tea = Y]\} \times P\{[\text{Coffee} = Y]\}} = \frac{0.2}{0.25 \times 0.9} = 0.89 < 1, \tag{2.3}$$

therefore, *Tea* and *Coffee* are actually negatively correlated. On the other hand, if the threshold of minimum support is high, the support-confidence framework may miss many rarely occurring but important patterns, such as diseases, network intrusions, earthquakes and so on.

In order to address this problem, some other correlation measures are proposed instead of the *support* measure. *All-confidence* is adopted for dependency measure of association rules in [?, ?, ?]. The *all-confidence* measure of a pattern X is defined as the minimum confidence of all the association rules that can be derived from X, and also can be calculated according to the following equations:

$$\begin{aligned} \textit{all-conf}(X) &= \frac{sup(X)}{\textit{max-item-sup}(X)}, && (2.4) \\ \textit{max-item-sup}(X) &= \max_{\forall i_j \in X} sup(i_j). && (2.5) \end{aligned}$$

Then association rules are extracted from the pattern X whose all-confidence is larger than a threshold of minimum all-confidence α.

There are many other solutions for defining significance of association rules. *H-confidence* proposed in [?] has a slight difference from the all-confidence. *H-confidence* examines only rules where there is only one item on the left-hand side of the rule. In [?], the authors also proved that *Coherence* measure was also appropriate for measuring correlation, for the reason that it is not influenced by the co-absence of items pairs in the transactions. *Mutual Dependence* is proposed in [?] for definition of significant mutually dependent itemsets X and Y based on the empirical conditional probability. Then the justification of X and Y which are strongly dependent with each other is characterized by a minimum dependency probability. Similarly, authors in [?, ?] proposed information-theory-based *Normal Mutual Information* measure for identifying depended itemsets.

Although it has been shown in many studies [?, ?, ?] that *all-confidence* reflects dependency relationships among items more accurately than other measures, but in some cases the *all-confidence* measure fails to mine correlated itemsets. For instance, suppose we have a pair of items $\{A, B\}$, where $sup(A) = sup(B) = 0.8$ and $sup(AB) = 0.64$. Factually, A and B are uncorrelated because $sup(AB) = sup(A) \times sup(B)$. Unfortunately, the *all-confidence* measure cannot filter out the uncorrelated items A and B, for the reason that the

$$\textit{all-conf}(AB) = \frac{sup(AB)}{\max\left(sup(A), sup(B)\right)} = 0.64/0.8 = 0.8 \qquad (2.6)$$

is significantly high. In contrast, another pair of items $\{\acute{A}, \acute{B}\}$ with $sup(\acute{A}) = sup(\acute{B}) = sup(\acute{A}\acute{B}) = 0.001$ is perfectly correlated despite low supports of $\acute{A}$ and $\acute{B}$.

In [?], Brin et al. proposed to measure the significance of association via the chi-squared test for correlation from classical statistics. However, the approximation of correlation based on chi-square statistics breaks down when the expected values are small, and the correlation rule is less accurate if the contingency table data are sparse. However, this work supports negative implications among patterns. For example, fire code inspectors trying to mine useful fire prevention measures may like to know any negative correlation between certain types of electrical wiring and the occurrence of fires.

In this dissertation, we agree to discover correlated pattens based on Pearson's correlation coefficient [?], and ϕ correlation coefficient is the computation form of Pearson's correlation coefficient for binary items. We can derive the support form of the ϕ correlation coefficient of two items A and B as:

$$\phi = \frac{sup(AB) - sup(A) \times sup(B)}{\sqrt{sup(A)sup(B)(1 - sup(A))(1 - sup(B))}} \tag{2.7}$$

As a result, we can see that the ϕ correlation coefficient is able to filter out the above examples of uncorrelated pattern $\{A, B\}$, and identify the rarely occurring but completely correlated pattern $\{\acute{A}, \acute{B}\}$. Therefore, correlated patterns discovery based on calculation of ϕ correlation coefficients are not restricted to frequently co-occurring items. Moreover, those infrequent but significant patterns that are too expensive to be obtained by association rule mining can also be discovered by calculation of correlation coefficients.

On the other hand, existing researches on correlation mining are primarily conducted on binary databases. However, most items in real-life databases are not restricted to taking only Boolean values. Instead, these items can be quantitative, which are numeric values (e.g., an employ's salary). The expressiveness of quantitative items aggravates the complexity of mining quantitative databases due to the large domain size of the items. We categorize the quantitative association rules of related works into two classes: one is numeric interval-based association rules [?, ?, ?, ?, ?, ?, ?], for example,

$$bread : [2 - 5] \Rightarrow butter : [1 - 2]; \tag{2.8}$$

on the other hand, an instance of ratio rules-based association [?, ?, ?] is shown as

$$bread : butter = 2 : 3. \tag{2.9}$$

These examples target to mine association rules from quantitative market basket database, where the amount of money (dollars) spent on products is recorded in each transaction. The former numeric interval-based rule indicates that "the amount of money which a customer spent on bread is within the range from 2 dollars to 5 dollars on bread, then he would spend larger than 1 dollar and less than 2 dollars on butter". However, the later ratio rule between *bread* and *butter* uncovers the ratio of the money spent on *bread* to that of *butter* equals 2 : 3. As discussed in [?], the ratio rules are much more informative than numeric interval-based rules:

- Ratio rules achieve more compact descriptions if the data points are linearly correlated. In the above two examples, ratio rules could answer the questions like "how much money would a customer spend on butter, if he had spent 6 dollars on bread". However, the numeric interval-based association rules cannot answer.
- Related to the above bullet point, ratio rules can perform extrapolations and predictions.

Additionally, ratio rules among prices of products can also be useful for determining positive and negative product associations in [?].

In more detail, [?] employed an eigensystem analysis to calculate correlations among items. The eigensystem analysis regarded the axes of greatest variation as the most important correlations. However, the eigensystem analysis was susceptible to noise; therefore, in [?], the authors proposed a robust and adaptive ratio rules mining algorithm for distributed and changing data sources. On the other hand, similar to the support-confidence framework for association rules mining in [?], algorithms in [?] are able to discover multiple frequent and confident ratio rules among pairs of items. For example, in Figure ??, the black solid line represents the only one ratio rule extracted by eigensystem analysis, while we can see that there are two potential kinds of linear relationships, called local ratio rules in [?] between variables X and Y.

2.2.2 Correlation Mining in Streaming Transaction Data

In terms of streaming transaction data, due to the technical challenges of data streams environment, existing studies in [?, ?, ?, ?, ?] achieved to approximate frequent itemsets with single scan of data. As we discussed in the above subsection, the problems of support-based pattern mining also exist in the streaming transaction data. However, corresponding correlation patterns mining algorithms for static data exist, but no work has been done so as to complete the same task for data streams.

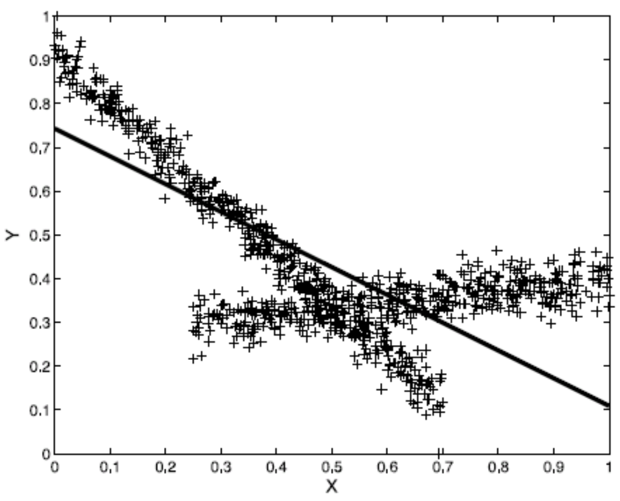


Figure 2.4: An example of multiple ratio rules

In this dissertation, we contribute to mine *Quantifiable Correlated Patterns (QCP)* to reveal the quantitative association among correlated items in streaming transaction data. Unfortunately, all of the above related techniques for static databases cannot be extended to the data stream environment easily. Firstly, most of the related techniques need level-wise processes for mining correlated itemsets, and consequently need to scan databases in multiple times. Although [**?**, **?**] realized correlation mining on *FP-tree* proposed in [**?**] in terms of *all-confidence* measure, transformation of a database into a *FP-tree* stored in main memory needs 2 times scan of database. Meanwhile, there is a combinatorial explosion problem for calculating ϕ correlation coefficients. Given n items, it is necessary to calculate correlation coefficient of all the $(2^n - 1)$ possible itemsets (for a data set with a million of items, the total number of possible itemsets is nearly a trillion). However, it is often the case for real-world data that the total number of highly correlated items is much smaller than the total number of possible itemsets. Therefore, storing all the itemsets and computing all the correlation coefficients are wasteful tasks and may be impossible in some cases. Secondly, there is a more severe combinatorial explosion problem of infinite different potential frequent ratio patterns among items, due to the real values of ratio patterns. These two combinatorial explosion problems result in high complexity for mining quantitative transaction data stream, and can severely degrade the mining efficiency.

As shown in Figure **??**, our proposed method firstly achieves to mine both of the correlations and quantitative associations in data stream environment with only single scan of data and limited memory. We devise an efficient method using *min-hashing* technique to find possibly correlated itemsets without counting supports, and use *ratio ranges* to enforce quantitative associations among itemsets. In Chapter 3, we elaborate our proposed methods for these challenges.

2.3 Correlation Discovery over Time-series Data Streams

2.3.1 Correlated-clusters Mining

Most of the work on stream mining has focused on finding interesting streaming patterns from multiple data streams. CluStream [?] is a flexible clustering framework with online and offline components. The online component extends micro-cluster information by incorporating exponentially-sized sliding windows while coalescing micro-cluster summaries. Actual clusters are found by the offline component. However, clustering over high dimensional data streams poses dual challenges to traditional clustering algorithm: (1) all pairs of data samples tend to be almost equidistant from one another due to sparsity of data [?]; and (2) continuous changes exist in data distribution and underlying correlations. Perlman and Java [?] have proposed a two phase approach to mine astronomical time series streams. The first phase clusters sliding window patterns of each time series. Using the created clusters, an association rule discovery technique is used to create affinity analysis among the created clusters of time series.

On the other hand, correlation mining over multiple time-series for knowledge discovery has also been proposed in many studies. Several studies report highly correlated pairs of data streams. StatStream [?] used the Discrete Fourier Transforms (DFT) to summarize streams within a finite window and then compute the highest pairwise correlations among all pairs of streams, at each time stamp. The system works over an arbitrarily chosen sliding window. BRAID [?] addressed the problem of discovering lag correlations among multiple data streams. The focus is on time and space efficient methods for finding the earliest and highest peak in the cross-correlation functions between all pairs of streams. Yeh et al. [?] proposed online and efficient algorithms to cluster multiple data streams based on calculation of pair-wise correlation.

Moreover, incremental learning of correlations on the whole collection of streaming data has attracted much attention in the last few decades. The algorithms for Incremental Principal Component Analysis (IPCA) [?] have been proposed since 1970s. Most of them are based on the Singular Value Decomposition (SVD). The main difference among the several SVD-based IPCA algorithms is how to express the covariance matrix incrementally. However, the incremental algorithms by SVD are not applicable to data streams since the computational complexity is very high. Guha et al. [?] improves on discovering correlations, by first doing dimensionality reduction with random projection, and then periodically computing the SVD. However, the method incurs high overhead because of the SVD re-computation and it cannot easily handle missing values. Recently, CCIPCA [?] has been proposed as an algorithm for unsupervised incremental feature selection. It incrementally computes the principal components of data sequences using an estimate. This estimate is efficient for some well-known distribution (*e.g.,* Gaussian). However, in data streams environment, data distribution is always unknown; therefore, the highest efficiency is not guaranteed. Authors of [?] proposed an incremental and robust algorithm. In this algorithm, the square criterion as used in PCA is replaced by Steady Criterion Function (SCF). However, this algorithm assumes that the correlations among features do not change. This assumption is in conflict with the characteristic of data evolution in data streams.

In Chapter 4, we propose a new IPCA method to learn correlations incrementally and adaptively. This method requires very limited memory and short processing time per time tick. Additionally, corresponding to the data evolution characteristic of data streams, our proposed generalized multivariate regression measure is responsible for detecting the change of correlation.

2.3.2 Flexible Timeline Clustering

Besides of summarizing the correlation among multiple streaming time series, some studies report to apply clustering techniques to multiple steaming data periodically for discovering cross-relations [?, ?]. In this dissertation, we support clustering data streams at any time during the whole collection of streaming time-series data. In the users' requests of clustering, users specify the interested time periods. Here we call it timeline clustering of data streams. Therefore, unlike prior studies [?, ?, ?], the objective in this work is to partition data streams (variables), rather than their data samples, into clusters. Therefore, we can discover groups of streams with similar behavior. The clustering over evolving streams is also discussed in [?]. However, the objective in [?] is to continuously report clusters satisfying the specified distance threshold. The clusters are generated according to the values from the beginning of the stream to the current time. Therefore, it does not have the ability to observe correlations within a period of interest.

In order to realize flexible clustering, in the context of data streams, how to summarize the huge number of data resources for online calculation of similarity, and how to efficiently retrieve abstractions of streaming data during a certain period in response to users' queries are important issues. Several high-level representations have been proposed for approximation of time series, including Discrete Fourier Transform [?], Discrete Wavelet Transform [?], Singular Value Decomposition [?] and Piecewise Linear Approximation (PLA) [?]. Among these representations, PLA which approximates data samples into linear segments is the one that is widely used because of its simplicity and is applicable to online approximation. Additionally, for the PLA linear approximation, due to the advantages of smooth approximation and low computational complexity discussed in [?], linear interpolation becomes our technical choice for online segmentation algorithm. A classic SW method in [?] is proposed for online segmental approximation of subsequences. In Chapter 5, we introduce a new segmentation criterion called *tolerant slope interval* to reduce the computational complexity of the classic SW method.

In Chapter 5, we devise a framework to dynamically and flexibly cluster multiple evolving data streams for flexible timeline clustering requests. The proposed framework consists of two phases: *online statistics maintenance phase* and *offline clustering phase.* The online statistics maintenance phase realizes online collection and maintenance of summary statistics of fast data streams. Once a clustering request is submitted, the offline clustering phase devises an adaptive abstraction algorithm to abstract statistics for approximating the user-desired subsequences as precisely as possible from the summary statistics hierarchies, and outputs the results of correlated streaming data by clustering over the statistics.

2.4 Stock Price Prediction Based on Textual Information

So far, there have been many studies related to the examination about impacts of different textual information on the financial markets. Most of the related works treat the analysis of textual data as a text classification problem. The outputs are used as indicators of forthcoming trends (directions) in price movements. The first systematic method was conducted in [?], which compared the movements of Dow Jones Industrial Average with general news during the period from 1966 to 1972. Peramunetilleke et al. [?] investigated how the market news headlines can be used to forecast intra-day currency exchange rate movements; [?] claimed that Internet stock message boards can help to predict financial market volatility, and that disagreement among the posted messages is associated with increased treading volume. Choudhury et al. [?] developed a simple model to analyze communication dynamics in blogosphere and use those dynamics to determine interesting correlations with stock market movement; in [?] the authors studied a new problem to predict the risk of stocks by their corresponding news of companies, and so on.

News articles as one of the popular public information resources, potentially contain useful information which can be used to forecast movements of stock prices. In this dissertation, we focus on the influence of news articles on the behavior of stock prices data. [?] developed an algorithm to classify each given news article into the predefined classes (manually labeled 5 classes) in terms of the referred company's financial well-being. In order to address the problem of expensively labeled training data, the authors used a "Self-Confident" sampling method and a vote entropy based criteria to assign a label to an unlabeled article automatically.

On the other hand, in many studies the labeled training data is related with movements in numeric stock prices series, [?] targeted to predict the trends (up, down and steady) of closing values of Hang Seng Index versus that of previous day. Some probabilistic rules were generated using the approach in [?]. Lavrenko et al. in [?] and [?] concluded that piecewise linear regression based on *t-test* criteria is a useful tool for describing trends in time series, and demonstrated that Bayesian language models represent a good framework for associating news articles with forthcoming trends. Similar to the preprocessing of time series in [?], [?] used a *t-test* based segmentation algorithm consisting of splitting and merging for stock prices series preprocessing and SVM [?] for impact classification of news articles on the stock prices. [?] aimed to discover relationship between news and abnormal stock price behavior. This work categorized articles whose incidence correlates with abnormal forecast errors as interesting, and then discovered that classifying these interesting news improves the performance more than using all news. Gidofalvi et al. in [?] defined the window of influence of news articles, and found definite predictive power for the short-term stock price movement in the interval starting 20 minutes before and ending 20 minutes after news articles had become publicly available. Specially, the movements of stock prices in [?] are defined based on β-value. NewsCATS system proposed in [?] could automatically analyze and categorize press releases, and also generate stock trading recommendations.

In contrast to the above studies which predict the short-term (from minutes to hours within a day) financial market, the authors in [?] proposed to do regression analysis using monthly reports of Bank of Japan as well as monthly price data, and forecast in higher accuracy in terms of both the magnitude and the direction of long-term market trends.

In Chapter 6, we realize a more flexible and accurate investigation of the influences of online news articles to stock prices:

1) Identification of trends in stock prices over sliding windows satisfies both short-term and long-term needs of prediction;

2) Representation of news articles as vectors of terms weighted with class relevance and discrimination improves the accuracy of prediction;

3) Dynamic choice of sliding windows in terms of the consecutive news articles for identifying trends of stock prices realizes online investigation of the influence of news articles to stock prices;

4) Prediction of forthcoming stock prices in terms of direction and magnitude is much more informative and helpful for investors' decision-making.

Chapter 3

Quantifiable Correlated Patterns Mining

We are able to achieve *bi-level* quality control in mining Quantifiable Correlated Patterns from streaming transactional data. First, we devise an efficient method using *min-hashing* technique to find possibly correlated attributes without counting supports. Second, we use *ratio ranges* to enforce quantitative association among attributes. The proposed methods address the key challenges of two combinatorial explosion problems for mining correlated attributes as well as frequent ratio relations among these correlated attributes.

3.1 Introduction

In this chapter, we focus on correlation discovery from quantitative streaming transaction data. For example, the market basket data which record the purchase of customers, where quantitative values of each attribute may be the amount of money spent on each product. We model the regular bulks of streaming transaction data as sliding windows, and aim to mine correlation patterns from each sliding window. We propose the novel notion of Quantifiable Correlated Pattern (QCP) to describe the frequent ratio relationships among highly correlated attributes. Considering the computation and space constraints in data streams environment, there are two combinatorial explosion challenges inherent to the two subtasks: generating highly correlated attributes and mining frequent ratios among the correlated attributes.

Given a set of m items, it is necessary to calculate correlation coefficients of all the $(2^m - 1)$ possible itemsets. However, it is impossible to scan transactions in multiple times for calculating correlation coefficients. Furthermore, because the ratios are real data, it is also impossible to count the supports of all the possible ratios which are actually infinitely different values for mining frequent ones. In order to address these combinatorial explosion problems, we make the following contributions:

(1) We propose a rule for generating candidates of correlated attributes based on Jaccard distance.

(2) According to the above rule, we devise an efficient method using min-hashing technique [?] to estimate Jaccard distance without counting supports of all the possible attributes. This method is efficient and effective to reduce the number of candidates greatly.

(3) By introducing a concept of ratio range and downward closure property, we can enumerate and prune candidate ratios patterns efficiently.

(4) Proposed methods of the two subtasks use limited memory space in a single on-line scan of transactions.

The rest of this chapter is organized as follows: Section 3.2 gives a formal definition of the problem for mining quantifiable correlated patterns. In Section 3.3 and Section 3.4, we elaborate the proposed methods for solving the two subtasks, respectively. In Section 3.5 we discuss our experimental results. Finally, we conclude this chapter in Section 3.6.

3.2 Problem Statement

In this section, we give a formal problem statement of mining quantifiable correlated patterns over a sliding window of steaming transaction data.

Let $\Sigma = \{a_1, a_2, \cdots, a_m\}$ be a set of quantitative attributes. A set of transactions W of the recent sliding window wherein each transaction T is a subset of Σ and their values. The value of attribute a_i is a real number, represented by $v(a_i)$. Figure **??** is an example of market basket data with window size $|W| = 4$. Each transaction has a time stamp, which is used as the *tid* (transaction ID) of the transaction. Here, the attributes a_i can be the purchased products by customers, and consequently the number of attributes represents the amount of money spent on the products.

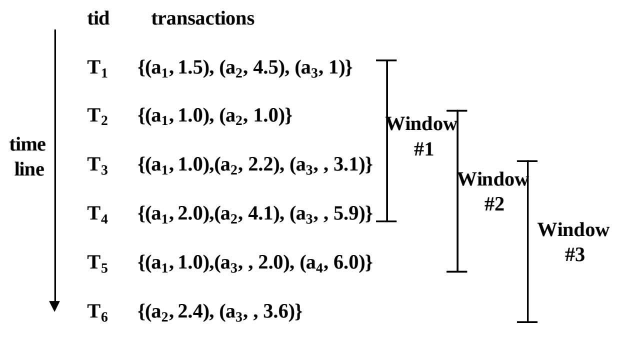

tid
transactions
time line
T1 {(a1, 1.5), (a2, 4.5), (a3, 1)}
T2 {(a1, 1.0), (a2, 1.0)}
T3 {(a1, 1.0),(a2, 2.2), (a3, , 3.1)}
T4 {(a1, 2.0),(a2, 4.1), (a3, , 5.9)}
T5 {(a1, 1.0),(a3, , 2.0), (a4, 6.0)}
T6 {(a2, 2.4), (a3, , 3.6)}
Window #1
Window #2
Window #3

Figure 3.1: A running example of quantitative transactional data.

$T_1 = \{(a_1, 2.5), (a_2, 3), (a_3, 1)\}$ is a transaction of purchase, where $v(a_1) = 2.5, v(a_2) = 3$ and $v(a_3) = 1$.

A pattern P consists of a set of attributes and the ratios of the attributes' values. Attributes in P are in alphabetic order. Additionally, we require the ratio of form $(1 : r_2 : \cdots : r_m)$, where the first value is fixed to 1 and ratios of the other attributes are calculated against the value of the first item. For example, in pattern $P = \{a_1, a_2, a_3, 1 : 2 : 3\}$, $v(a_2) : v(a_1) = 2 : 1$ and $v(a_3) : v(a_1) = 3 : 1$. The size of a pattern is defined as the number of attributes in the pattern.

Definition 1 (*Quantifiable Correlated Patterns*) Given a pattern $P = \{a_1, a_2, \cdots, a_m, 1 : r_2 : \cdots : r_m\}$, it is a quantifiable correlated pattern if it satisfies the following conditions:

- $(a_1, a_2, \cdots, a_m)$ are highly correlated *w.r.t* δ.
- $(1, r_2, \cdots, r_m)$ is frequent *w.r.t* s and d.

Problem Statement The problem is to mine quantifiable correlated patterns in sliding windows consisting of $|W|$ transactions within continuous and streaming transaction data.

Therefore, we do two phases to mine the quantifiable correlated patterns. One is to mine highly correlated attributes, and the other is the mine frequent ratio associations among attributes. Additionally, the conduction of the first phase can be synchronized with the second phase. To this end, we define the following definitions.

Definition 2 (*Highly Correlated Attributes*) Given a user-specified *minimum correlation coefficient* δ ($\delta \in (0, 1)$), a set of attributes $\{a_1, a_2, \cdots, a_m\}$ is highly correlated with each other if all of the pairs' correlation coefficients are greater than δ.

Definition 3 (*Tolerant Support Counting of ratio patterns*) Given a tolerance bounding d ($0 \leq d \leq 1$) on ratios, a transaction T supports a ratio pattern $P = \{a_1, a_2, \cdots, a_m, 1 : r_2 : \cdots : r_m\}$ *w.r.t* d, if

- $\forall a_i \in P (1 \leq i \leq m)$, a_i appears in T; and
- $v(a_i)/v(a_1) \in [r_i(1-d), r_i(1+d)]$ for $(2 \leq i \leq m)$.

The support of pattern P in W is the fraction of the total number of transactions that supports pattern P satisfying specified fault tolerance bounding d.

Definition 4 (*Frequent Ratios*) Given a pattern $P = \{a_1, a_2, \cdots, a_m, 1 : r_2 : \cdots : r_m\}$ and a user specified *minimum support* s ($s \in (0,1)$), ratio $(1 : r_2 : \cdots : r_m)$ is a frequent ratio among the attributes set $\{a_1, a_2, \cdots, a_m\}$, if the support of P with respect to a tolerance bounding d is greater than s.

3.3 Generation of Highly Correlated Attributes

In the process of generating highly correlated attributes, we treat the attributes in the data as binary attributes. Normally, the number of highly correlated attributes is quite small in comparison with the total number of sets. Therefore, it is wasteful to compute correlation coefficients for all possible attributes. Here, we propose a candidate generation rule based on Jaccard distance, and devise an efficient generation method using min-hashing functions [?].

3.3.1 Candidate Generation Rule

For an arbitrary attribute a, we denote $U(a)$ as the set of transactions that contain a, and $sup(a)$ as the *support* of a as defined in Section 2.2.1. For a pair of attributes a and b, the Pearson correlation coefficient (also called the ϕ correlation coefficient for binary attributes) can be expressed in terms of the supports:

$$\phi_{(a,b)} = \frac{sup(ab) - sup(a)sup(b)}{\sqrt{sup(a)sup(b)(1-sup(a))(1-sup(b))}} \tag{3.1}$$

We are looking for pairs (a, b) such that $\phi_{(a,b)} \geq \delta$. Intuitively, we know that, if a and b are highly correlated, the set $U(a) \cap U(b)$ should not be too small, compared to $U(a)$ or $U(b)$. We now establish this relationship formally. Without loss of generality, we assume that $sup(b) \geq sup(a)$ (equivalently $|U(b)| \geq |U(a)|$). Assume that a and b are highly correlated:

$$\phi_{(a,b)} = \frac{sup(ab) - sup(a)sup(b)}{\sqrt{sup(a)sup(b)(1 - sup(a))(1 - sup(b))}} \geq \delta \tag{3.2}$$

Because $sup(a) \geq sup(ab)$, replacing $sup(ab)$ with $sup(a)$ in (**??**), we obtain

$$\sqrt{\frac{sup(a)(1 - sup(b))}{sup(b)(1 - sup(a))}} \geq \delta \tag{3.3}$$

Let $S = \sqrt{\frac{sup(a)(1-sup(b))}{sup(b)(1-sup(a))}}$. By Inequality (**??**), $S \geq \delta$. By the assumption that $sup(a) \leq sup(b)$, $S \leq 1$. From (**??**) and the fact that $sup(ab) = \frac{|U(a) \cap U(b)|}{|W|}$, we transform

$$\begin{aligned}
\frac{|U(a) \cap U(b)|}{|U(b)|} &= \frac{sup(ab)}{sup(b)} \\
&\geq \frac{\delta \cdot \sqrt{sup(a)sup(b)(1 - sup(a))(1 - sup(b))} + sup(a)sup(b)}{sup(b)} \\
&= \delta \cdot \sqrt{\frac{sup(a)(1 - sup(b))}{sup(b)(1 - sup(a))}} \cdot (1 - sup(a)) + sup(a)
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\frac{|U(a) \cap U(b)|}{|U(b)|} &\geq \delta \cdot \sqrt{\frac{sup(a)(1 - sup(b))}{sup(b)(1 - sup(a))}} \cdot (1 - sup(a)) + sup(a) \\
&\geq \delta \cdot S \cdot (1 - sup(a)) + sup(a) \\
&= \delta \cdot S + (1 - \delta \cdot S)sup(a) \\
&\geq \delta \cdot S
\end{aligned}$$

The last inequality comes from $\delta \leq 1$, $S \leq 1$. Similarly,

$$\begin{aligned}
\frac{|U(a) \cap U(b)|}{|U(a)|} &\geq \delta \cdot \sqrt{\frac{sup(b)(1 - sup(a))}{sup(a)(1 - sup(b))}} \cdot (1 - sup(b)) + sup(b) \\
&\geq (\delta/S) \cdot (1 - sup(b)) + sup(b) \\
&= \delta/S + (1 - \delta/S)sup(b) \\
&\geq \delta/S
\end{aligned}$$

Here the last inequality comes from $S \geq \delta$. Now consider the ratio $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}$. We have

$$\begin{aligned}
\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|} &= \frac{|U(a)\cap U(b)|}{|U(a)|+|U(b)|-|U(a)\cap U(b)|} \\
&= \frac{1}{\frac{|U(a)|}{|U(a)\cap U(b)|}+\frac{|U(b)|}{|U(a)\cap U(b)|}-1} \\
&\geq \frac{1}{S/\delta + 1/(\delta\cdot S) - 1} \\
&= \frac{\delta}{S+1/S-\delta}
\end{aligned}$$

Given $\delta \leq S \leq 1$, $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}$ achieves its minimum value of δ^2 when $S=\delta$. Therefore, our rule is thus to prune when $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|} < \delta^2$.

Candidate-Generation Rule: We put the pair (a,b) into the candidate set only when $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|} \geq \delta^2$. That is, (a,b) is selected if the Jaccard distance between $U(a)$ and $U(b)$ is smaller than $1-\delta^2$.

Note that this bound is tight. That is, if we prune a pair (a,b) when $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}$ is slightly larger than δ^2, we may remove a pair whose correlation coefficient is slightly above δ. To see this, consider two attributes a and b with $U(a)\cap U(b) = U(a)$. Also, assume $sup(a)$ and $sup(b)$ are very small. In this case, $(1-sup(a)) \to 1$, and $(1-sup(b)) \to 1$. The correlation coefficient reduces to $\sqrt{\frac{sup(a)}{sup(b)}}$, i.e., $\phi_{(a,b)} \approx \sqrt{\frac{sup(a)}{sup(b)}}$. Because $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|} = \frac{sup(a)}{sup(b)} = \delta^2$, $\phi_{(a,b)} \approx \delta$. Furthermore, $\phi_{(a,b)}$ increases with $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}$. Hence, if we prune a pair when $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}$ is above δ^2, we may have removed a pair whose correlation coefficient is above δ.

3.3.2 Candidate Generation Method

We describe a method that can efficiently prune (or generate candidate) according to our rule. The candidate-generation method uses *min hashing*, which was introduced in [?]. A min-hashing function h_{min} maps an attribute a in the data set to a number:

$$h_{min}(a) = min_{u\in U(a)}\{h(u)\}$$

$h(u)$ is a hashing function of transaction u which contains a.

EXAMPLE 1. *Assume that there are* 10 *transactions in the current sliding window. Table* **??** *represents the hashing values of these transactions. If we know that item a appears in transactions*

Table 3.1: An example of min-hashing.

u	0	1	2	3	4	5	6	7	8	9
h(u)	17	21	9	44	5	16	1	20	37	8

2, 5 *and* 8, *then*

$$h_{min}(a) = min\{h(2), h(5), h(8)\} = min\{9, 16, 37\} = 9$$

The min-hashing function has the following property:

Property 1 The probability for mapping attributes a and b to the same value by min-hashing can be expressed as

$$P(h_{min}(a) = h_{min}(b)) = \frac{|U(a) \cap U(b)|}{|U(a) \cup U(b)|}$$

According to the above property, we can see that the larger the ratio $\frac{|U(a) \cap U(b)|}{|U(a) \cup U(b)|}$ is, the more likely the fact that the two attributes will be hashed to the same value is. This suggests a simple candidate-generation method:

Candidate-Generation Method: We put attributes that have the same min-hashing value in the candidate set.

By the candidate-generation method, the pairs that satisfy our rule $(\frac{|U(a) \cap U(b)|}{|U(a) \cup U(b)|} \geq \delta^2)$ will be placed into the candidate set with high probability and the pairs that do not satisfy our rule $(\frac{|U(a) \cap U(b)|}{|U(a) \cup U(b)|} \leq \delta^2)$ will be selected with low probability.

However, the gap between the two probabilities is not wide enough to result in powerful pruning while maintaining a small false-negative probability. We use another technique from [**?**] to widen the gap. We use k independent min-hashing functions and define an equivalence relation "$\simeq$". For two attributes a and b, $a \simeq b$ if and only if a and b have the same min-hashing values for all the k hashing functions. The equivalence relation can be used to partition the attributes into equivalence classes. If, with one min-hashing function $P_1(a \simeq b) = x$, then with k independent functions, $P_k(a \simeq b) = x^k \ll x$. We repeat the whole process t times, each time with a different set of k min-hashing functions (totally $k \cdot t$ independent min-hashing functions are required). The probability that a and b belong to the same equivalence class in at least one of the trials is $1 - (1 - x^k)^t$. We put into the candidate set all the pairs whose two items belong to the same

Table 3.2: An example of k-min-hashing.

attributes	k-min-hashing values		
	t_1	t_2	t_3
a_1	3, 8	12, 7	4, 8
a_2	1, 7	9, 20	17, 6
a_3	3, 8	10, 22	17, 6
a_4	5, 7	7, 12	17, 6

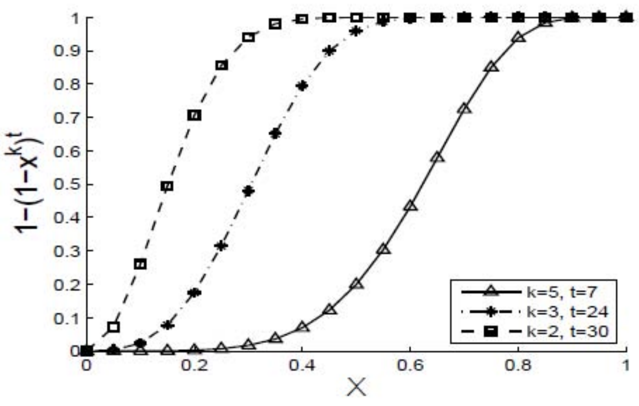


Figure 3.2: Shape of function $f(x) = 1 - (1 - x^k)^t$.

equivalence class.

EXAMPLE 2. *In Table* **??**, *we obtain* t_1, t_2 *and* t_3, *totally* $t = 3$ *trails of* 2*-min-hashing* ($k = 2$) *values of* 4 *attributes. i.e., in* t_1 *trail, the min-hash values of item* a_1 *for two different min-hashing functions are* 3 *and* 8. *Then we generate candidates of correlated attributes by finding sets of attributes which have the same* 2 *dimensional vector of min-hash values. As a result, we find the pair* (a_1, a_3) *in* t_1 *trail, none in* t_2 *trail, and* (a_2, a_3, a_4) *in* t_3 *trail. Both of the two sets of attributes are put into candidate sets.*

As illustrated in Example 2, it can be seen that our proposed method is capable of generating l-length ($l \geq 2$) sets of correlated attributes at the same time. In addition, the k-min-hashing of all attributes can be performed in just one scan of the transactions in a sliding window.

3.3.3 Discussion of False Negatives

The probability that attributes a and b are hashed to the same vector of hashing values in at least one of the trial is $1-(1-x^k)^t$, where x is the probability for one min-hashing function. We illustrate the function $f(x)=1-(1-x^k)^t$ for several values of k and t in Figure **??**.

From the figure, we can see that $f(x)$ is an "s" shape and approximates a threshold function, i.e., a function $g(x)$ that takes the value 1 when x is larger than the threshold and 0 when x is smaller than the threshold. $f(x)$ is an approximation of such a threshold function. We observe that k determines the sharpness of the transition from 0 to 1. t works with k to determine where the transition happens. For an ideal pruning result, we want the transition to be sharp. By choosing values for k and t properly, we can locate the candidate sets that satisfy our rule ($\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}\geq\delta^2$) with probability close to one and pairs that do not satisfy our rule with probability close to zero.

We can see that the parameters k controls the effectiveness of the pruning. By using a large k, we can prune more sets that do not satisfy our rule. In general, k can be determined by the limits on the available computational resources. While parameter t controls the probability of false negatives. The larger t is, the smaller the probability that we drop a set satisfying our rule is.

Once we have a value for k, we choose t according to the threshold δ and false-negative tolerance τ. We first discuss the meaning of τ and then describe how we choose its value. Note that, when we fix the values of both k and t, the stronger a set's correlation is, the more likely the fact that this set will be placed in the candidate set is. This is determined by the probability function in Figure **??**. A set (a,b) has the probability $1-(1-x^k)^t$ to be in the candidate set, where $x=\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}$. The stronger the set's correlation is, the larger $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}$is, and therefore the set is more likely to be in the candidate set. In other words, our candidate generation favors strongly correlated sets. We choose t such that, for a pair (a,b), $\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}=\delta^2$, the probability of omitting this pair from the candidate set is below τ (This is why we call τ as the "false-negative tolerance"). Note that this does not mean that our candidate generator misses every highly correlated set with a probability τ. Only the pairs (a,b) that are at the border ($\frac{|U(a)\cap U(b)|}{|U(a)\cup U(b)|}=\delta^2$) have probability of τ being left out. The other sets with stronger correlation have smaller false-negative probability than τ. The formula for choosing t is then $t=\log_{1-\delta^{2k}}\tau$.

3.3.4 Algorithm for Finding Highly Correlated Attributes

We describe algorithm for generating highly correlated itemsets in Figure **??**. For transactions in a sliding window, we do the process using a hash table. The hash table takes a vector of size k as a key and hashes the attributes into one of its buckets. The attributes in the same bucket are correlated. This process is repeated t times. When a new transaction arrives, we calculate the hashing values of the new transaction, and compare them with the currently stored minimum value. The current minimum is replaced if necessary.

3.4 Mining Frequent Ratios among Attributes

In fact, the process of mining frequent ratios is proceeded concurrently with the mining of highly correlated sets of attributes.

3.4.1 Ratio Range for Pruning

Because real numerical ratios can take any values, it is impossible to assign counters to infinite ratios. The only way is first to find out the ratios occurring in the data, and then to figure out ratios with enough support.

Firstly, we consider the question that what is the ratio r in a 2-sized pattern $P = \{a_1, a_2, 1 : r\}$ which is supported by all transactions in sliding window W. We denote the minimum value of r is r_{min}, and the maximum value is r_{max}. Suppose all transactions in W support the pattern P, then according to Definition 3 of *tolerant support*, we have

$$r(1-d) \leq r_{min} \leq r(1+d),$$

and

$$r(1-d) \leq r_{max} \leq r(1+d).$$

where d is the ratio tolerance bound. That is

$$\frac{r_{max}}{1+d} \leq r \leq \frac{r_{min}}{1-d}$$

For r to exist, we must have

$$\frac{r_{max}}{r_{min}} \leq \frac{1+d}{1-d}$$

Therefore, if $\frac{r_{max}}{r_{min}} \leq \frac{1+d}{1-d}$, all of the transactions in S support pattern P, where $\frac{r_{max}}{1+d} \leq r \leq \frac{r_{min}}{1-d}$.

For convenience, we also represent such patterns as $\{a_1, a_2, 1 : [\frac{r_{max}}{1+d}, \frac{r_{min}}{1-d}]\}$, where $[\frac{r_{max}}{1+d}, \frac{r_{min}}{1-d}]$ is called a *ratio range.*

For the rest of the paper, we also denote the ratio among attributes $\{a_1, a_2, \cdots, a_m\}$ as $\{1 : [r_{2_{min}}, r_{2_{max}}] : \cdots : [r_{m_{min}}, r_{m_{max}}]\}$.

Then we consider the value r satisfying that at least $N = s \cdot |W|$ transactions support pattern P. We sort the values of $v(a_2)/v(a_1)$ in S in increasing order of array RV. According to the above analysis, in order to have at least N transactions to support P, we check $RV[i+N-1]/RV[i]$ against $\frac{1+d}{1-d}$ for each i. If $RV[i+N-1]/RV[i] \leq \frac{1+d}{1-d}$, the corresponding transactions support the pattern $\{a_1, a_2, 1 : [RV[i+N-1]/(1+d), RV[i]/(1-d)]\}$.

3.4.2 Property for Candidate Pruning

Similar to the association rules mining in [?], we also utilize the downward closure property to prune the search space.

Definition 5 (*Sub Ratio Patterns*) Given two patterns $P = \{a_1, a_2, \cdots, a_n, 1 : r_2 : \cdots : r_n\}$, $Q = \{\acute{a}_1, \acute{a}_2, \cdots, \acute{a}_m, 1 : \acute{r}_2 : \cdots : \acute{r}_m\}$, we say Q is a sub-pattern of P, if

- $\{\acute{a}_1, \acute{a}_2, \cdots, \acute{a}_m\} \subseteq \{a_1, a_2, \cdots, a_n\}$; and
- For any pair of i and j, $v(\acute{a}_i)/v(\acute{a}_j)$ is in the ratio tolerance bound d of $v(a_i)/v(a_j)$ in P.

For example, given $P = \{a_1, a_2, a_3, 1 : 2 : 3\}$, $Q = \{a_2, a_3, 1 : 1.3\}$, Q is P's sub-pattern *w.r.t* $d = 0.2$.

Property 2 (*Downward Closure Property*) If pattern Q is one of P's sub-patterns, then $support(P) \leq support(Q)$.

Therefore, if the ratio of attributes in P is a frequent ratio pattern, the ratio s of attributes in all P's sub-patterns must be frequent as well. This property could be utilized to prune candidate patterns if some of their sub-patterns are not frequent.

3.4.3 Candidate Enumeration

Suppose L_k is the set of all ratio patterns of size k. Given L_k, we generate L_{k+1} with the following steps. For every pair of patterns $P_1 = \{a_1, \cdots, a_{k-1}, a_k, 1 : \cdots, [r_{k_{min}}, r_{k_{max}}]\}$ and

Table 3.3: Real datasets description.

Dataset	Transactions	Quantitative Attributes
image	2310	19
spambase	4601	57
covtype	581012	10

$P_2 = \{a_1, \cdots, a_{k-1}, a_{k+1}, 1 : \cdots, [\acute{r}_{(k+1)_{min}}, \acute{r}_{(k+1)_{max}}]\}$ in L_k, if their first $(k-1)$ attributes are the same, we consider a size of $(k+1)$ candidate pattern of the form $P_1 = \{a_1, \cdots, a_{k-1}, a_k, a_{k+1}, 1 : \cdots : r_{k-1} : r_k : r_{k+1}\}$, where

$$\begin{aligned}
r_2 &= [r_{2_{min}}, r_{2_{max}}] \cap [\acute{r}_{2_{min}}, \acute{r}_{2_{max}}]; \\
r_3 &= [r_{3_{min}}, r_{3_{max}}] \cap [\acute{r}_{3_{min}}, \acute{r}_{3_{max}}]; \\
\cdots \\
r_{k-1} &= [r_{(k-1)_{min}}, r_{(k-1)_{max}}] \cap [\acute{r}_{(k-1)_{min}}, \acute{r}_{(k-1)_{max}}]; \\
r_k &= [r_{k_{min}}, r_{k_{max}}] \cap [\acute{r}_{k_{min}}, \acute{r}_{k_{max}}]; \\
r_{k+1} &= [r_{(k+1)_{min}}, r_{(k+1)_{max}}] \cap [\acute{r}_{(k+1)_{min}}, \acute{r}_{(k+1)_{max}}].
\end{aligned}$$

If any of the above ratio ranges $\{r_2, \cdots, r_k, r_{k+1}\}$ is empty, P is pruned directly. Otherwise, we intersect the transaction tids associated with P_1 with those of P_2. If the number of transactions containing both P_1 and P_2 is less than s, P is deleted as well.

Finally, we match the results of candidates of correlated attribute sets with the results of attributes whose ratios are frequent. If a set of attributes occurs in both of the two processes, then the pattern consisting of the set of attributes and the frequent ratios among the attributes is output as a quantifiable correlated pattern.

3.5 Experimental Results

In Sections 6.1 and 6.2, experimental results on synthetic data show that our pruning methods for two subtasks prune the unwanted patterns efficiently in one scan of the transactions in a sliding window, respectively. In Section 6.3, we show that mining ratio relationships among correlated attributes is more effective than mining just frequent ratio rules using real data.

Synthetic Data We first generate a table of potential ratio values. The size of each ratio is computed as a Poisson distribution with mean of 6. Attributes and their ratios in a pattern are selected randomly. In this step, 2500 ratio patterns are generated from 1000 attributes. Next, the size of transactions is determined by a Poisson distribution whose mean is 6. Then we randomly choose ratios to generate transaction data.

Real data We use three real datasets from UCI machine learning repository [1] as listed in Table **??**. For each dataset, we choose quantitative attributes for test.

3.5.1 Efficiency of Generating Correlated Sets

We set the false-negative tolerance $\tau = 0.005$, $k = 4$ and $t = \log_{1-\delta^k} \tau$. We compare performances of our method with that of NMI-based method [**?**].

In Figure **??** and Figure **??**, we plot the running time for a sliding window and for the whole dynamic data streams, respectively. Our method achieves less overall running time, due to the very small candidate set generated by our method. In addition, in Figure **??**, we can see that in the favor of incremental pruning process of our method at each new arrival transaction, the running time for the whole dynamic data streams is significantly less than NMI-based method.

In Figure **??**, we measure how small the candidate set is, compared to the possible sets. Then we can see that candidate set generated by our method is often one order of magnitude smaller than NMI-based method. Figure **??** plots running time of our method against the number of attributes in the data streams to examine scalability. It shows that running time increases linearly with the number of attributes.

3.5.2 Efficiency of Mining Frequent Ratios

In Figure **??**, Figure **??**, Figure **??** and Figure **??**, we plot the running time for a sliding window and for the whole dynamic data streams, respectively, against different s and different d. For both of the two types of running time, we see that with a higher s, running time decreases. As d increases, a pattern is likely to be supported by more transactions. Hence more ratio patterns are mined, and the increase in the running time is expected.

[1]http://archive.ics.uci.edu/ml/

From Figure **??** and Figure **??**, we observe that with increase of s, the pruned ratio increases, and with increase of d, the pruned ratio decreases. In Figure **??** it can be seen that the running time scales linearly with the number of attributes.

3.5.3 Quantifiable Correlated Patterns vs. Frequent Ratio Rules

To further justify the feasibility of our methods, we compare the resultant ratio patterns with frequent ratio rules (FRR). We implement the algorithm proposed in [**?**] to mine FRRs in this experiment. We test five settings of s: $s = 0.1\%, 1\%, 10\%, 20\%, 30\%$. Figure **??** presents the cumulative probability distribution of correlation over the attributes in FRRs from *image* data set. When $s \leq 10\%$, around 90% of FRRs have low correlations by less than 10%. When $s = 20\%$, 60% of the FRRs have correlations limited to 20% only. Although half of the patterns have correlations greater than 90% when $s = 30\%$, these FRRs are mostly composed of attributes with trivial correlations, which are unlikely to be considered as useful knowledge.

On the contrary, the support distribution of ratio patterns in Figure **??** shows that most of the ratio patterns do not have high support. In fact, many of the ratio patterns are rare and not easy to be discovered, but they are significant, as the attributes in these patterns are highly correlated. Mining such patterns using FRR requires a small s, while returning a large number of uncorrelated patterns at the same time.

We do not present the results of FRR for *spambase* and *covtype* data, because FRR runs out of memory for all values of s. The massive number of generated FRRs not only results in high memory consumption, but also gives rise to difficulties in further analysis of the patterns.

3.5.4 Experiments on US Census Data

We also carry out a case study on the US census data for the year 2000 [2]. Each record of the census contains the population information of a city or a state. The information includes social characteristics such as education level; economic characters such as occupation; and housing characteristics such as the number of vehicles owned. We downloaded the records for three states; *Alabama* (588 records), *California* (1323 records) and *New York* (2101 records). Quantifiable correlated patterns

[2]htt://www.census.gov/main/www/cen2000.html

are mined from the three states. We discuss some interesting patterns discovered.

Figure **??** shows the quantifiable correlation between private wage and salary workers and the number of people with some college education in Alabama (AL) and New York (NY). We see that the ratio of NY is larger than that of AL. The implication is that more people in New York are private wage and salary workers (other categorizes are government workers, self-employed and unpaid family workers).

Figure **??** presents the quantifiable correlation between the number of houses with 2 vehicles owned and the number of people with some college education in Alabama and California. The patterns show that more families in Alabama have 2 vehicles in contrast to vehicle ownership in California.

In summary, quantifiable correlated patterns could successfully distinguish the moderate census differences among states.

3.6 Summary

In this chapter, we consider the problem of correlation mining in streaming quantitative transaction data. We propose a novel quantifiable correlated pattern for describing frequent ratio relationship among highly correlated attributes. In order to address two combinatorial explosion problems of naive method, we propose efficient and effective methods for generating correlated sets and mining frequent ratios. Our experiment results show that our pruning methods yield a candidate set much smaller than that of related work. Additionally, we show that mining ratio relationship among correlated attributes is much more interesting than mining frequent ratio rules.

Algorithm 1 Generating Highly Correlated Attribute Sets

We use $k \cdot t$ hash functions $h_0, h_1, \cdots h_{k \cdot t-1}$ and a hash table *HT*, *HT* uses a hash function $\hat{h}$ that maps a vector of k integers (the key) to one of its buckets.

C: The sets of highly correlated attribute

H: H[*i*] is a vector that stores the $k \cdot t$min-hash values of attribute *i*.

```
Compute Min-Hash:
  for each attribute $i$ and $u$ from 0 to $k \cdot t - 1$ do
    H[$i$][$u$] ← m
  end for
  In one scan of transactions in a sliding window:
  for each attribute $i$ in transaction $j$ do
    for $u$ from 0 to $k \cdot t - 1$ do
      if H[$i$][$u$] > $h_u(j)$ then
        H[$i$][$u$] ← $h_u(j)$
      end if
    end for
  end for

Generate Candidate Sets:
  for $i$ from 0 to $t$-1 do
    set all buckets of HT to $\phi$
    for each attribute $j$ do
      $v$ ← H[$j$][$i*k : (i+1)*k - 1$]
      HT[ $\hat{h}(v)$ ] ← HT[ $\hat{h}(v)$ ] U $j$
    end for
    for all bucket HT[$u$] do
      if HT[$u$] has more than one attribute then
        for every set of attributes $p$ in HT[$u$] do
          $C \leftarrow C$ U $p$
        end for
      end if
    end for
  end for

Incremental K-Min-Hash Process at New arrival Transaction $j$:
  for $i$ from 0 to $k \cdot t - 1$ do
    calculate $h_0(j), h_1(j), \cdots h_{k \cdot t-1}(j)$
    for each attribute $u$ do
      compare $h_0(j), h_1(j), \cdots h_{k \cdot t-1}(j)$ with $h_{0\min}(u), h_{1\min}(u), \cdots h_{k \cdot t-1\min}(u)$
      if $h_r(j) < h_{r\min}(u)$ then
        $h_{r\min}(u) \leftarrow h_r(j)$
      end if
    end for
  end for
```

Figure 3.3: Algorithm for generating highly correlated attributes

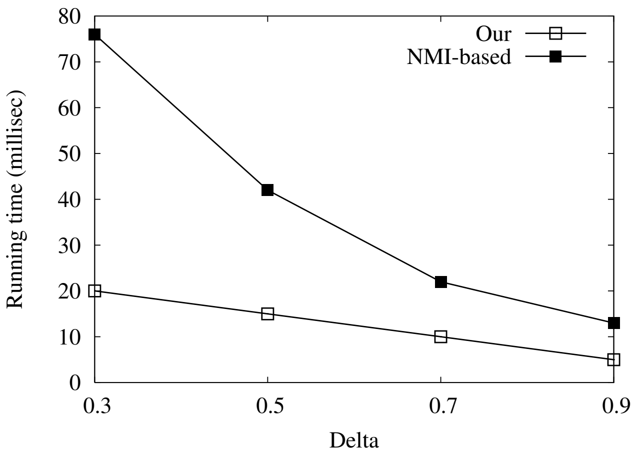


Figure 3.4: A sliding window

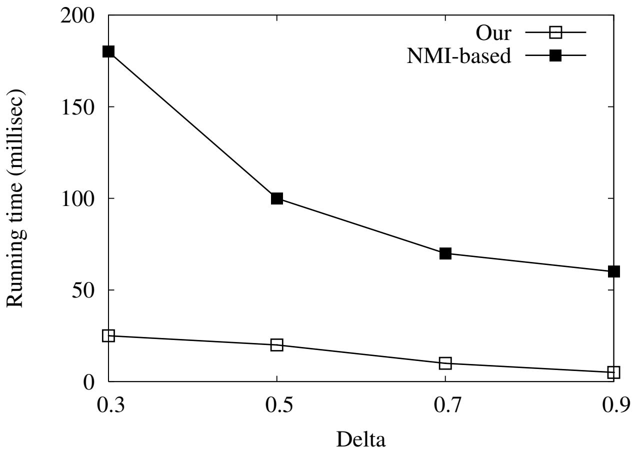


Figure 3.5: Whole dynamic data streams

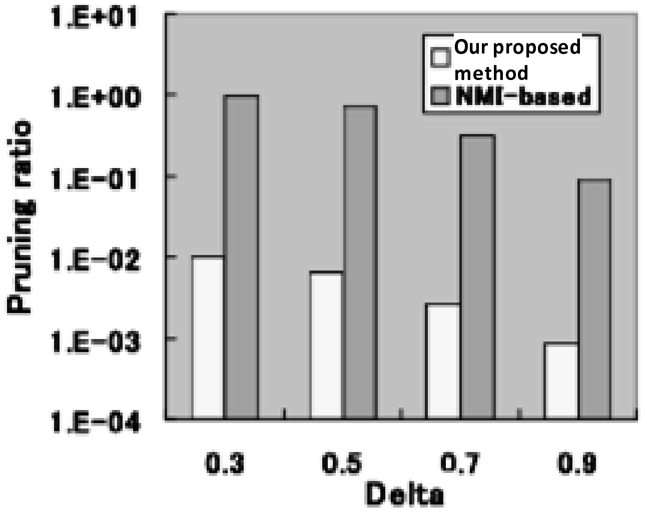


Figure 3.6: Pruning power comparison

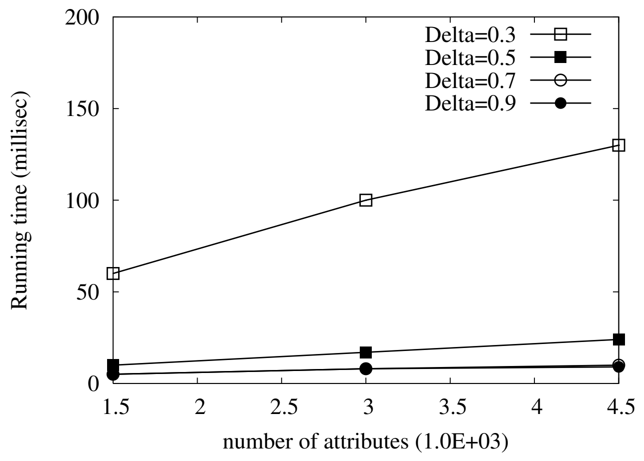


Figure 3.7: Scalability for generating correlated sets

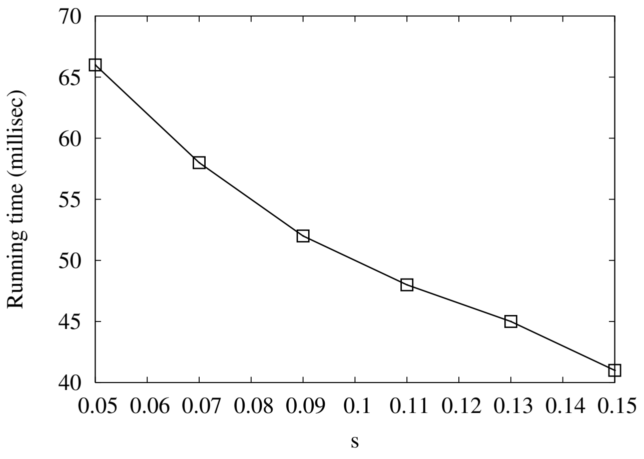


Figure 3.8: A sliding window vs. support s

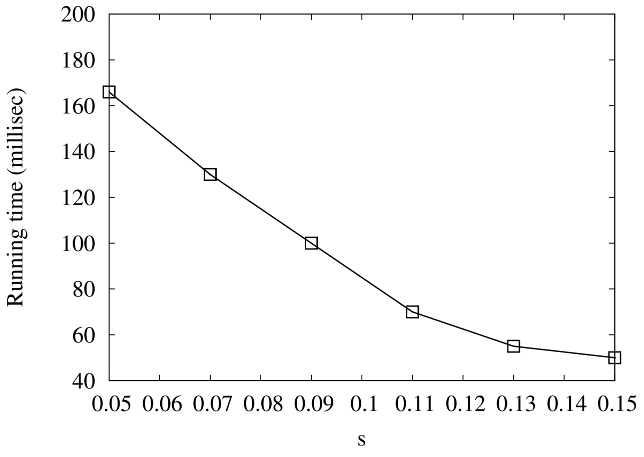


Figure 3.9: Whole dynamic data vs. support s

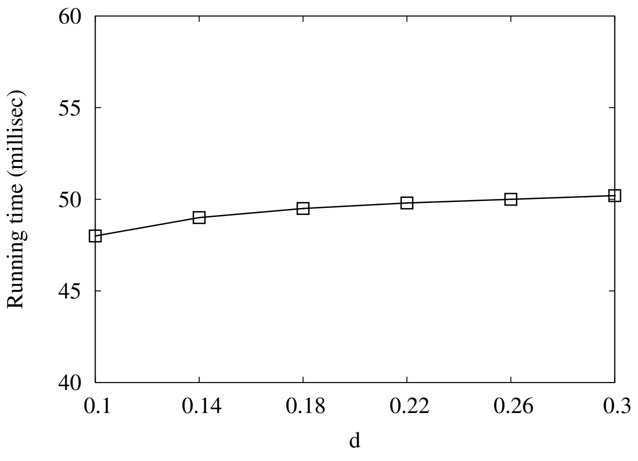


Figure 3.10: A sliding window vs. ratio tolerance d

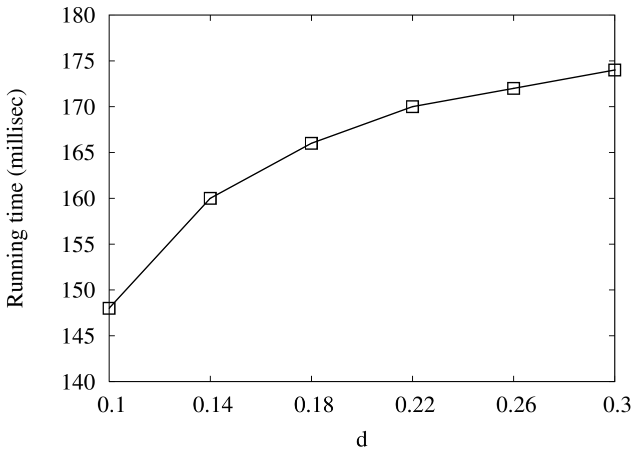


Figure 3.11: Whole dynamic data vs. ratio tolerance d

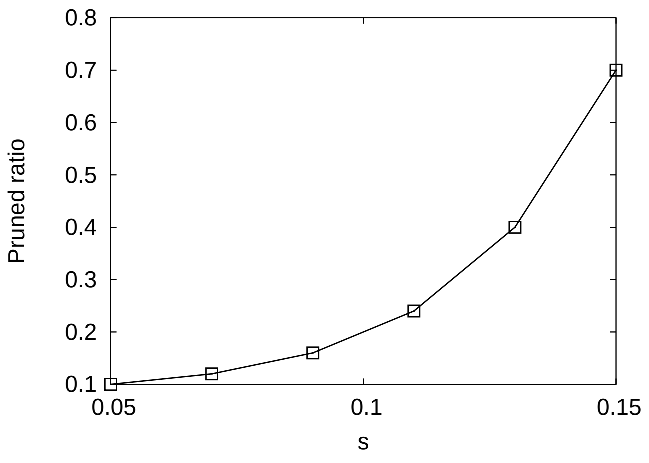


Figure 3.12: Pruned ratio vs. support s

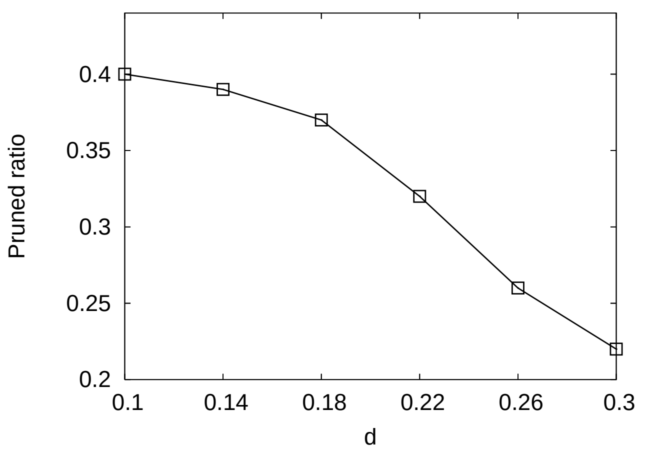


Figure 3.13: Pruned ratio vs. ratio tolerance d

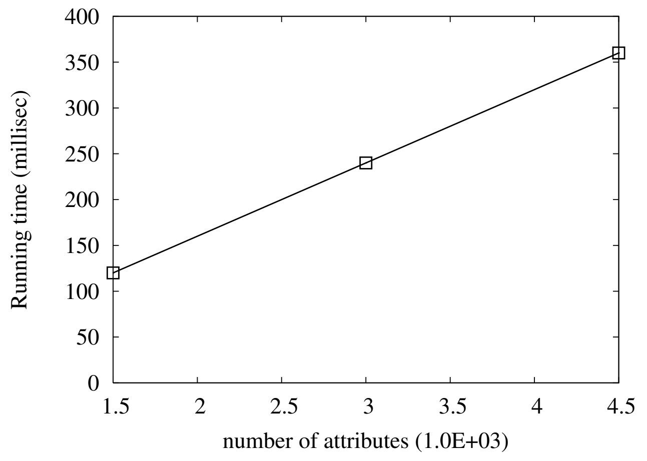


Figure 3.14: Scalability of mining frequent ratios

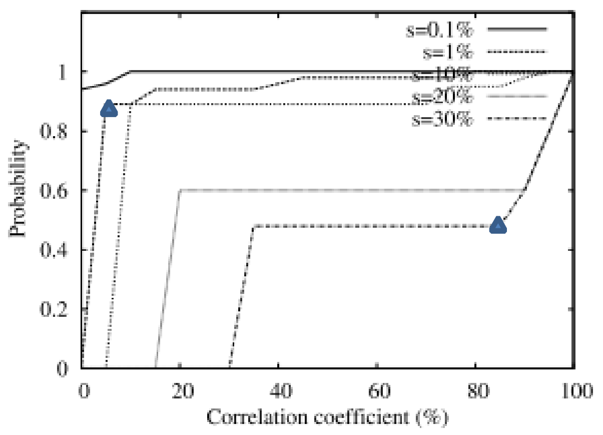


Figure 3.15: Cumulative probability distribution of correlation coefficient

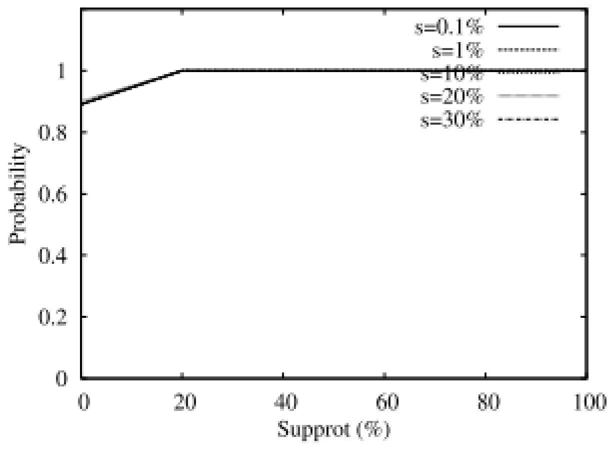


Figure 3.16: Cumulative probability distribution of support

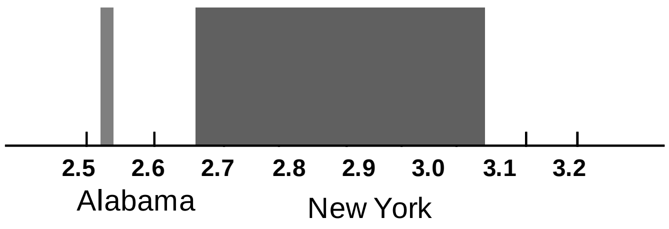


Figure 3.17: Quantifiable correlation between number of private wage and salary workers and number of people with some college eduction

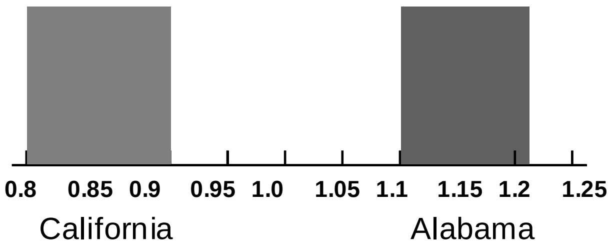


Figure 3.18: Quantifiable correlation between number of houses owning 2 vehicles and number of people with some college eduction

Chapter 4

Underlying Correlated-clusters Mining

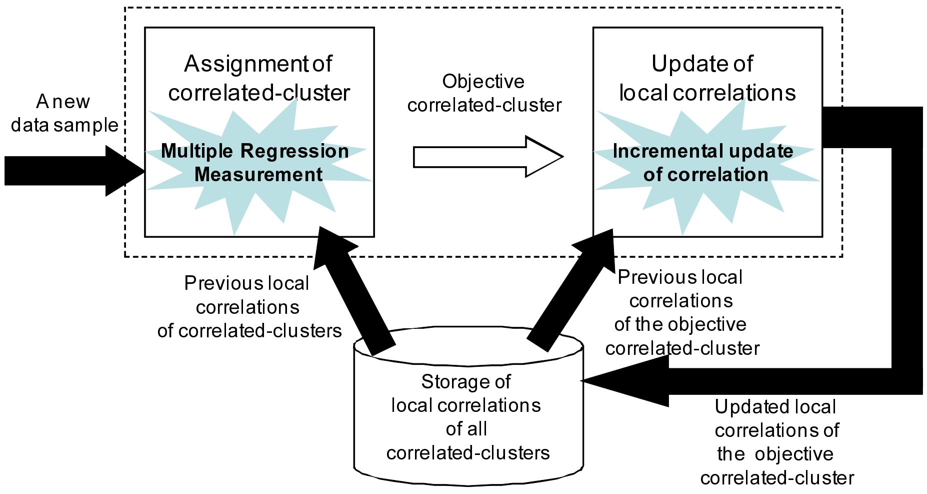
High dimensional data streams pose challenges to traditional clustering algorithm, due to their inherent sparsity. In this chapter, we resolve the problem of mining *hidden variables* which summarize the key trends of massive evolving streaming time-series data. Moreover, taking data evolution in data streams into account, we propose methods to mine correlations incrementally and adaptively. At each time tick t, according to our proposed multiple regression measure, we cluster the newly arrived data sample to one of correlated-clusters whose local correlations fit to the data sample, and also update the local correlations adaptively, based on the incremental Principal Component Analysis technology.

4.1 Introduction

We describe methods to mine correlated-clusters from high dimensional data streams. Our methods automate the change detection and maintenance of underlying correlations in streaming data. Notable aspects include:

- A generalized multivariate regression measure (GR^2 measure) detects the change of correlations. At each time tick t, according to the GR^2 measure we cluster the newly arrived data sample to one of the correlated-clusters whose local correlations fit to the data sample. The data sample that does not belong to any cluster is output as the first data sample of a new correlated-cluster. The GR^2 measure allows the user to control the amount of information loss.

- A new incremental Principal Component Analysis (IPCA) method updates local correlations



A new data sample
Assignment of correlated-cluster
Multiple Regression Measurement
Objective correlated-cluster
Update of local correlations
Incremental update of correlation
Previous local correlations of correlated-clusters
Previous local correlations of the objective correlated-cluster
Storage of local correlations of all correlated-clusters
Updated local correlations of the objective correlated-cluster

Figure 4.1: Overview of the proposed incremental process

of the objective correlated-cluster incrementally and adaptively. This process is independent of the length of data streams. It requires very limited memory and short processing time per time tick.

- The results of experiments on high dimensional synthetic data and real data demonstrate that our methods can achieve higher accuracy of query than other related works, and perform much more efficiently. Additionally, our proposed methods are able to forecast missing values in streaming data successfully.

Figure **??** illustrates the process of our incremental and adaptive mining of correlated-clusters at each newly arrived data sample. At time tick t, we cluster the data sample to one of the existing correlated-clusters whose local correlations fit to the data sample, according to our proposed GR^2 measure, and then we adaptively update local correlations of the objective correlated-cluster.

The rest of this chapter is organized as follows. In Section 4.2, we give the problem definition. In Section 4.3, we elaborate our proposed methods for clustering continuous data samples into correlated-clusters and updating local correlations adaptively. In Section 4.4, we present the performance results on synthetic and real data. Section 4.5 offers the final concluding remarks.

4.2 Problem Definition

Given $n(n \geq 2)$ data streams $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, ..., $\boldsymbol{X}_n$ as

$$\begin{pmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \vdots \\ \boldsymbol{X}_n \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1t} & \dots \\ x_{21} & x_{22} & \dots & x_{2t} & \dots \\ \vdots & \vdots & \vdots & \vdots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nt} & \dots \end{pmatrix},$$

we reorganize data samples in the n-dimensional space as:

$$\boldsymbol{p}_1 = \begin{pmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{n1} \end{pmatrix}, \boldsymbol{p}_2 = \begin{pmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{n2} \end{pmatrix}, \dots, \boldsymbol{p}_t = \begin{pmatrix} x_{1t} \\ x_{2t} \\ \vdots \\ x_{nt} \end{pmatrix}, \dots$$

In our processes of streaming data, it is required to process data sample $\boldsymbol{p}_t$ incrementally without rescanning and recomputing all the previous data samples.

We aim to cluster data samples from data streams into meaningful groups, referred to correlated-clusters. Based on the local correlations within each correlated-cluster, it is possible to reduce high-dimensional data samples into just a handful of hidden variables that compactly describe the key trends and dynamically reduce the complexity of further data analysis.

4.2.1 Identification of a Correlated-cluster

Definition 1 (*Correlated-cluster*) A correlated-cluster c is a set of locally correlated n-dimensional data samples $\boldsymbol{D}_c$. We record the vector $(d_c, \boldsymbol{\Phi}_c, \boldsymbol{m}_c, \boldsymbol{O}_c)$ to identify the correlated-cluster c, where

- d_c is the number of local correlations (number of Principal Components (PCs));
- $\boldsymbol{\Phi}_c$ is the set of local correlations, and $\boldsymbol{\Phi}_c^{(i)}$ denotes the i-th PC $(1 \leq i \leq d_c)$;
- $\boldsymbol{m}_c$ is n-dimensional mean point of data samples in correlated-cluster c;
- $\boldsymbol{O}_c = [O_c^{(d_c+1)} \cdots O_c^{(n)}]$ is centroid, which represents the position of the mean point $\boldsymbol{m}_c$ of c along eliminated $(n - d_c)$ dimensions;

For example, in an original 2-dimensional feature space, c is a correlated-cluster as shown in Figure **??**, where the dimensionality of data samples is reduced to 1, that $d_c = 1$: therefore, only

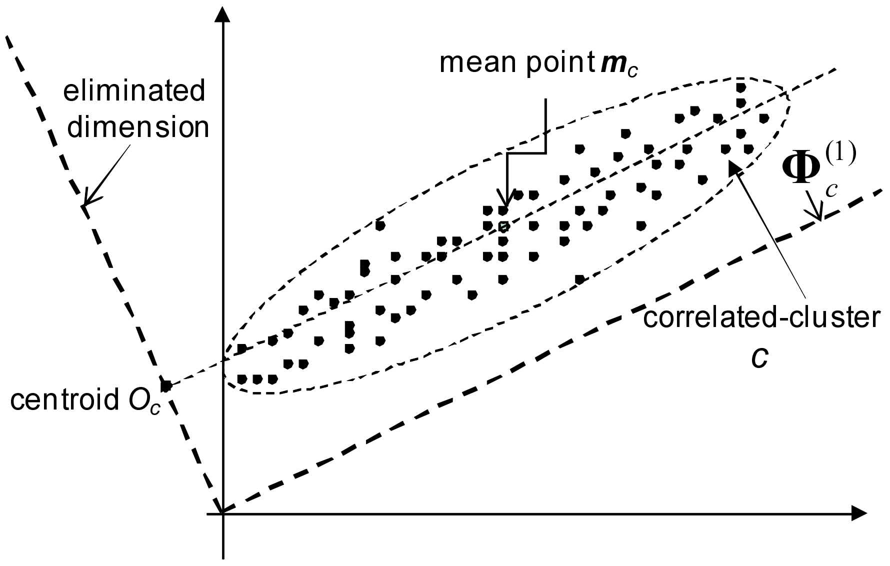


Figure 4.2: An example of correlated-cluster

the local correlation $\boldsymbol{\Phi}_c^{(1)}$ is used to preserve the variant of the data samples. The point $\boldsymbol{m}_c$ is the mean point of c, and correspondently, O_c is the centroid, by projecting the mean point onto eliminated dimensions.

Next, we discuss how to represent local correlations. We assume the local correlation $\boldsymbol{l}$ existing in the data set $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_t$ as a linear subspace of the entire dimensions space is expressed as:

$$\boldsymbol{l} = \boldsymbol{m} + \alpha \boldsymbol{e},$$

where $\boldsymbol{m}$ is a point in n-dimensional space, α is an arbitrary scalar, and $\boldsymbol{e}$ is a unit vector in the direction of $\boldsymbol{l}$. When we project $\boldsymbol{p}_i$ to $\boldsymbol{l}$, and we have point $\boldsymbol{m} + \alpha_i \boldsymbol{e}$ corresponding to $\boldsymbol{p}_i$, then the squared error of $\boldsymbol{p}_i$ is:

$$\boldsymbol{u}_i = (\boldsymbol{m} + \alpha_i \boldsymbol{e}) - \boldsymbol{p}_i.$$

Thus, the sum of all the squared-errors is to be:

$$\begin{aligned}
\sum_{i=1}^{t} \parallel \boldsymbol{u}_i \parallel^2 &= \sum_{i=1}^{t} \parallel (\boldsymbol{m} + \alpha_i \boldsymbol{e}) - \boldsymbol{p}_i \parallel^2 \\
&= \sum_{i=1}^{t} \parallel \alpha_i \boldsymbol{e} - (\boldsymbol{p}_i - \boldsymbol{m}) \parallel^2 \\
&= \sum_{i=1}^{t} \alpha_i^2 \parallel \boldsymbol{e} \parallel^2 - 2\sum_{i=1}^{t} \alpha_i \boldsymbol{e}^T (\boldsymbol{p}_i - \boldsymbol{m}) + \sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2 \\
&= \sum_{i=1}^{t} \alpha_i^2 - 2\sum_{i=1}^{t} \alpha_i \boldsymbol{e}^T (\boldsymbol{p}_i - \boldsymbol{m}) + \sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2 .
\end{aligned}$$

The sum of squared errors must be minimized. Note that $\sum_{i=1}^{t} \parallel \boldsymbol{u}_i \parallel^2$ is a function of $\boldsymbol{m}, \alpha_i$ and $\boldsymbol{e}$. Partially differentiating it with respect to α_i and setting the derivative to be zero, we can obtain:

$$\alpha_i = \boldsymbol{e}^T (\boldsymbol{p}_i - \boldsymbol{m}). \tag{4.1}$$

Now, we should determine a vector $\boldsymbol{e}$ to minimize $\sum_{i=1}^{t} \parallel \boldsymbol{u}_i \parallel$. Substituting (**??**) to it, we have:

$$\begin{aligned}
\sum_{i=1}^{t} \parallel \boldsymbol{u}_i \parallel^2 &= \sum_{i=1}^{t} \alpha_i^2 - 2\sum_{i=1}^{t} \alpha_i \alpha_i + \sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2 \\
&= -\sum_{i=1}^{t} \alpha_i^2 + \sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2 \\
&= -\sum_{i=1}^{t} [\boldsymbol{e}^T (\boldsymbol{p}_i - \boldsymbol{m})]^2 + \sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2 \\
&= -\sum_{i=1}^{t} [\boldsymbol{e}^T (\boldsymbol{p}_i - \boldsymbol{m})(\boldsymbol{p}_i - \boldsymbol{m})^T \boldsymbol{e}] + \sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2 \\
&= -\boldsymbol{e}^T \boldsymbol{S} \boldsymbol{e} + \sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2, \qquad (4.2)
\end{aligned}$$

where $\boldsymbol{S} = \sum_{i=1}^{t} (\boldsymbol{p}_i - \boldsymbol{m})(\boldsymbol{p}_i - \boldsymbol{m})^T$, called *scatter matrix.*

Obviously, the vector $\boldsymbol{e}$ that minimizes above equation also maximizes $\boldsymbol{e}^T \boldsymbol{S} \boldsymbol{e}$. We can see Lagrange multipliers to maximize $\boldsymbol{e}^T \boldsymbol{S} \boldsymbol{e}$ subject to the constraint $\parallel \boldsymbol{e} \parallel = 1$. Let:

$$\mu = \boldsymbol{e}^T \boldsymbol{S} \boldsymbol{e} - \lambda(\boldsymbol{e}^T \boldsymbol{e} - 1).$$

Differentiating μ with respect to $\boldsymbol{e}$, we have:

$$\frac{\partial \mu}{\partial \boldsymbol{e}} = 2\boldsymbol{S}\boldsymbol{e} - 2\lambda \boldsymbol{e}.$$

Therefore, in order to maximize $\boldsymbol{e}^T\boldsymbol{S}\boldsymbol{e}$, $\boldsymbol{e}$ must be the eigenvector of the scatter matrix S:

$$\boldsymbol{S}\boldsymbol{e} = \lambda \boldsymbol{e}. \tag{4.3}$$

Finally, we need $\boldsymbol{m}$ to complete the solution. $\sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel$ should be minimized since it is always non-negative. To minimize it, $\boldsymbol{m}$ must be the average of $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_t$. With $\boldsymbol{m}$ as the average data samples and $\boldsymbol{e}$ from (**??**), the local correlation $\boldsymbol{l}$ is determined: we can define local correlations in correlated-cluster in a simple form as follows:

Definition 2 (*Local Correlations*) Assume that there are d_c local correlations in a correlated-cluster c, then we can represent each local correlation at time tick t as follows:

$$\frac{\boldsymbol{p}_t(1) - \bar{\mathrm{X}}_1}{{e_1}^{(i)}} = \frac{\boldsymbol{p}_t(2) - \bar{\mathrm{X}}_2}{{e_2}^{(i)}} = \ldots = \frac{\boldsymbol{p}_t(n) - \bar{\mathrm{X}}_n}{{e_n}^{(i)}}, \tag{4.4}$$

where $\boldsymbol{p}_t(j)$ is the value of j-th dimensional element of data sample $\boldsymbol{p}_t$ $(1 \leq j \leq n)$, $[{e_1}^{(i)}, {e_2}^{(i)}, \ldots, {e_n}^{(i)}]^T$ is the eigen-vector corresponding to i-th largest eigen-value of the scatter matrix $\boldsymbol{S}$ $(1 \leq i \leq d_c)$. $\bar{\boldsymbol{X}}_j (j = 1, 2, \ldots, n)$ is the average of sequence $\boldsymbol{X}_j$.

In the next section, we elaborate our proposed methods for mining local correlations incrementally and adaptively in the collection of high dimensional streaming data.

4.3 Proposed Correlated-clusters Mining Process

We resolve the problem of mining correlated-clusters by two-phases process. As illustrated in Figure **??**, for each newly arrived data sample, we run the phases for assigning correlated-cluster and updating local correlations.

4.3.1 Assignment of Correlated-cluster

In the last section, we discussed about utilizing a linear subspace of the entire feature space to represent local correlations, and in this subsection we answer the question: how do we measure fitness of linear local correlations to the newly arrived data sample? We propose a multiple regression measure to evaluate the goodness-of-fit of a local correlation to the new data sample.

We cluster the data sample to the correlated-cluster whose local correlations fit to the data sample best.

In the following equation (**??**), the R^2 measure is used as a measure for the goodness-of-fit in 2-dimensional linear regression models. In equation (**??**), u_i is reconstruction error by using this regression model to predict a sequence y_i. The value of R^2 measure is always between 0 and 1. The closer the value is to 1, the better the regression line fits to the data samples.

$$R^2 = 1 - \frac{\sum_{i=1}^{t} u_i^2}{\sum_{j=1}^{n} \sum_{i=1}^{t} (y_i - \bar{y})^2} \tag{4.5}$$

Similarly, we can extend the measurement for multiple dimensional data, referred to GR^2 measure. It is expressed as:

$$GR^2 = 1 - \frac{\sum_{i=1}^{t} \parallel \boldsymbol{u}_i \parallel^2}{\sum_{j=1}^{n} \sum_{i=1}^{t} (x_{ji} - \bar{\boldsymbol{X}}_j)^2}, \tag{4.6}$$

where $\boldsymbol{u}_i$ is reconstruction distance defined as vertical distance of data sample $\boldsymbol{p}_i$ to a local correlation.

For example, in the 2-dimensional space as shown in Figure **??**, c is an existing correlated-cluster, and its local correlation is $\boldsymbol{\Phi}_c^{(1)}$. When a new data sample $\boldsymbol{p}_t$ arrived, then $u_{\boldsymbol{p}_t}$ is the reconstruction distance of $\boldsymbol{p}_t$ to $\boldsymbol{\Phi}_c^{(1)}$, supposing that $\boldsymbol{p}_t$ is partitioned into c.

From equation (**??**) we can further derive:

$$\begin{aligned} GR^2 &= 1 - \frac{\sum_{i=1}^{t} \parallel \boldsymbol{u}_i \parallel^2}{\sum_{j=1}^{n} \sum_{i=1}^{t} (x_{ji} - \bar{\boldsymbol{X}}_j)^2} \\ &= 1 - \frac{\sum_{i=1}^{t} \parallel \boldsymbol{u}_i \parallel^2}{\sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2}. \end{aligned}$$

Substitute equation (**??**), then we get

$$\begin{aligned} GR^2 &= \frac{\boldsymbol{e}^T \boldsymbol{S} \boldsymbol{e}}{\sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2} \\ &= \frac{\boldsymbol{e}^T \lambda \boldsymbol{e}}{\sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2} \\ &= \frac{\lambda}{\sum_{i=1}^{t} \parallel \boldsymbol{p}_i - \boldsymbol{m} \parallel^2}. \end{aligned}$$

We can derive the following important properties of the GR^2 measure:

1. $0 \leq GR^2 \leq 1$.

2. $GR^2 = 1$ means the n data streams have exact linear correlation with each other.

3. GR^2 is invariant to the order of $\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_n$: i.e., we can arbitrarily change the order of the n data streams, while the value of GR^2 does not change.

Proof.

1. According to equation (**??**), we have:

$$\sum_{i=1}^{t} \| \boldsymbol{u}_i \|^2 = -\boldsymbol{e}^T \boldsymbol{S} \boldsymbol{e} + \sum_{i=1}^{t} \| \boldsymbol{p}_i - \boldsymbol{m} \|^2 \geq 0.$$

   Therefore

$$\sum_{i=1}^{t} \| \boldsymbol{p}_i - \boldsymbol{m} \|^2 \geq \boldsymbol{e}^T \boldsymbol{S} \boldsymbol{e} = \lambda \geq 0,$$

   so we conclude $0 \leq GR^2 = \frac{\lambda}{\sum_{i=1}^{t} \| \boldsymbol{p}_i - \boldsymbol{m} \|^2} \leq 1$.

2. If $GR^2 = 1$, then $\sum_{i=1}^{n} \| \boldsymbol{u}_i \|^2 = 0$, which means the local correlation fits to the t data samples perfectly. Therefore, the n sequences have exact linear correlation with each other.

3. According to expression (**??**), this is obvious.

According to equation (**??**), at each time tick t we can calculate the value of GR^2 measure incrementally to evaluate the goodness-of-fit of local correlations, if the vector $\boldsymbol{e}$ can be estimated incrementally. In the next subsection, we will introduce a method for estimation. We define a threshold, then the new arriving data sample is clustered into the correlated-cluster whose value of GR^2 measure is larger than the threshold. If the values of GR^2 measure in all existing correlated-clusters are less than the threshold, then we determine that this data sample is the first data sample of a new correlated-cluster.

After determining the objective correlated-cluster, in the following subsection we discuss how to update its local correlations incrementally and adaptively.

4.3.2 Update of Local Correlations

In this section, we discuss how to update local correlations of the objective correlated-cluster c. We propose a new IPCA algorithm based on Normalized Least mean Squares (NLMS) adaptive filter technique [?] as shown in Figure ??. Line 0 initializes i-th local correlation $\boldsymbol{e}_i (1 \leq i \leq d_c)$ to a unit vector. g_t denotes the energy of data samples when they are projected to the i-th local correlation. It is initialized to a small positive value. When a new data sample $\boldsymbol{p}_t$ arrives, for each i-th local correlation we do the following process:

- Compute the projection $\boldsymbol{y_i}$ by projecting $\boldsymbol{p}_t$ onto $\boldsymbol{e_i}$ in Line 4;
- Estimate energy g_t and reconstruction error $\boldsymbol{u}_t$ based on $\boldsymbol{y_i}$ in Line 5 and Line 6;
- Update the estimation of $\boldsymbol{e_i}$ in Line 7. The larger the reconstruction error $\boldsymbol{u}_t$ is, the more $\boldsymbol{e_i}$ is updated. However, the magnitude of this update should also take into account the past data currently "captured" by $\boldsymbol{e_i}$. Therefore, the update is inversely proportional to the current energy $\boldsymbol{g}$; and
- In order to update the remainder local correlations, we remove the projection of original data sample on $\boldsymbol{e_i}$ in Line 8 and Line 9. Then we repeat the above steps with remainder local correlations.

On the other hand, for the determination of the number of local correlations d_c in the objective correlated-cluster, we can adopt energy-based dynamic determination of d_c. That is to say, after the update of local correlations, we compare the ratio for energy of reconstructed data with that of original data: if the ratio is less than the user-defined minimum ratio of preservation, then the number of local correlations increases by 1, and we repeat the above IPCA algorithm to generate the new local correlation. Otherwise, if the ratio is larger than the user-defined maximum ratio of preservation, then the number of local correlations decreases by 1, and we delete the last local correlation.

4.4 Experiments

In this section we present experimental results to show efficiency of our proposed methods and effectiveness of discovered local correlations used for query in high dimensionality space, and for forecasting missing data. We compare our incremental process for mining local correlations with the incremental CCIPCA approach proposed in [?].

4.4.1 Experimental Methodology

Range query In our experiments, we compare accuracy of query in the two reduced dimensional data. We do range query in our experiments. For one of original n-dimensional data P, we aim to retrieve all objects Q in the dataset which satisfy the distance between the query anchors P and Q is less than user-defined threshold ρ ($D(Q, P) \leq \rho$). We use Euclidean distance as the distance metric. The accuracy of query is defined as $\frac{|R_{original}|}{|R_{reduced}|}$, where $R_{original}$ and $R_{reduced}$ are the sets of answers returned by query on original space and reduced-dimensional space respectively [**?**,**?**]. The accuracy measures the information-loss incurred by dimension reduction and hence the query cost. We compare the accuracy of these two approaches with fixed reduced dimensionality: therefore, the higher the accuracy is, the lower the cost of query is and also the more efficient the technique is.

Forecasting missing data The principal components give us a much more compact representation of the original data streams, with guarantees of high reconstruction accuracy. When our streams exhibit correlations, the number of principal components is much smaller than the original high dimensionality. Therefore, we can apply forecasting algorithm to the vector of principal components, instead of the original data samples. This reduces the time and space complexity by orders of magnitude, because typical forecasting methods are quadratic or worse on the number of dimensionality. In particular, for auto-regression we found that one auto-regression model per principal component provides results comparable to multivariate auto-regression. In our experiments, we use the forecast based on $\boldsymbol{X}_{t-1}$ to estimate missing values in $\boldsymbol{X}_t$. We then use these estimated missing values to update the principal component estimates, as well as the forecasting models.

4.4.2 Experiments on Synthetic Data

We use the same method to generate synthetic data as done in [**?**]. We generate different correlated-clusters with different orientations and dimensionality. Some important input parameters to the data generator are shown in Table **??**. The generator makes up k correlated-clusters with a total of $N \cdot (1 - o)$ data samples, and the average subspace dimensionality being d. Each correlated-cluster is generated as follows. For a correlated-cluster with size N_i and subspace dimensionality

Table 4.1: Input parameters of synthetic data generator.

Parameter	Description	Default Value
N	Total number of data samples	2000
n	Original dimensionality	64
k	Number of correlated clusters	5
d	Average subspace dimensionality	10
p	Maximum displacement of data samples	
	among each non-subspace dimension	0.2
o	Fraction outliers	0.05

d_i, we randomly choose d_i dimensions among the n dimensions as the subspace dimensions and generate N_i data samples in d_i-dimensional plane. Along each of the remaining $(n - d_i)$ non-subspace dimensions, we assign a randomly chosen coordinate to all the n_i data samples in the cluster. Each data sample is displayed by a distance of at most p in either direction along each non-subspace dimension. The value of p determined the degree of correlation. The lower the value is, the more closer the correlation is. After all the correlated-clusters are generated, we randomly generate $N \cdot o$ data samples as the outliers.

We generated 100 range queries randomly and chose the threshold ρ so that the average query selectivity is about 2%. All of our measurements are averaged over the 100 queries. We carry out a sensitivity analysis of CCIPCA and our proposed methods with parameters: the number of clusters k, degree of correlation p, and total number of data samples N. In each experiment, we vary the parameter of interest while the remaining parameters are fixed at their default values. We fix the reduced dimensionality of the batch technique to 10, which is the same as the average subspace dimensionality. Figure **??** compares the accuracy of the two approaches for various values of k. As expected, for one correlated-cluster, two approaches are identical. As k increases, the number of data samples in each correlated-cluster becomes fewer. The accuracy of our proposed incremental approach deteriorates, but it remains similar to that of CCIPCA. Nevertheless, execution time of CCIPCA increases as illustrated in Figure **??**, because it performs more iterations of re-clustering. Therefore, our incremental approach performs much more efficiently than CCIPCA. Figure **??** compares the two approaches for various values of p. As the degree of correlation decreases (*i.e.*, the value of p increases), the accuracy of both techniques drops but the result of our incremental approach is still similar to that of CCIPCA. In Figure **??**, we can see that since the total number of data samples increase, our methods perform much more efficient than CCIPCA, because it requires very short processing time which is independent of the length of data stream. Therefore, we can get the conclusion that our incremental approach can achieve similar query accuracy of CCIPCA but much more efficiently.

4.4.3 Experiments on Real Data

4.4.3.1 Image data

In these experiments, we use sequences of images obtained from Columbia Object Image Library [1] as shown in Figure **??**. In this image dataset, there are 20 objects rotated in their vertical axis, generating 72 images per object. The images are scaled to 80×80 pixels. We evaluate the accuracy of query in sequences of images for all of the 20 objects. In each experiment, we learn 7000 images of one object to build a set of view-based local correlations among pixels, and then do 100 range queries randomly. Therefore, all of our measurements are averaged over the 2000 queries.

From Figure **??**, we can see that our proposed approach achieves higher accuracy than CCIPCA: therefore, our proposed method is superior. Our approach adapts the local correlations with respect to the view change, while CCIPCA corrupted quickly owing to the influence of intensity. In Figure **??**, as the number of reduced dimensionality increases, that is, we adopt more reduced dimensionality for representation of original image data, the batch approach also achieves high accuracy, however it is less effective on execution time and memory space basis.

From this result, we can see that our proposed approach is superior and is applicable to solve the problems in practice, such as modeling face, modeling background and so on.

4.4.3.2 Room temperatures data

The *Critter* dataset consists of 8 streams. Each stream comes from a small sensor that connects to the joystick port and measures temperature. The sensors were placed in 5 neighboring rooms. Each time tick represents the average temperature during one minute. Our proposed methods are able to deal successfully with missing values in streams. Figure **??** shows the results on the

[1]http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php

blanked version (1.5% of the total values in five blocks of consecutive time ticks, starting at a different position for each stream) of *Critter*. In particular, on sensor 2 (second row in Figure **??**), the correlations picked by the single hidden variable successfully capture the missing values in that region (consisting of 270 time ticks). On sensor 1, (first row in Figure **??**, 300 blanked values), the upward trend in the blanked regions also picked up by the correlations. Even though the trend is slightly miss-estimated as soon as the values are observed again, our proposed methods very quickly gets back to near-perfect tracking.

4.5 Summary

In this chapter, we proposed a method to mine underlying correlated-clusters as well as local correlations from data streams incrementally and adaptively. We discussed a generalized multiple regression measurement for clustering data samples, and an IPCA algorithm for updating local correlations. An extensive experimental study demonstrated that our methods can achieve higher accuracy of query than other related works, and performs much more efficiently. Additionally, our proposed methods are able to forecast missing values in streaming data successfully.

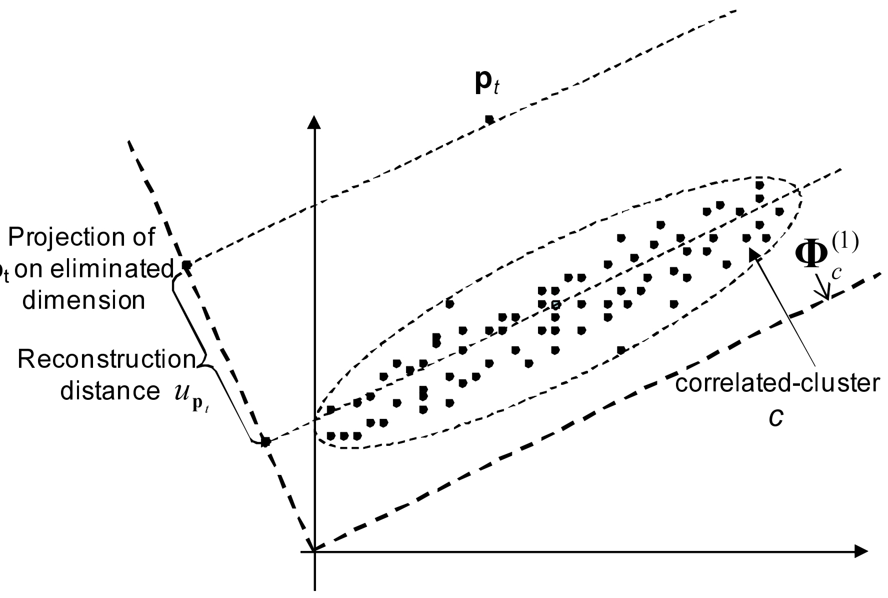

$\mathbf{p}_t$
Projection of $\mathbf{p}_t$ on eliminated dimension
Reconstruction distance $u_{\mathbf{p}_t}$
$\mathbf{\Phi}_c^{(1)}$
correlated-cluster c

Figure 4.3: Reconstruction error

```
Incremental Principal Component Analysis:
0. Initialize $\mathbf{e}_i$ to a unit vector, $g_i$ to small positive values $(i = 1, ..., d_c)$ .
1. for a newly arrived data sample $\mathbf{p}_t$
2.   $\hat{\mathbf{p}}_1 := \mathbf{p}_t$
3.   for 1< $i$ < $d_c$
4.       $y'_{t,i} := \mathbf{e}^T_{(t-1),i}\hat{\mathbf{p}}_i$                  // compute $i$- th PC
5.       $g_{t,i} := \delta g_{(t-1),i} + {y'_{t,i}}^2$             // energy of $i$ -th PC
6.       $\mathbf{u}_{t,i} := \hat{\mathbf{p}}_i - y'_{t,i}\mathbf{e}_i$                 // reconstruction error based on $i$ -th PC
7.       $\mathbf{e}_{t,i} := \mathbf{e}_{(t-1),i} + \frac{1}{g_{t,i}} y'_{t,i}\mathbf{u}_{t,i}$       // update coefficient of $i$ -th PC
8.       $\hat{\mathbf{p}}_{i+1} := \hat{\mathbf{p}}_i - y_{t,i}\mathbf{e}_{t,i}$              // output the actual $i$ - th PC
9.       $y_{t,i} := \mathbf{e}^T_{t,i}\hat{\mathbf{p}}_i$                     // repeat with remainder PCs
10.  Endfor
11. Endfor
```

Figure 4.4: Incremental update of local correlations

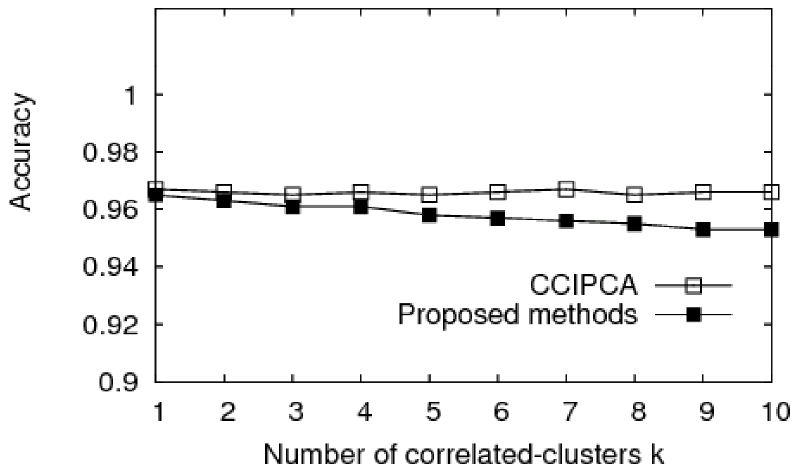


Figure 4.5: Accuracy vs. the number of hidden variables k

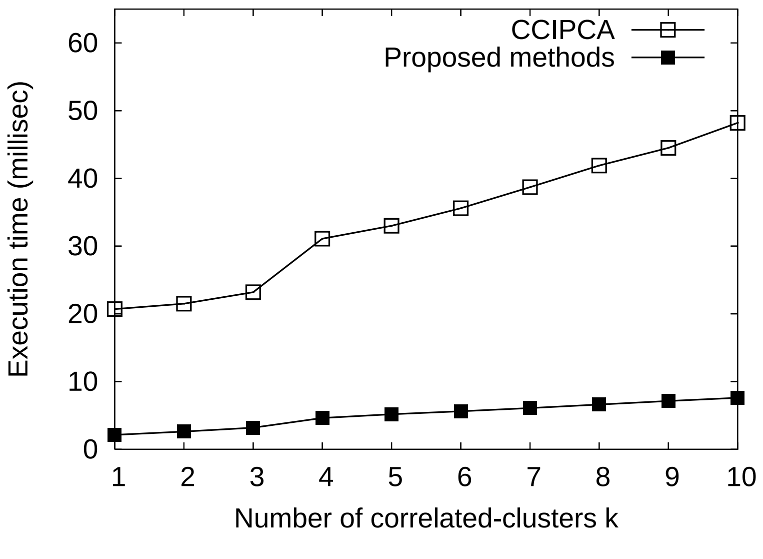


Figure 4.6: Execution time vs. the number of hidden variables k

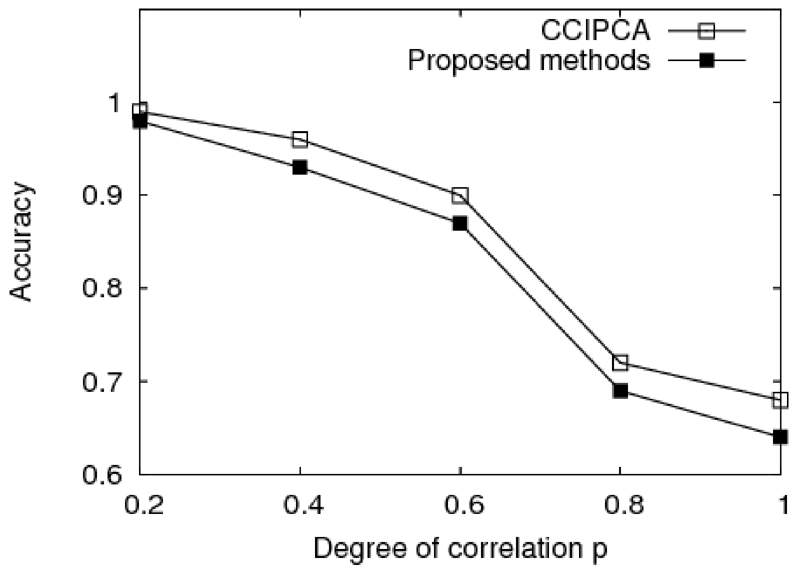


Figure 4.7: Accuracy vs. degree of correlation p

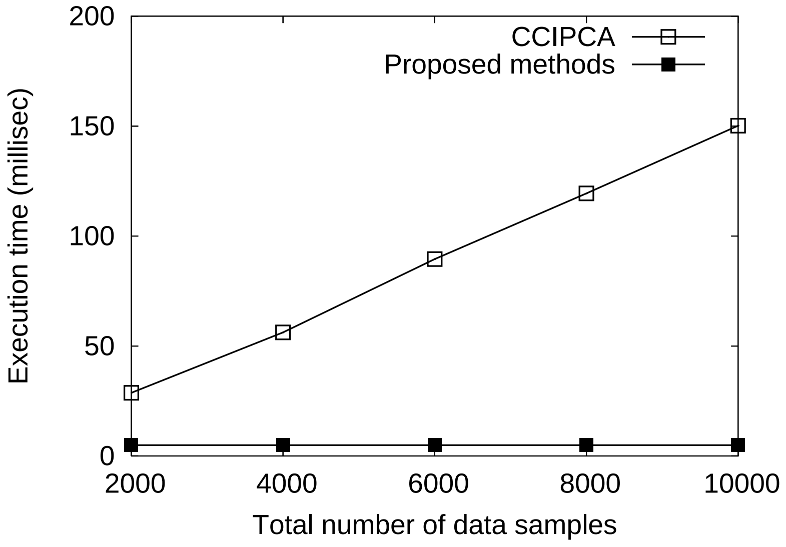


Figure 4.8: Execution time vs. total number of data samples N

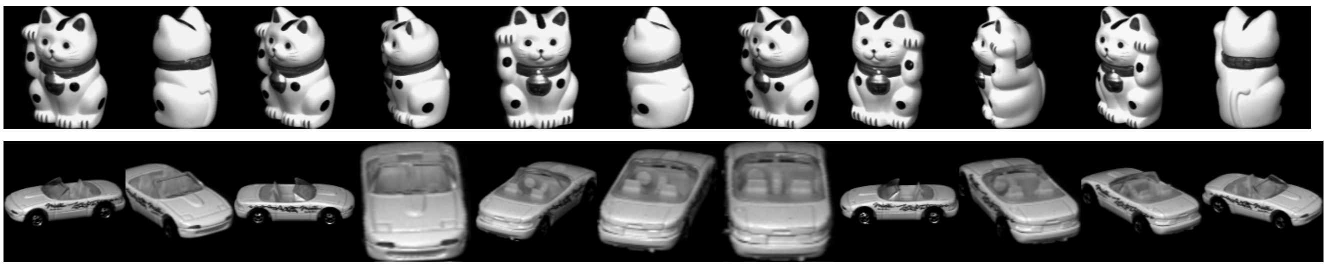


Figure 4.9: Example of real sequences of images

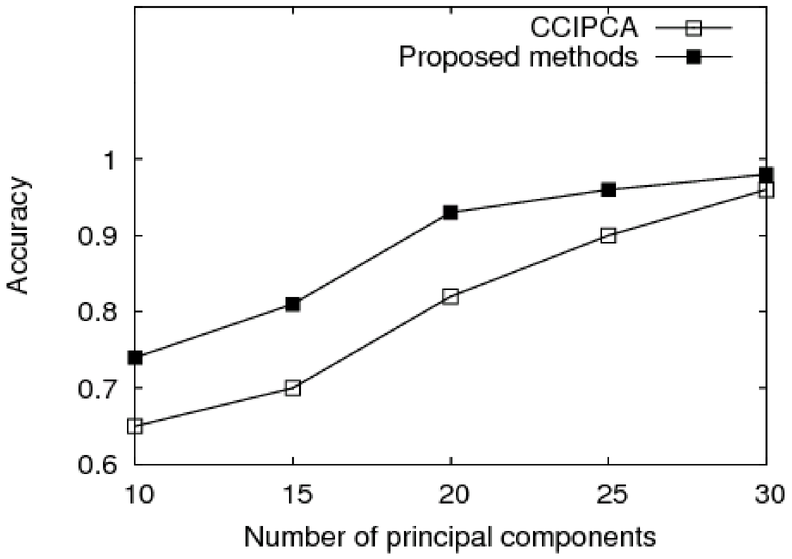


Figure 4.10: Comparison of accuracy

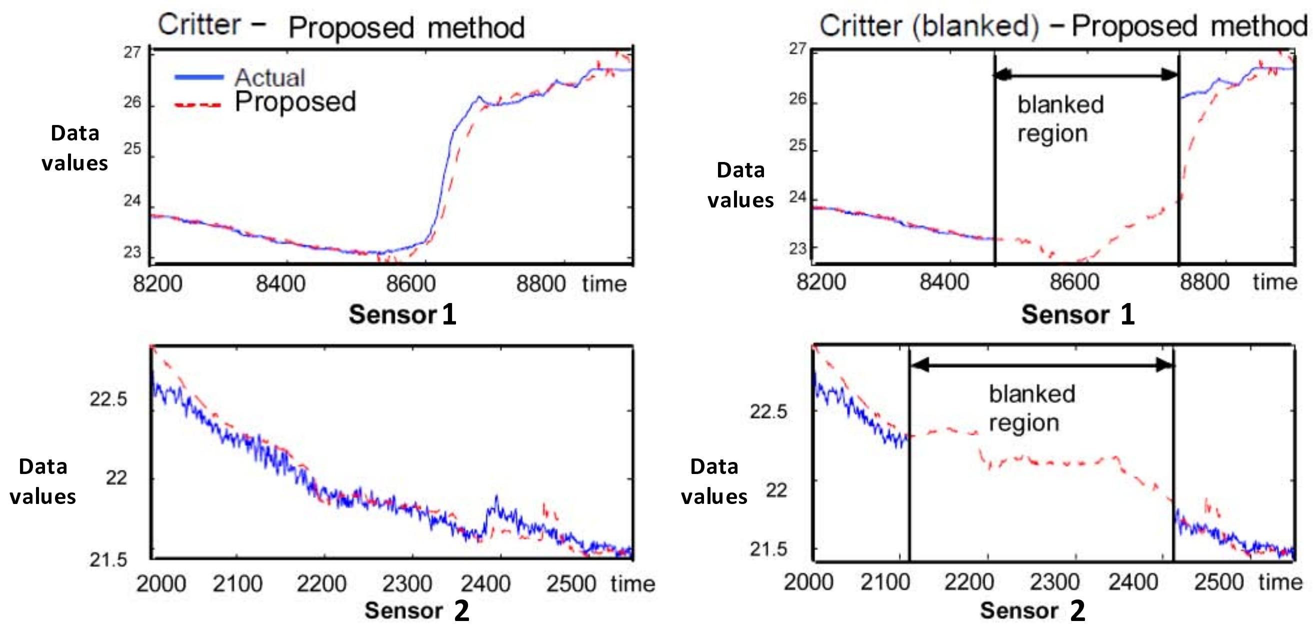


Figure 4.11: Detail of forecasts with blanked values.

Chapter 5

Flexible Timeline Clustering over Data Streams

Clustering techniques are also applied to multiple data streams for discovering continuous correlations. However, the existence of data evolution in data streams leads to another important issue of various clustering requirements at the same time. In this chapter, we discuss about the problem of flexible timeline clustering over multiple data streams. In the applications of flexible timeline clustering over data streams, user may be interested in clustering all stream pairs that are similar with each other during an arbitrary period of time. In the context of data streams, how to summarize the huge number of data resources for online calculation of similarity, and how to efficiently retrieve abstractions of the streaming data during a certain period in response to users' request are important issues.

5.1 Introduction

We devise a framework to dynamically and flexibly cluster multiple evolving data streams. The proposed framework consists of two phases: namely, *online statistics maintenance phase* and *offline clustering phase.* The online statistics maintenance phase realizes online collection and maintenance of summary statistics of fast data streams. Once a request of clustering is submitted, the offline clustering phase devises an adaptive abstraction algorithm to abstract statistics for approximating the user-desired subsequences as precisely as possible from the summary statistics hierarchies, and outputs the results of clustering over the statistics. Since the offline phase requires only the summary statistics as inputs, it turns out to be very efficient in practice. As shown in the

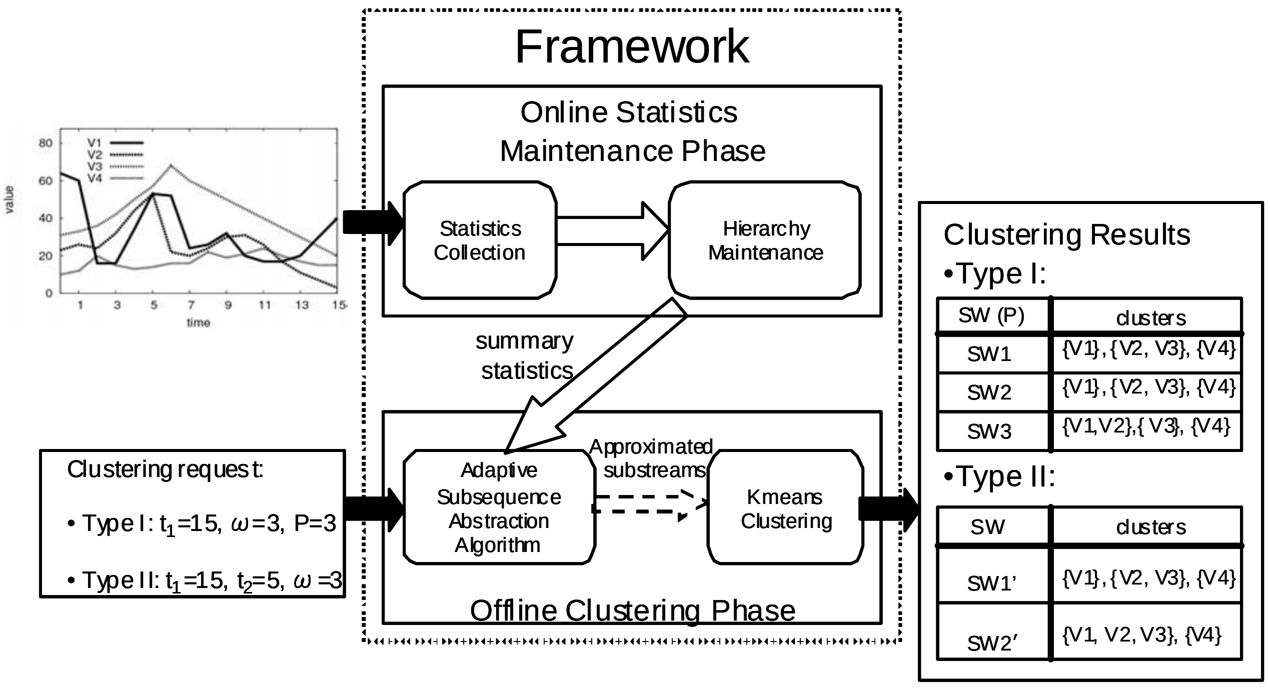


Figure 5.1: An overview of the proposed framework for clustering multiple evolving data streams.

complexity analyses and also validated by our empirical studies, the algorithms of our framework perform very efficiently in the data stream environment while producing clustering results of very quality.

The rest of this chapter is organized as follows: Section 5.2 describes the definitions of the proposed framework. Section 5.3 discusses how the summary statistics are collected and stored in the online statistics maintenance phase. Section 5.4 discusses how the offline clustering phase generates clusters over different time horizons at all possible time points using the statistics from the hierarchy structures. Section 5.5 analyzes complexity of the proposed framework. Section 5.6 reports our performance study on real and synthetic data sets. Section 5.7 concludes our study.

5.2 Framework

Figure ?? illustrates an example of the flexible clustering process over multiple evolving data streams. Continuous data streams are input into the online statistics maintenance phase, and the online statistics maintenance phase summarizes data streams into statistics and maintains the statistics in hierarchies. Once a clustering request is submitted into the offline clustering phase, the adaptive abstraction algorithm retrieves the statistics of the desired subsequences from the summary hierarchies, clusters statistics, and then outputs clustering results. In a clustering request, the user specifies arbitrary interested subsequence at arbitrary time point. By realizing the clustering over different subsequences, our framework also supports a user to explore the evolution of the nature of clusters over different sliding windows (Ws). Here, a subsequence in a sliding window is referred to as the subsequence from the user specified time point and with the length of user specified time horizon. The user may be interested in consecutive sliding windows or discrete sliding windows. In Figure **??**, we give the examples of consecutive sliding windows and discrete sliding windows as Type I and Type II, respectively. As shown in the example of Type I in Figure **??**, we get the results of clustering specifying $P = 3$ consecutive sliding windows with the time horizon size of $\omega = 3$ time points before time point t_1. Additionally, in the example of Type II, we observe the results of clustering before t_1 and t_2 specifying the same size of time horizon $\omega = 3$.

5.3 Online Statistics Maintenance Phase

In this phase, for a data stream, data samples are approximated into linear segments based on a new segmentation criterion to realize online summary statistics collection, and all these statistics are maintained in a compact hierarchy which supports flexible clustering in the offline clustering phase.

5.3.1 Online Statistics Collection

Several high-level representations have been proposed for approximation of time series, including Discrete Fourier Transform [?], Discrete Wavelet Transform [?], Singular Value Decomposition [?] and Piecewise Linear Approximation (PLA) [?]. Among these representations, PLA which approximates data samples into linear segments is one that is widely used because of its simplicity and is applicable to online approximation. Additionally, for the PLA linear approximation, due to the advantages of smooth approximation and low computational complexity discussed in [?], linear interpolation becomes our technical choice for online segmentation algorithm. A classic SW method in [?] is proposed for online segmental approximation of subsequences.

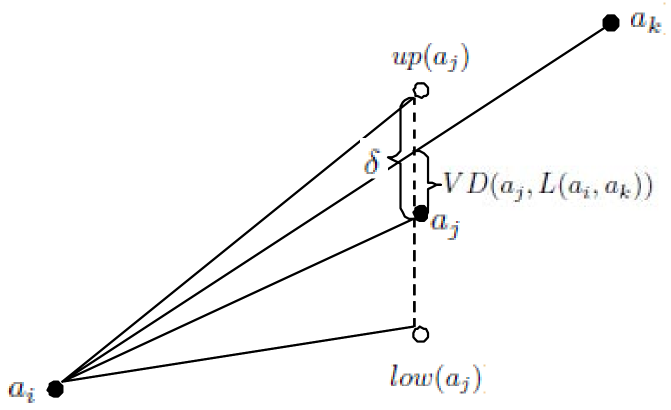


Figure 5.2: An example of tolerance slope interval.

We introduce a new segmentation criterion which reduces the computational complexity of the classic SW method in subsection 5.3.1.1. Based on this criterion, we propose a linear approximation method guaranteeing the approximation error of each data sample is less than the user specified maximum tolerance error δ in subsection 5.3.1.2.

5.3.1.1 Segmentation Criterion

It can be simply defined as the sum of the squares of vertical distance between actual data samples and the approximation line. The sum of the squares are used to evaluate the goodness-of-fit for potential segments. Another measure is the maximum vertical distance (MVD) between the data points and the approximation line, which is commonly used in linear interpolation [**?**]. In our method, we propose a new segmentation criterion based on MVD to make the approximation process more efficient.

[**Property 1.**] For $\delta > 0$, $i < j < k$, when we approximate the time series $T = (a_i, a_j, a_k)$ guaranteeing $VD(a_j, L(a_i, a_k)) \leq \delta$ *iff* $l(a_i, low(a_j)) \leq l(a_i, a_k) \leq l(a_i, up(a_j))$. Here $l(a_i, a_k)$ is slope of line $L(a_i, a_k)$ whcih is the result of linear approximation of the time series, and $low(a_j)$ and $up(a_j)$ are the tolerant approximate points of a_j with respect to the maximum vertical tolerance error δ.

This property is simply illustrated in Figure **??**. The interpolation line passing through a_i and a_k is the potential approximation line for a_i, a_j and a_k. For any data sample a_j where $i < j < k$, we need to determine whether the vertical distance of a_j to the interpolating line,

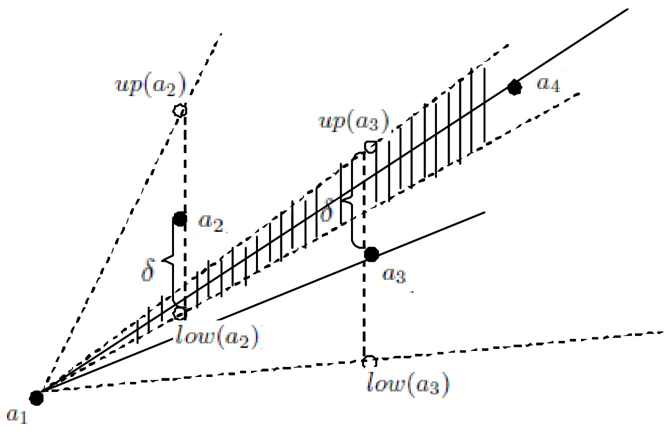


Figure 5.3: Illustration of tolerance slope interval of $L(a_1, a_4)$.

$VD(a_j, L(a_i, a_k))$, is within δ. In Figure **??**, we can see that in order to keep $VD(a_j, L(a_i, a_k)) \leq \delta$, $L(a_i, a_k)$ must lie between the lines $L(a_i, up(a_j))$ and $L(a_i, low(a_j))$, which is equivalent to $l(a_i, low(a_j)) \leq l(a_i, a_k) \leq l(a_i, up(a_j))$, and vice versa. Based on this property, we propose our segmentation criterion as follows:

[**Segmentation criterion.**] For the current segment $S = (a_i, \ldots, a_j)$ and $\delta > 0$, let $lowl_{i:j} = max(l(a_i, low(a_t)))$ and $upl_{i:j} = min(l(a_i, low(a_t)))$, where $i < t \leq j$. The newly arriving data sample a_{j+1} can be added into the segment S *iff* $lowl_{i:j} \leq l(a_i, a_{j+1}) \leq upl_{i:j}$.

This segmentation criterion is derived directly from Property 1:

$$\begin{aligned}
lowl_{i:j} &\leq l(a_i, a_{j+1}) \leq upl_{i:j} \\
&\Leftrightarrow \max_{i<t\leq j} l(a_i, low(a_t)) \leq l(a_i, a_{j+1}) \leq \min_{i<t\leq j} l(a_i, low(a_t)) \\
&\Leftrightarrow l(a_i, low(a_t)) \leq l(a_i, a_{j+1}) \leq l(a_i, up(a_t)), i < t < j \\
&\Leftrightarrow VD(a_t, L(a_i, a_{j+1})) \leq \delta, i < t < j \quad (Property 1) \\
&\Leftrightarrow a_{j+1}\, can\, be\, added\, to\, the\, segment\, S.
\end{aligned}$$

According to the segmentation criterion, each time a new data sample arrives. In order to determine the approximation line including the new data sample, we do not need to recompute all the vertical distances between the data samples and the new interpolating line as the classic SW method. Instead, we simply compare the slope of the new interpolating line with $[lowl, upl]$ of the current segment and update them accordingly. Here we note $[lowl, upl]$ as *tolerance slope*

Figure 5.4: A proposed approximation algorithm.

```
**Linear approximation algorithm based on a new segmentation criteria**
**Input:** time sequence $a_1, a_2, \ldots a_n, \ldots$ , maximum tolerance error $\delta$
**Output:** segmenting points $(s_1, s_2, \ldots s_k, \ldots)$
**Initial:** $i \leftarrow seg\_no \leftarrow csp\_id \leftarrow 1, s_1 \leftarrow a_1, lowl \leftarrow l(a_1, low(a_2)), upl \leftarrow l(a_1, up(a_2))$
**While** not finished segmenting time series
    $i \leftarrow i+1;$
    $upl \leftarrow \min(upl, l(s_{seg\_no}, up(a_i))), lowl \leftarrow \max(lowl, l(s_{seg\_no}, low(a_i)));$
    **If** $upl < lowl$ **then**
        $seg\_no \leftarrow seg\_no + 1;$
        $s_{seg\_no} \leftarrow a_{csp\_id};$        // the farthest CSP is a new segmenting point.
        $i \leftarrow csp\_id;$
        $lowl \leftarrow l(a_{csp\_id}, low(a_{csp\_id+1})); upl \leftarrow l(a_{csp\_id}, up(a_{csp\_id+1}));$        // reset
    **Else**
        **If** $lowl \le l(s_{seg\_no}, a_i) \le upl$ **then**
            $csp\_id \leftarrow i;$        // record ID of farthest CSP.
        **End**
    **End**
**End**
```

interval.

5.3.1.2 Approximation Method

The key idea of the proposed approximation method is to search for the farthest candidate segmentation point (CSP) to make the current segment as long as possible, and to reduce the number of segments for approximation. We call a data sample as a CSP if it may be chosen to be the next eligible segmentation point: i.e., the distances of all the points lying between the last segmentation point and the new chosen one to the new approximation line are all within δ.

According to the segmentation criterion, when a new data sample arrives, the newly updated tolerance slope interval $[lowl, upl]$ is contained in the older one. $[lowl, upl]$ is nonempty, which implies that it is possible for future data samples to be CSPs, even though the current new data sample is not a CSP. For example, in Figure **??**, the vertical distance of a_2 to $L(a_1, a_3)$ exceeds δ because $l(a_1, a_3)$ is outside of $[l(a_1, low(a_2)), l(a_1, up(a_2))]$; so a_3 is not CSP, but after updating the tolerance error interval, $lowl < upl$, it is possible for future data samples to be CSPs. In this example, a_4 is the furthest CSP because $l(a_1, a_4)$ falls into the tolerance slope interval $[max(l(a_1, low(a_t))), min(l(a_1, up(a_t)))]$ where $1 < t \le 3$, which is the shaded area in Figure **??**: i.e., the vertical distances of data samples a_2 and a_3 to $L(a_1, a_4)$ are within the maximum error

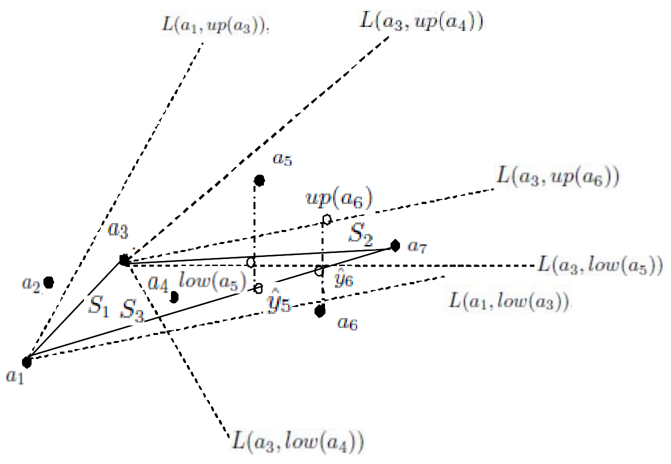


Figure 5.5: An example of integration of two segments.

tolerance δ. The tolerance slope interval shrinks as the new data sample arrives. When the tolerance slope interval becomes empty, (i.e., $upl < lowl$), it is impossible for any future data point to be a CSP. Therefore, because we choose the farthest CSP to be the next segmentation point, the current segment is certainly made the longest.

Finally, the approximation algorithm is given in Figure **??**.

5.3.2 Multi-level Summary Statistics Hierarchy

A Multi-level summary statistic hierarchy is proposed to store the statistics of each data stream. It plays an important role for our flexible clustering objective. It supports abstracting statistics of desired subsequences directly from the existing summary statistic without parsing raw data samples again. Therefore, clustering over statistics rather than original data samples addresses the time and space constraints in data streams environment. In this subsection, we discuss how to construct the multi-level summary statistic hierarchy for a data stream. In the next offline clustering phase, how to retrieve statistics of subsequences will be discussed in detail.

5.3.2.1 Segments Integration Criterion

In the hierarchy, statistics of lower level are stored in segments; therefore, in order to generate statistics of higher level, it is possible to integrate segments of lower level. In the example illustrated in Figure **??**, for the current segments $S_1 = (a_1, a_2, a_3)$ and $S_2 = (a_4, a_5, a_6, a_7)$, we

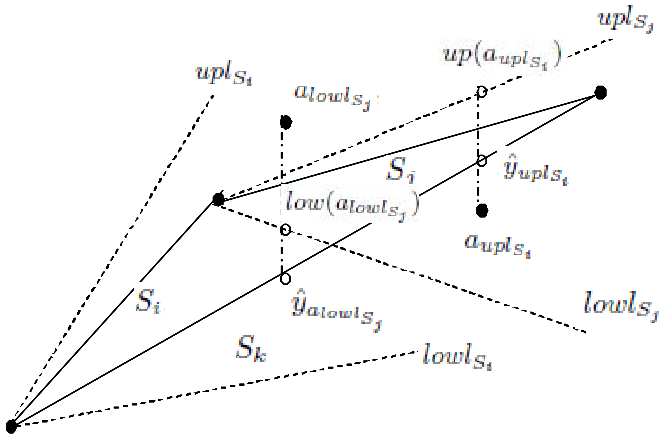


Figure 5.6: Geometric interpretation of segment integration criterion.

consider whether it is possible to integrate $S1$ and $S2$ into $S_3 = (a_1, a_2, a_3, a_4, a_5, a_6, a_7)$ for approximating all the data samples. Firstly, because $l(a_1, a_7)$ is included in the tolerance slope space $[low(a_3), up(a_3)]$ of S_1, S_3 guarantees that the approximation errors of data samples a_1, a_2, a_3 are less than δ. For the data samples in segment S_2, note that the tolerance slope space $[up(a_6), low(a_5)]$ of S_2 is referred to data sample a_3, not a_1; therefore, $[up(a_6), low(a_5)]$ is not the tolerance slope space for approximating segment S_2 using approximation segment S_3. Therefore, the segmentation criterion for data sample is not suitable for integration of segments. Focusing on the data samples a_5 and a_6 which related to the tolerance slope space of segment S_2, we find that $low(a_5) = y_5 - \delta$ and $up(a_6) = y_6 + \delta$ are the lower and upper boundary of approximation error of all the data samples in segment S_2. Therefore, if the approximation data of a_5 and a_6: $\hat{y}_{a_5}$ and $\hat{y}_{a_6}$ calculated from S_3 respectively, within $[low(a_5), up(a_6)]$, then S_3 also guarantees that the approximation error of each data sample in S_2 is less than δ. Otherwise, S_3 cannot approximate the data samples in segment S_2. In this example, because $\hat{y}_{a_5} < low(a_5)$, $S1$ and $S2$ cannot be integrated.

As illustrated in Figure **??**, for two consecutive segments S_i and S_j, we denote $[lowl_{S_i}, upl_{S_i}]$ as the tolerance slope interval of S_i. $a_{lowl_{S_j}}$ and $a_{upl_{S_i}}$ are the data samples which determine the tolerance slope interval of S_j. $\hat{y}_{lowl_{S_j}}$ and $\hat{y}_{upl_{S_i}}$ are the approximation data of $a_{lowl_{S_j}}$ and $a_{upl_{S_i}}$ calculated from approximation segment S_k, which is the potential approximation line after integrating S_i and S_j. We propose criterion for segments integration as follows:

[Segments Integration criterion.] For the current consecutive segments S_i and S_j, the potential approximation segment S_k by integration of S_i and S_J is approved *iff* the following two conditions are satisfied:

(a) Slop of S_k within $[lowl_{S_i}, upl_{S_i}]$.

(b) $\hat{y}_{lowl_{S_j}} \geq low(a_{upl_{S_i}})$ and $\hat{y}_{upl_{S_i}} \leq up(a_{upl_{S_i}})$ where $low(a_{upl_{S_i}}) = y_{lowl_{S_j}} - \delta$ and $up(a_{upl_{S_i}}) = y_{upl_{S_i}} + \delta$.

5.3.2.2 Summary Statistic Hierarchy

Figure **??** describes the procedure of constructing the summary statistic hierarchy for a data stream. When a new data sample arrives, it is summarized into the temporary bucket firstly according to the segmentation algorithm discussed in Section 5.3.1.2. Until the number of data samples in the temporary bucket (TB) reaches B_t, the statistic of all segments in TB will be stored into a new basic window at level 0. When B_h new basic windows are accumulated at level $(H-1)$, a new basic window at level H is generated. Note that in this step segments in basic windows may be integrated. The integration process is explained in Figure **??**. For segments of a basic window of level $(H-1)$, obviously, the integration of two segments possibly occurs at the connection of two basic windows of level $(H-1)$, and other segments need to be copied into the new basic window at level H. If the two segments of two connected basic windows of level $(H-1)$ satisfy the segments integration criterion as discussed in subsection 5.3.2.1, then the two segments are integrated. Then it is necessary to check whether the newly integrated segment and the next segment of the connected basic window can be integrated or not. If the segments integration criterion is not satisfied, the integrated segment and all of the remaining segments of the connected basic window will be copied into the new basic window at level H.

Generally, n hierarchies are maintained for n data streams. Due to the space limitation in the streaming environment, only the latest α basic windows are maintained at each level. Note that α should not be set smaller than B_h in order to keep enough basic windows at lower level for generation of a basic window at a higher level.

```
Procedure of constructing summarized statistics hierarchy (for one data steam)
Input: data samples $a_1, a_2, \cdots a_n, \cdots$ , size of temporal bucket $B_t$ , $B_h$ , maximum tolerance error $\delta$
Output: summarized statistics hierarchy
Initial: $level \leftarrow 0$ , empty temporal bucket, $seg \leftarrow null$, $i \leftarrow 0$
    While not finished data streams
1.      For each incoming data sample, approximate the data samples into temporary bucket
        incrementally according to proposed approximation algorithm.
2.      If the number of data samples in the temporary bucket is less than $B_t$, then return to Step 1.
        Else
3.          Create a new basic window at level 0.
4.          Store the statistics of segments in temporal bucket into the new basic window.
5.          Empty temporal bucket.
6.      If the number of basic window at $level$ equals to $B_h$, then
7.          Create a new basic window at $level + 1$.
8.          While not finish $B_h$ basic windows at $level$
9.              $seg \leftarrow$ first segment of the first basic window
10.             $seg \leftarrow$ Procedure for integrating segments ($seg$ , $i$ , 1, 0 ).
11.             Initialize $seg$ .
12.             $i \leftarrow i + 1$ .
13.         End
14.     End
15.     End
16. End
```

Figure 5.7: Procedure for constructing summarized statistics hierarchy.

5.4 Offline Clustering Phase

In this section, we discuss the clustering process when a clustering request is submitted. Note that user-specified time horizon w would be different from the size of basic windows maintained in the summary statistic hierarchies. It is important to retrieve statistics of the user desired subsequences as precisely as possible. Here we define an *entry* which contains the statistics of the desired subsequence. We have the following definitions for appropriate level selection in order to abstract entries which are most recent from user-specified time point t_1.

Definition 1. The *highest level* and *lowest level* for generation of entries are defined as

$$H_{max} = \lfloor \log_{B_h}(\frac{w}{B_t}) \rfloor \tag{5.1}$$

$$H_{min} = argmin\{H | w \leq (t_1 - t_{[H,i,k_i]}) + \alpha_H h_H\} \tag{5.2}$$

where α_H is the exact number of basic windows at level H, $t_{[H,i,k_i]}$ is the end time point of the last segment (k_i-th segment) of the ith basic window at level H from t_1, and $h_H = B_t {B_h}^H$ is the size of a basic window of level H.

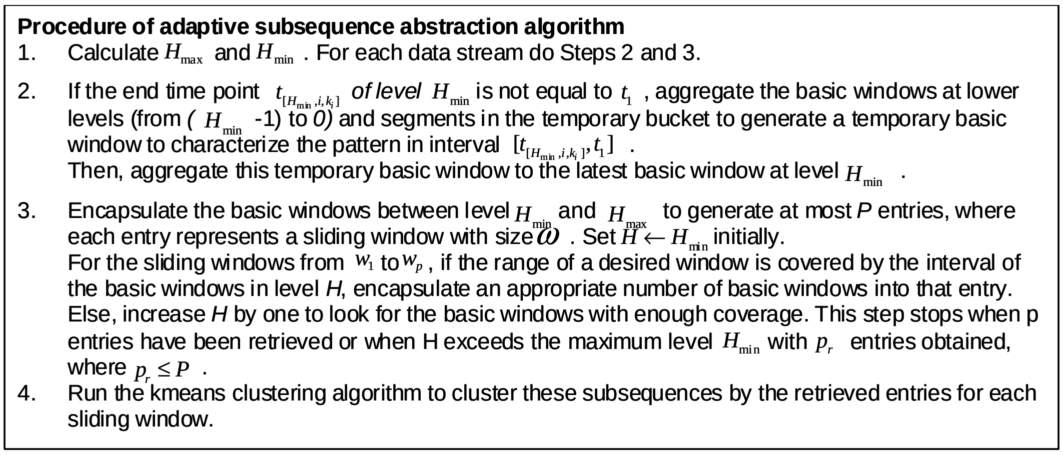
In the above definitions, window size of level H_{max} on the hierarchy is no larger than w, and

```
Procedure for integrating segments (segment $seg$, int $i$, int $j$, int $flag$)
Input: $j$-th segment of $i+1$ basic window $temp_j$

1.  If $seg$ is the last segment of basic window $i$ or $flag = 1$ then
2.      If $seg$ and $temp_j$ satisfy the segments integration criterion, then
3.          Integrate the two segments as $seg$: data samples within the two segments are
            approximated by a interpolation line passing the fist data sample of $seg$ and
            the last data sample of $temp_j$.
4.          $j \leftarrow j+1$.
5.          $flag \leftarrow 1$.
6.          $seg \leftarrow$ Procedure for integrating segments ($seg$, $i$, $j$, $flag$).
        Else
7.          While not finish the segments at basic window $i$
8.              $seg \leftarrow temp_j$.
9.              Store $seg$ into the new basic window at the higher level.
10.             $j \leftarrow j+1$.
11.             $flag \leftarrow 0$.
12.         End
13.     End
14. End
```

Figure 5.8: Procedure for integrating segments.

H_{min} is the lowest level whose basic windows are still enough to illustrate the pattern with window size w. Therefore, from level H_{min} to H_{max}, we can abstract $(H_{max} - H_{min})$ entries which closely satisfy the user's request and most recent from t_1. The outline of adaptive abstraction algorithm for statistics is shown in Figure ??. In the first step the values of H_{min} and H_{max} are calculated according to the time horizon w. Then, entries are retrieved from each data stream by Step 2 and Step 3, where $t_{[H_{min},i,k_i]}$ means the end time points of the last segment (k_ith segment) of the last basic window (ith basic window) of level H_{min}. In Step 3, we encapsulate the basic windows at one level of the hierarchy which cover the desired time horizon w into an entry. Let us consider Step 2 in the proposed abstraction algorithm. In the example shown in Figure ??, assume the level $H_{min} = 2$. Since the end time point $t_{[2,1,k_1]}$ is not equal to t_1, we aggregate the basic windows from lower levels (from level 1 to 0) and segments from the temporary bucket to generate a temporary basic window which characterizes the interval between $t_{[2,1,k_1]}$ and t_1. Then, it is aggregated into the latest basic window of level 2. After abstracting the statistics of user-desired subsequences, the *k-means clustering* algorithm is utilized to declare the group behavior of data streams in each sliding window.



Procedure of adaptive subsequence abstraction algorithm

1. Calculate $H_{\max}$ and $H_{\min}$. For each data stream do Steps 2 and 3.
2. If the end time point $t_{[H_{\min},i,k_i]}$ *of level* $H_{\min}$ is not equal to t_1 , aggregate the basic windows at lower levels (from *(* $H_{\min}$ -1) to *0)* and segments in the temporary bucket to generate a temporary basic window to characterize the pattern in interval $[t_{[H_{\min},i,k_i]}, t_1]$.
   Then, aggregate this temporary basic window to the latest basic window at level $H_{\min}$.
3. Encapsulate the basic windows between level $H_{\min}$ and $H_{\max}$ to generate at most *P* entries, where each entry represents a sliding window with size ω . Set $H \leftarrow H_{\min}$ initially.
   For the sliding windows from w_1 to w_p , if the range of a desired window is covered by the interval of the basic windows in level *H*, encapsulate an appropriate number of basic windows into that entry. Else, increase *H* by one to look for the basic windows with enough coverage. This step stops when p entries have been retrieved or when H exceeds the maximum level $H_{\min}$ with p_r entries obtained, where $p_r \leq P$.
4. Run the kmeans clustering algorithm to cluster these subsequences by the retrieved entries for each sliding window.

Figure 5.9: The outline of the adaptive subsequences abstraction algorithm

A typical function for measure quality of cluster in sliding window W_i is

$$Cost(CL(W_i) = \frac{\sum_{C_j(W_i)} \sum_{S_q(W_i) \in C_j(W_i)} dis(S_q(W_i) - C_j(W_i).center)}{n * w}.$$

where $C_j(W_i).center$ is the center of cluster $C_j(W_i)$, and $dis(S_q(W_i) - C_j(W_i).center)$ is the distance of each member of cluster $C_j(W_i)$ with the center.

Euclidean distance is used as the distance measurement between two subsequences. If more than one window is observed, the total quality will be the average clustering quality of the retrieved windows, as follows:

$$Cost(CL) = \frac{\sum_{i=1}^{P_r} Cost(Cl(W_i))}{P_r}. \tag{5.3}$$

where P_r is the number of windows actually retrieved.

5.5 Complexity Analysis

The complexities of the online and offline phases in our framework are as follows, where n is the number of streams and m is the number of data samples in each stream. We denote the number of segments and average segment length of a series by K and L, respectively; thus, $m = K * L$.

Time Complexity of Approximation. For our approximation method, in the worst case, theoretically, it is possible for the tolerance slope interval to stay nonempty even if a new CSP never arrives. For example, in Figure **??**, suppose the current segmentation point is a_1; if every

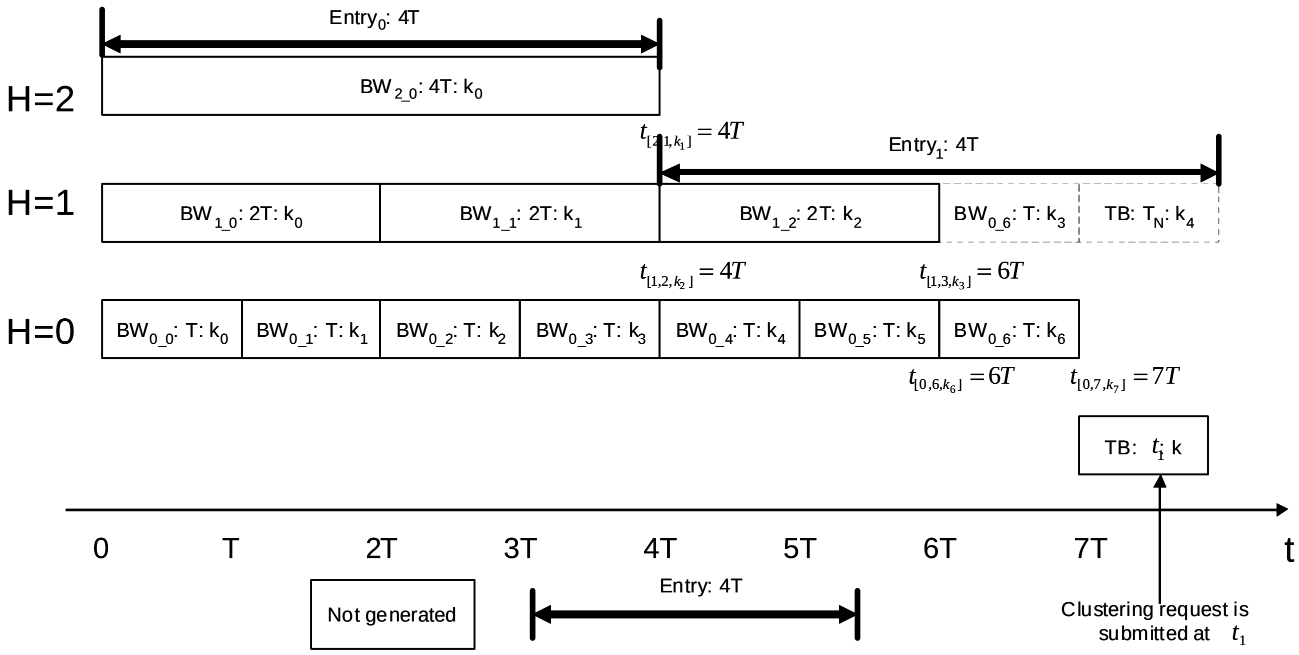


Figure 5.10: The illustration of summary hierarchy, where $BW_{1\_0} : T : k_1$ represents the 0-th basic window at level 1 of time horizon $B_t = T$, k_1 is the number of segments in this basic window, $t_{[1,1,k_1]}$ records the end time point of the last segments of the first basic window at level 1, $Entry_1$ is the generated entry for the most recent sliding window from t_1 and $B_h = 2$.

data sample $a_i (i = 3, \ldots)$ is not a CSP but is very close to the straight line $L(a_1, low(a_2))$: i.e., $VD(a_i, L(a_1, a_2)) > \delta$. The tolerance slope interval will surely stay nonempty. When the tolerance slope interval finally becomes empty after searching through large numbers of non-CSPs, the next segmentation point is just a_2. Therefore, the time complexity of our approximation method is $O(Kn)$ in the worst case. While for the classic SW method [?], each time a new data sample arrives, we have to remeasure the approximation error of all data samples. Since the length of a potential segment grows from 2 to L and we do not need to measure the endpoints of the segment, the time to compute a single segment is $\sum_{i=1}^{L-1} O(i) = O(L^2)$. Therefore, the time complexity is $K * O(L^2) = O(Lm)$. While for segments integration, our method can realize incremental process, the classic SW method has to parse the original data sample to calculate the approximation error, and this computation may be expensive, especially for long segments.

Time Complexity of Online Phase. For each data stream, in level 0, $\frac{m}{B_t}$ basic windows can be generated in time $O(m)$ after m data samples arrive. In each basic window, B_t

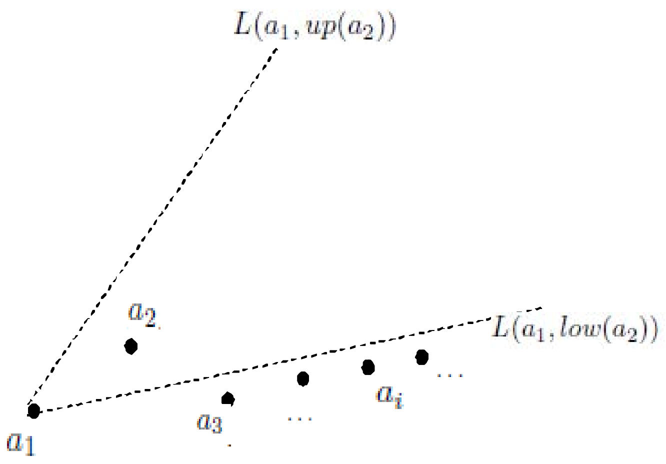


Figure 5.11: The worst case of our approximation method

times iteration will be processed to generate K segments. In the worst case, we can get the total time complexity at level 0 is mKB_t. Aggregating B_h basic windows of lower level and generation into a new basic window of higher level requires integration or copy of segments, whose complexity is $O(1)$. Accordingly, the total number of basic windows at higher levels is $\frac{m}{B_t} \times \left(\frac{1}{B_h} + \frac{1}{{B_h}^2} + \cdots + \frac{1}{\log_{B_h}(\frac{m}{B_t})}\right) \leq \frac{m}{B_t} \times \left(\frac{\frac{1}{B_h}}{1-\frac{1}{B_h}}\right) = O\left(\frac{m}{B_t(B_h-1)}\right)$. The time for building these basic windows will be $O\left(\frac{m}{B_t(B_h-1)}\right)$. Consequently, the overall time complexity of the online recent-biased statistics maintenance phase for n streams is $nmKB_t$.

Space Complexity of Online Phase. For each data stream, in level 0, $\frac{m}{B_t}$ basic windows have been created after m points arrived. B_t basic windows at lower level are aggregated to a basic window at higher level. Accordingly, the height of the summary hierarchy is $\log_{B_h}(\frac{m}{B_t})$ and each level maintains at most α basic windows. Note that each basic window contains a constant number of parameters. Consequently, totally $n\left(\alpha \log_{B_h} \frac{m}{B_t}\right)$ basic windows are maintained for n streams. We denote the number of segments of one basic window as K. Therefore, the space complexity of the online recent-biased summary statistic phase is $O\left(n\alpha K \log_{B_h} \frac{m}{B_t}\right)$.

Space Complexity of Offline Phase. The complexity of k-means clustering algorithm is $O(knd)$, where d is the number of dimensions. Since an entry of a clustering request is represented by at most α basic windows with constant number of parameters. Therefore, the time complexity of the offline clustering phase is $O(kn\alpha)$ for one entry.

From the above analysis, we can see that both of the time and space complexity of the online

$\chi_j(t)$	$\lambda_i > \lambda$	$\lambda_j \leq \lambda$
$state_j = 0$	0	$\frac{t \bmod LEN}{LEN}$
$state_j = 1$	1	$1 - \frac{t \bmod LEN}{LEN}$

Figure 5.12: Fast/Slow Synthetic Dataset

maintenance phase are linear with the number of segment K. Therefore, our proposed approximation method of finding the least number of segments makes the framework more efficient.

5.6 Empirical Results

To access the performance of our framework, we have conducted a series of experiments. In Section 5.6.1, we introduce the test environment and data sets. Next, we investigate the sensitivity analyses of parameters in Section 5.6.2. Then the performance of the offline clustering phase of our framework on the real data set is evaluated in Section 5.6.3. Finally, we examine the scalability of the online statistics maintenance phase in Section 5.6.4.

5.6.1 Test Environment and Data Sets

All of our experiments are conducted on a PC with AMD Athlon(tm) 64X2 Dual Core Processor and 1G memory, which runs Windows Vista Business operating system. In order to evaluate our framework, both the synthetic data set and the real data set are used.

Real dataset. We obtain average daily temperatures of 290 cities around the world from Temperature Data Archive, the University of Dayton. The daily average temperatures of each city are recorded from 1 January 1995 to present. Each city is regarded as a data stream and each stream has 3, 416 data samples.

Synthetic dataset. This data is designed to provide different fractions of fast/slow changing data sets . For a given fraction of fast data, λ, each series is the concatenation of several segments with length LEN. Data Samples of the j-th segment are generated as follows:

$$g_j(t) = (S + \eta) * \chi_j(t) + \varepsilon(t), \tag{5.4}$$

where η and $\varepsilon(t)$ are drawn from a standard normal distribution $N(0, 1)$, t is in the range $[0, LEN-1]$, and $\chi_j(t)$ is decided according to the rules in Figure **??**.

The slope of the jth segment is determined by λ_j and $state_j$, where λ_j is a value drawn uniformly from $[0, 1]$, and $state_j$ is either 0 or 1. Initially, $state_0$ is set to 0. Then, each $state$ value is set according to its value in the previous segment and the current λ_j value. More specifically, $state_j \leftarrow state_{j-1}$ if $\lambda_j > \lambda$, and $state_j \leftarrow 1 - state_{j-1}$ if $\lambda_j \leq \lambda$. By this setting, about $(1 - \lambda)$ fraction of segments in a series are slowly changing segments with the corresponding slopes being close to 0. On the other hand, about λ fraction of segments in a series are fast changing segments with the slopes being close to $\pm\frac{S}{LEN}$. In our experiments, S and LEN are set as 30 and 50, respectively. Therefore, the slopes of fast changing segments are about ± 0.6. This data supports the need of varying the number of data samples and the number of data streams for the scalability analysis.

We provide extensive empirical comparison of our approach with the classic SW method. It has been shown in [**?**] that the performance of the approximation algorithm is influenced by the setting of maximum tolerance error δ. For example, when δ is very small, data samples form $(n-1)$ segments (any two adjacent data samples form a segment in the interpolation case). When δ becomes positive infinite, all algorithms give only one segment. In our experiments, we use the relative series values, called the Maximum Error Percentage (MEP), instead of the absolute maximum tolerance error. The quality of these methods is measured as the ratio of the quality obtained by running the k-means algorithm on the raw data streams. Because the choice of initial cluster centers in a k-means algorithm affects the quality of results, the best set of clusters from multiple restarts is employed to alleviate this effect. Note that parameter B_t represents the number of values to be merged into a basic window while parameter B_h represents the number of basic windows to be merged into a new basic window in a higher level. Both of these two parameters imply approximation of several data samples with a coarser summary. Therefore, they are set as the same value in our experiments to avoid presenting two experiments with similar effects. Without loss of generality, we assume that the bucket size $B_t = B_h = B$.

5.6.2 Sensitivity Analysis

In this section, two parameters of the summary hierarchy, which are the number of basic windows maintained in each level α and bucket size B are investigated on the weather data with a fixed time horizon $w = 50$. In addition, the sensitivity of our approximation method with parameter δ is also discussed here. In the complexity analysis, the worse case which retrieves at most α basic windows for clustering, is discussed. Note, in practice this method does not encounter the worst case oftenly. Therefore, the clustering time is mostly dominated by the execution time of the k-means clustering algorithm. As shown in Figure **??** and Figure **??**, α does not have much influence on the costs and execution time of clustering. Note that since the clustering algorithms are only applied to the statistics maintained, the clustering method is much efficient than the clustering on the raw data streams, though with slight additional time for calculation from the summary hierarchy. Therefore, when the bucket size is too small, the execution time of framework will be larger. As shown in Figure **??** and Figure **??**, clustering on the summary hierarchies of our framework is able to achieve almost the same quality as that on the raw data streams efficiently. As shown in Figure **??** and Figure **??**, obviously, with larger maximum tolerance error, the number of segments decrease; so the response time will be shorter. Also, we can see that our proposed approximation method is more efficient than classic SW method with less approximation error.

5.6.3 Clustering Quality

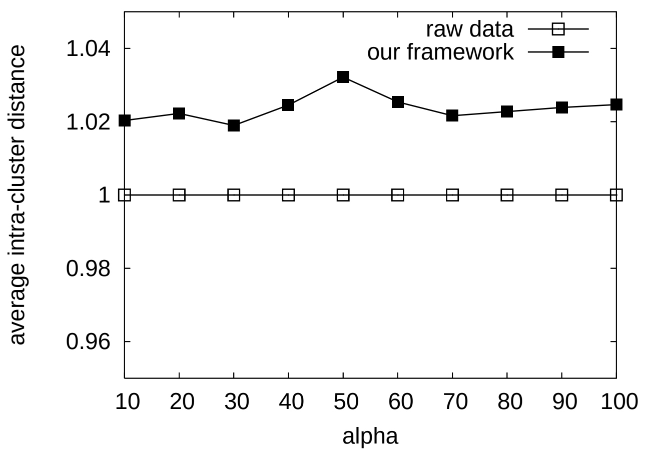
The performance of the framework on real data is then evaluated with a fixed bucket size $B = 12$. As shown in Figure **??** and Figure **??**, while varying the time horizon and the number of sliding windows observed, the framework can attain the same good clustering quality as the clustering on the raw data streams, with significantly shorter execution time, as shown in Figure **??** and Figure **??**.

5.6.4 Scalability Analysis

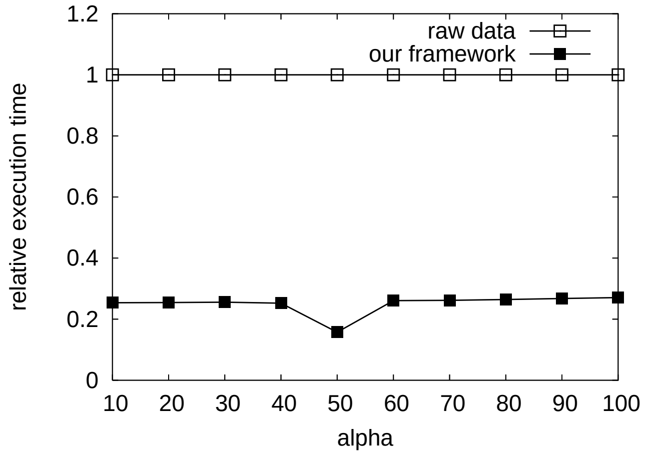
To evaluate the scalability of the online maintenance phase, the scale-up experiments on both the number of data samples and the number of data streams are conducted. As shown in Figure **??**, as the number of data samples in each stream increases from $1,000$ to $10,000$ the execution time grows linearly. This property also holds when the number of data streams varies from 100 to $1,000$ as shown in Figure **??**. These results conform to our analyses in Section 5 which state that the time complexity of the online maintenance phase of the framework is linear in both the number of streams and number of data samples in each stream.

5.7 Summary

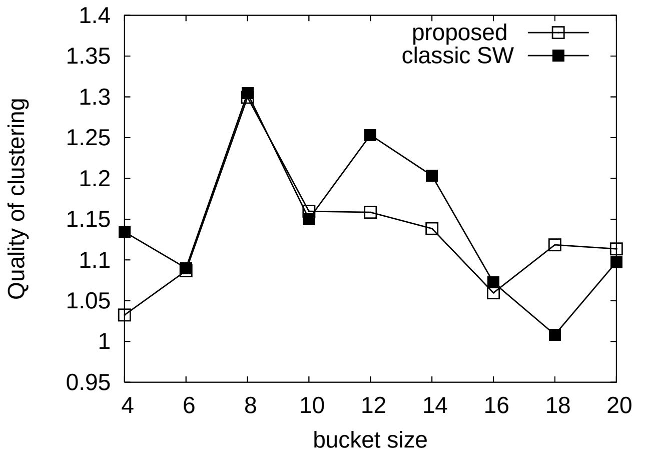
In order to support flexible timeline clustering, we devised a two-phased framework to dynamically cluster multiple evolving data streams. While providing a general framework of clustering on multiple data streams, our framework has two advantages: single data scan for online statistics collection and compact multi-resolution approximation, which can address respectively the time and space constraints in a data stream environment. Furthermore, with the multi-resolution hierarchies of data streams, flexible clustering demands can be supported. The online statistics maintenance phase provided efficient algorithms to generate and maintain statistics in time linear in both the number of streams and the number of data samples in each stream. On the other hand, an adaptive abstraction algorithm is devised to abstract statistics of a user-interested subsequences from the summary hierarchies as precise as possible. As shown in empirical studies, the algorithms performed very efficiently in data stream environment.



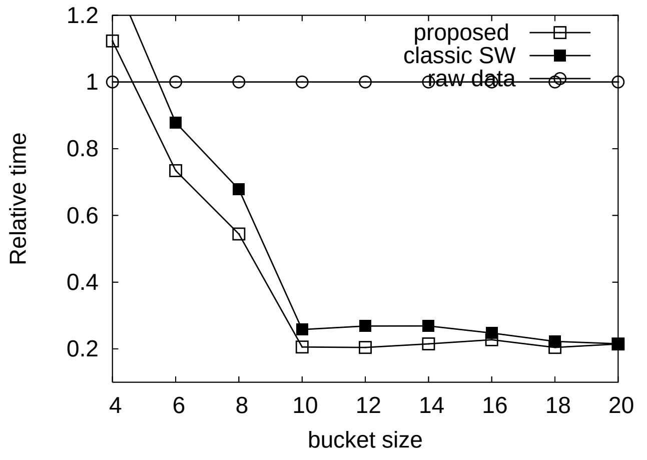
(a) Clustering Quality versus alpha



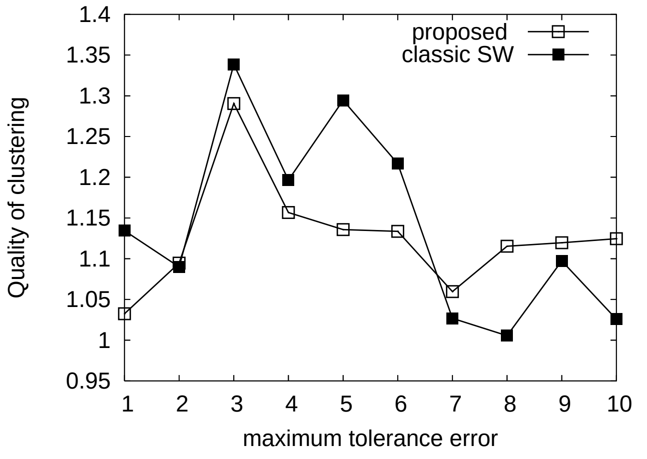
(b) Time versus alpha



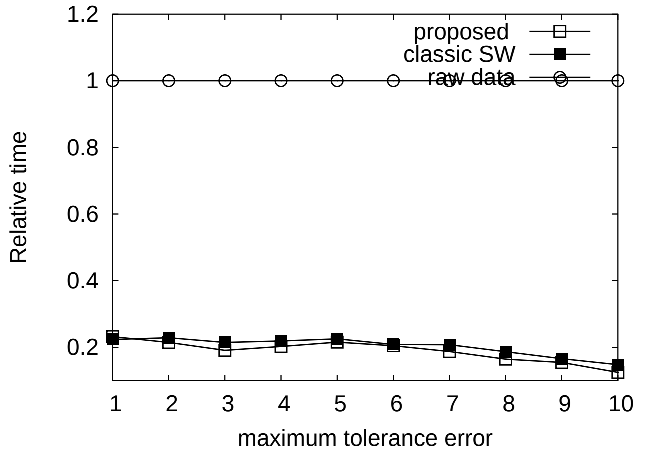
(c) Clustering Quality versus bucket size



(d) Time versus bucket size

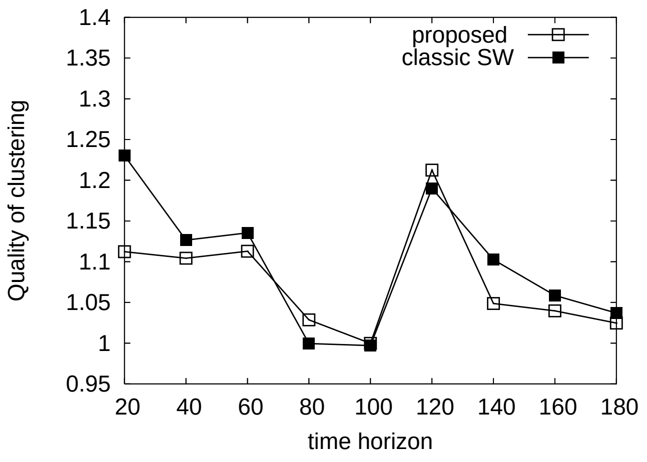


(e) Clustering Quality versus δ

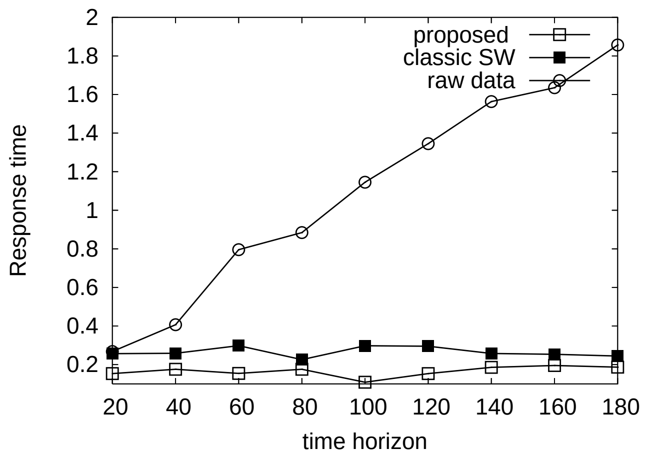


(f) Time versus δ

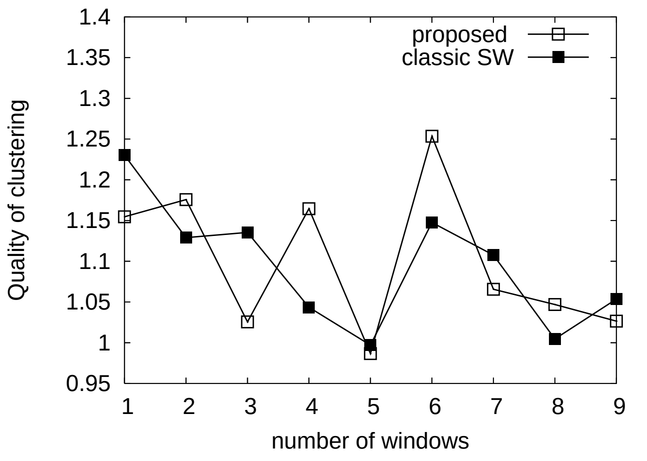
Figure 5.13: Sensitivity Analysis



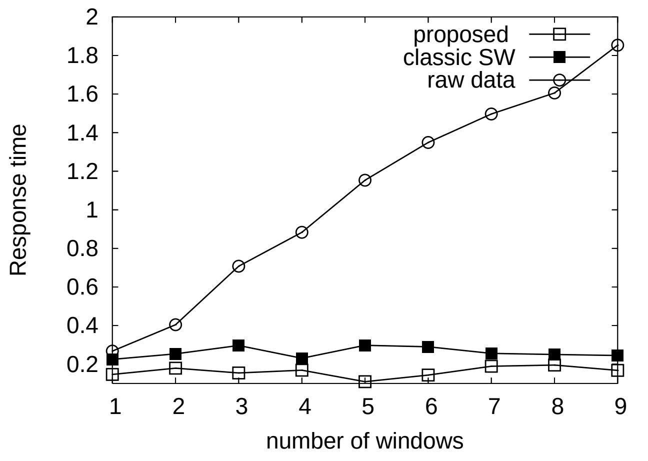
(a) Clustering Quality versus size of time horizon



(b) Time versus size of time horizon

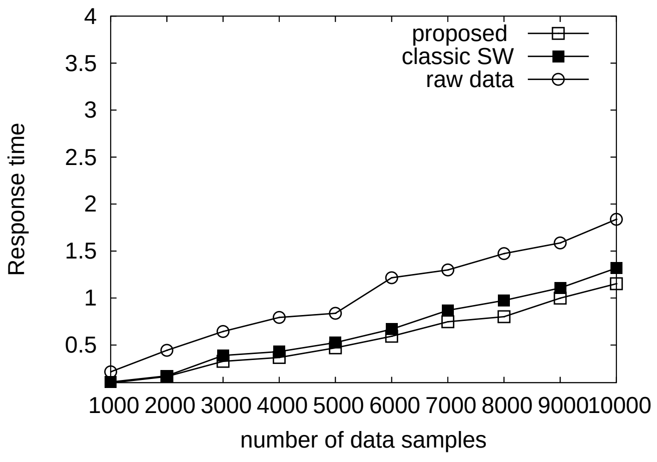


(c) Clustering Quality versus number of entries

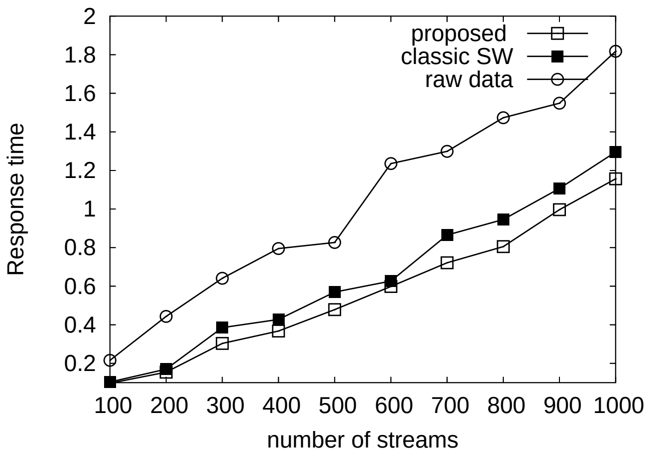


(d) Time versus number of entries

Figure 5.14: Experimental results on Real Data



(a) Time versus number of data samples



(b) Time versus number of data streams

Figure 5.15: Scalability Analysis

Chapter 6

Cross-domain Correlation Mining among Multiple Data Sources

In this chapter, given a news article referring to one company, we decide whether it is a piece of good news that is followed by a moving up trend in the company's stock market or a piece of bad news reversely. The novelty of our proposed framework is the achievement of dynamic analysis of the complex correlation between online news articles and stock price series. Existing research work did not support flexible identification of the trends in stock price series, or take account of the case that temporal consecutive news articles may influence the stock market sensitively. In our proposed framework, we combine the investigations of the discrete correlation as well as the continuous correlation.

In this problem, the process of classification achieves to mine discrete correlation, for the reason that we treat the collection of news articles as transaction data consisting of words, and the news articles are independent with each other. While, in order to improve the accuracy of prediction, we also take account of continuous correlations. On one hand, in the generation of news articles for learning, we abstract trends of stock prices, and then label the news articles according to corresponding trends in stock prices. On the other hand, taking account of the evolving social environment, we can see that people may be more interested in the contents of consecutive news articles which may influence the stock market sensitively. Therefore, it is prefer to identify the trends in stock market dynamically. We propose dynamic mechanism of choosing sliding windows to identify trends of stock prices according to the contents of consecutive news articles as shown in the green part of this figure. In order to detect the sensitive topic in consecutive news articles, we observe continuous changes of the words' occurrences.

In this chapter, we take the individual company as the research objective. The news articles referring to the company are useful to predict the stock prices of the company itself. In fact, the proposed methods introduced in Chapter 4 is also appliable to the framework proposed in this chapter, and the results are useful to explain the financial market involving multiple companies. For example, firstly, we can discover the abonormal behaviors of stocks comparing to the key trends in the whole market according to the method proposed in Chapter 4, and we can explain how it happened by analysis the corresponding news articles.

Comparing the processes of identifying patterns or trends in time series data discussedn in Chapter 5 and this chapter, we can see that Chapater 5 provides mechanism to store historical data as well as the flexibility of multi-solution clustering. Nevertheless, in this chapter, we foucus on dynamically choosing sliding windows for abstracting trends now and predict future accurately. Thus, we scan the streaming stock price data samples as them arrive and then drop them immediately, without storing. Certainly, it is also possible to apply the methods proposed in Chapter 5 to discover similar patterns of different stocks and explain why these patterns happened.

6.1 Introduction

Although stock prices follow a random walk and are extremely difficult to predict, as pointed out in [**?**, **?**, **?**], the analysis of textual information concerning to the financial market is helpful to generate profitable action signals (buy or sell) accurately. As we can see that in contrast to numeric time series data, textual data contains not only the effect (e.g., "the stock of Apple Inc. fell about 1 percent") but also the possible causes of the event (e.g., "after warning in a regulatory filing that profit margins might narrow next year"). Unfortunately, because of the large volume of textual information, it is difficult for investors to track and captur each of them. Therefore, it would be very helpful to have a system that can automatically classify information which in turn can be used as indicators of forthcoming trends in stock market.

We investigate how news articles can be useful to forecast trends of stock prices by discovering the correlation between news articles and trends of stock prices. Given a news article, we decide whether it is a piece of good news followed by a moving up trend in stock market or a piece of bad news reversely, in addition to predicting how will the fluctuation of stock price be influenced by the news article. We propose classification and regression methods to address this problem.

Labels of historical news articles for learning are generated by diagnosis and definition of trends in historical stock price series. We propose an algorithm based on estimating variance of data density in sliding windows to segment continuous stock price series. Trends are defined as piecewise linear fitting to these segments, and are assigned with labels: *up*, *steady* and *down* by an agglomerative hierarchical clustering algorithm. Taking the advantage of parameter h_t as the length of sliding windows for identifying trends, it is flexible for investors to grasp short-term or long-term trends of stock prices. Additionally, we propose ideas to determine h_t automatically and dynamically, according to the contents of the news article sequences. For example, some hot topics in a period may influence the financial market sensitively.

Therefore, the news articles are labeled by alignment with trends in stock prices according to the time when articles were released and when each trend was promoted. In the discussion of how to represent articles for classification and regression, our ideas are as follows: (1) instead of characterizing a news article by words, we abstract pairs of syntactically depended words, namely terms, and present news articles as a vector of weighted terms; (2) for the reason that articles may describe the same subject with different vocabulary, we count the statistics of terms; taking into account of the semantic coherence among words; and (3) we weight terms with respect to class relevance and discrimination. For example, we assign higher weights to the terms if they occur in news article of *up* class but not occur in articles of the opposite *down* class.

Based on support vector classification technique, we classify articles that are highly associated with particularly labeled trends of stock prices. Meanwhile, we present a support vector regression method to predict the fluctuation of stock prices influenced by the news articles. Experiments of our proposed methods yield high accuracy of prediction. The proposed mechanism for dynamically choosing sliding window to identify trends is also proven to be effective.

We realize a more flexible and accurate investigation of correlation between news articles and stock prices:

1) Identification of trends in stock prices based on sliding windows satisfies both short-term and long-term needs;

2) Representation of news articles as vectors of terms weighted with class relevance and discrimination improves the accuracy of prediction;

3) Dynamic choice of sliding windows in terms of the consecutive news articles for identifying trends of stock prices realizes online investigation of the influence of news articles to stock prices;

4) Prediction of forthcoming stock prices in terms of direction and magnitude is much more informative and helpful for investors' decision-making.

The overview of our methods is illustrated in Figure **??**. In the following sections we elaborate our ideas in each of the procedures. Firstly, we identify and label trends of stock prices. Then according to the time when news articles are released and the labels of trends at the same time, we assign labels to news articles. Thus the labeled training news articles data is generated. In order to classify the news articles, we select terms of dependent words as features of articles. Finally, by utilizing the Support Vector Classification, we successfully get the classifiers of good news followed by moving up trend in stock market and bad news, reversely. Therefore, given a newly released news article, according to the classifiers of news articles, we can predict whether the forthcoming trend of stock price is *up* or *down*, as well as how much is the fluctuation influenced by the news article based on a proposed Support Vector Regression method. Additionally, according to the contents of consecutive news articles, we dynamically choose sliding windows to abstract trends in stock prices. Finally, we realize the dynamic stock prediction based on analysis of online news articles.

This chapter is organized as follows. Section 6.2 details the methods for diagnosis of trends in stock price series and assignment of labels to news articles. In Section 6.3, we describe the preprocessing of news articles. Section 6.4 explains the procedure for classification. Section 6.5 explains the prediction of how much will the fluctuation of prices be affected based on the support vector regression method. We extend our method to dynamic choice of sliding window in Section 6.6. Section 6.7 provides the experimental results and compares them with those of existing methods. Finally, Section 6.8 concludes this chapter.

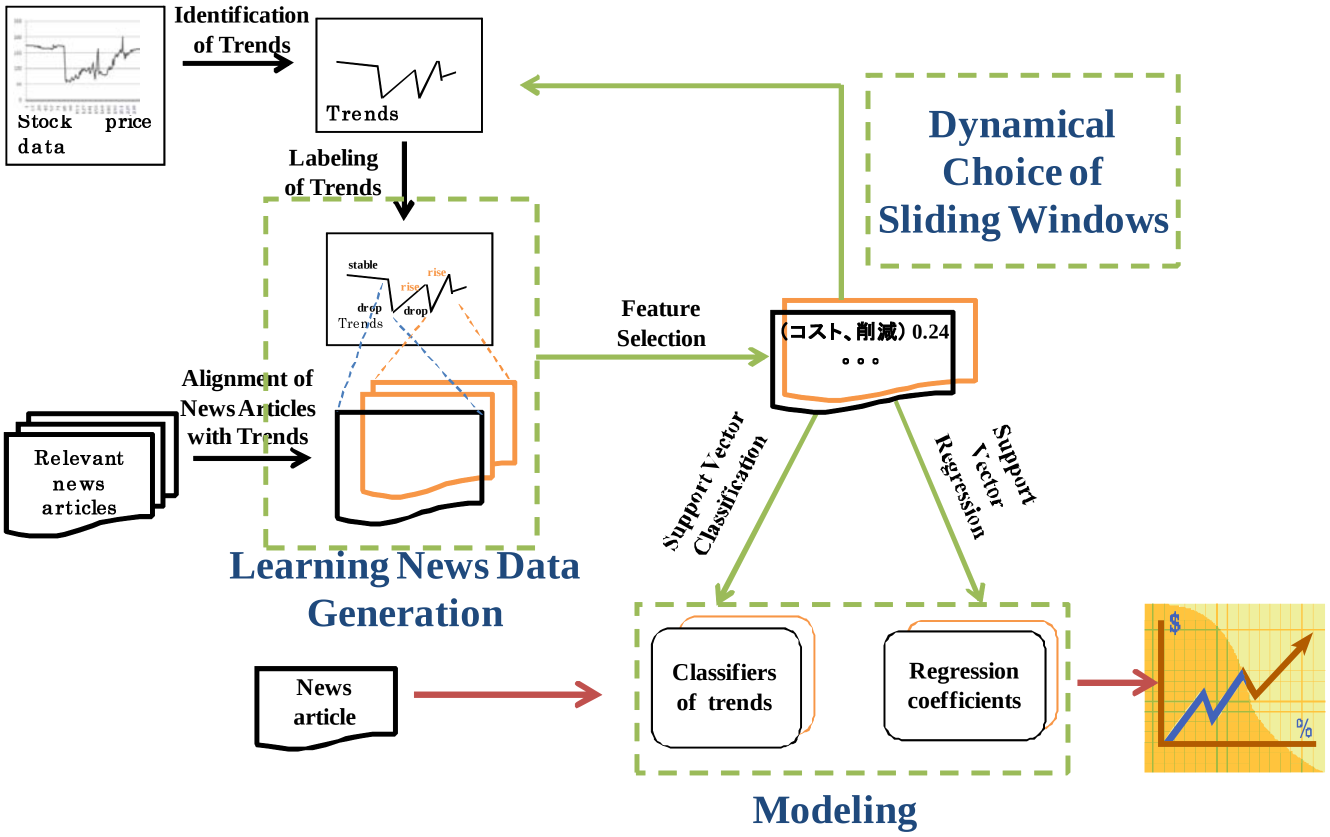


Figure 6.1: Overview of our proposed procedures for prediction of forthcoming trend in stock market based on analysis of news articles.

6.2 Generation of Labeled News Articles

We associate news articles with the trends of price series. We assign labels to news articles according to the trends of prices. In this section, we discuss two topics: the first one being how to represent the trends of price series, and the other is how to determine the influence window of news articles.

6.2.1 Representing Trends of Price Series

Most of the related work introduced in Chapter 2 identified trends by categorizing two consecutive data samples in the return series or the stock market volatility independently, and the results for categorization were not globally optimal. With a similar viewpoint to the re-description of price series in [?, ?], we define a trend as an interval in time by segmentation of continuous price series,

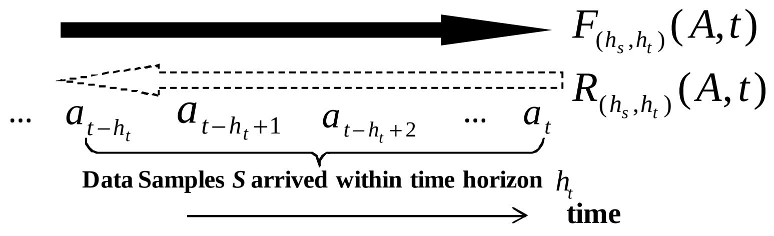

$F_{(h_s,h_t)}(A,t)$
$R_{(h_s,h_t)}(A,t)$
$\dots\ a_{t-h_t}\ \ a_{t-h_t+1}\ \ a_{t-h_t+2}\ \dots\ a_t$
Data Samples S arrived within time horizon h_t
time

Figure 6.2: Stock price series at time instant t.

and then cluster these segments into clusters whose segments are increasing, decreasing or remains relatively unaffected.

6.2.1.1 Identifying Trends

Although there are some well-known time series segmentation techniques which include Fourier coefficients [?] and parametric spectral models [?], stock series follow a random walk [?] representing very weak spectral signatures even locally. Therefore, we adopt the solution of piecewise linear segmentation for approximating the price series.

Referring to how to identify potentially important end points of linear segments, authors in [?] and [?] observed the error of piecewise linear regression of the price series, and used *t-test* to test end points which maximized the approximation error to make sure that there was a statistically significant difference. In addition to a top-down split phase in [?], authors in [?] executed a merging phase for the purpose of avoiding over-segmentation. However, because the spliting and merging phases are recursive processes, these methods cannot be applied to online segmentation of price series. More significantly, these methods are just useful for the intraday prediction of stock price.

On the other hand, authors in [?] advocated the piecewise linear representation of the price series should be in an up-down-up-down repetitive pattern (the zigzag shape) based on a wave theory of [?], and proposed an online segmentation and pruning method for identifying upper and lower end points in sliding windows based on a financial indicator, Bollinger Band Percent (%b). Empirical best choice of the sliding window's length and thresholds for identifying upper and lower points are provided.

Moving average method is helpful to smooth out short-term fluctuation and highlight long-term trends or cycles. Motivated from the *moving average* method, we observe trends of price data using sliding windows with the length of parameter h_t. However, the value of h_t can be specified by investors, or dynamically be determined by the contents of news articles discussed in Section **??**.

Considering the difficulty in choosing thresholds in [**?**], we observe rate of variance in data densities to characterize change points. Note that we estimate data densities taking account of the direction. Given a price series in Figure **??**, a_t is the price data at current time instant t. According to the size of sliding window h_t, we target on the data samples arrived in the time window $(t-h_t, t)$, denoted as S. We use a time-factored Gaussian Kernel Function

$$F_{(h_s,h_t)}(A,t) = \sum_{i=t-h_t}^{t} \left(\frac{i}{h_t}\right) \acute{K}_{(h_s)}(A,a_i). \tag{6.1}$$

where

$$\acute{K}_{(h_s)}(A,a_i) = \left(\frac{1}{\sqrt{2\pi h_s^2}}\right) \exp\left(-\frac{(A-a_i)^2}{2h_s^2}\right), \tag{6.2}$$

to estimate the density at a position A (a discrete possible value of stock prices) at time instant t, where h_s is the smooth parameter. We call $F_{(h_s,h_t)}(A,t)$ as forward data density. In order to observe the variance of data density, theoretically, we should compare the data density of data samples S with that of future data arriving in time window $(t, t+h_t)$. However, we have no information about future data. Therefore, we estimate the variance of data density. Firstly, we defined a reverse data density at position A as

$$R_{(h_s,h_t)}(A,t) = \sum_{i=t-h_t}^{t} \left(1-\frac{i}{h_t}\right) * \acute{K}_{(h_s)}(A,a_i). \tag{6.3}$$

Here, $R_{(h_s,h_t)}(A,i)$ is calculated from the same data samples S, but assuming that time was reversed and the data samples arrived in the reverse order. Then, the rate of variance in data density at position A can be estimated as

$$V_{(h_s,h_t)}(A,t) = \frac{F_{(h_s,h_t)}(A,t) - R_{(h_s,h_t)}(A,t)}{h_t}. \tag{6.4}$$

Note that if a great number of price data which are closer to value A arrived at the end of time window $(t-h_t,t)$, then $V_{(h_s,h_t)}(A,t)$ is positive. On the other hand, when a great number of price data which are closer to value A arrive at the beginning of the time window, then $V_{(h_s,h_t)}(A,t)$ is negative. If there is a repetitive fluctuation of price data towards value A, then $V_{(h_s,h_t)}(A,t)$ will almost be zero.

According to the sign of $V_{(h_s,h_t)}(A,t)$ for all possible positions, we can detect trend of data distribution at t by finding two fields:

Definition 1 (Coagulation Field R_t) *A coagulation field R_t at t is defined to be a set of positions A^* such that for any position A, we have $V_{(h_s,h_t)}(A,t) \leq V_{(h_s,h_t)}(A^*,t)$ and $V_{(h_s,h_t)}(A^*,t) \geq \delta \geq 0$.*

Definition 2 (Dissolution Field $\acute{R}_t$) *A dissolution field $\acute{R}_t$ at t is defined to be a set of positions A^* such that for any position A, we have $V_{(h_s,h_t)}(A,t) \geq V_{(h_s,h_t)}(A^*,t)$ and $V_{(h_s,h_t)}(A^*,t) \leq -\delta \leq 0$.*

Here, the value of threshold δ is set as the average rate of variance in data density at all possible positions. Therefore, the estimated data densities $\vec{s_t}$ at time t can be described as a change of data densities of direction $\overrightarrow{\acute{R}_t R_t}$ (a shift of data distribution from the dissolution field $\acute{R}_t$ to coagulation field $\acute{R}_t$). Then we compare $\overrightarrow{\acute{R}_t R_t}$ with $\overrightarrow{\acute{R}_{(t-1)} R_{(t-1)}}$ at time $(t-1)$: if there is any overlap between $\acute{R}_t$ and $R_{(t-1)}$, then we say that a_t is a change point.

Given an example in Figure **??**, we find change points $(C_1, C_2, C_3, C_4, C_5)$ with respect to $h_t = 5$, and consequently, we can abstract the trends of stock prices from t_1 to t_5 as shown in the dashed line. Consider a sliding window $(C_1, a_1, a_2, a_3, C_2)$ at t_2, there is a down-up-down pattern in the fluctuation of prices, but approaching to a_4, the values coagulate in the value field $[479, 490]$. We estimate the data density as $\overrightarrow{\acute{R}_{t_2} R_{t_2}}$, where R_{t_2} is $[479, 490]$, and $\acute{R}_{t_2}$ is $[470, 478]$. However, at next time point t_4 when a new data sample a_4 arrived, considering the sliding window $(a_1, a_2, a_3, C_2, a_4)$, we detect a dissolution field $\acute{R}_{t_4}$ equals to $[479, 490]$. Because there is an overlap between R_{t_2} and $\acute{R}_{t_4}$, then we regard a_4 as a change point.

After finding the end points of segments, we regard each piecewise fitting segment as a trend. The significance of a trend is defined by its regression statistics: slope (m) and correlation coefficient (R) The slope of the line indicates whether the trend is of interest. Very steep slopes characterize opportunities for maximizing profit by buying and selling when they are found in stock price data, while flatter segments, in general, recommend nothing. Likewise, high values of

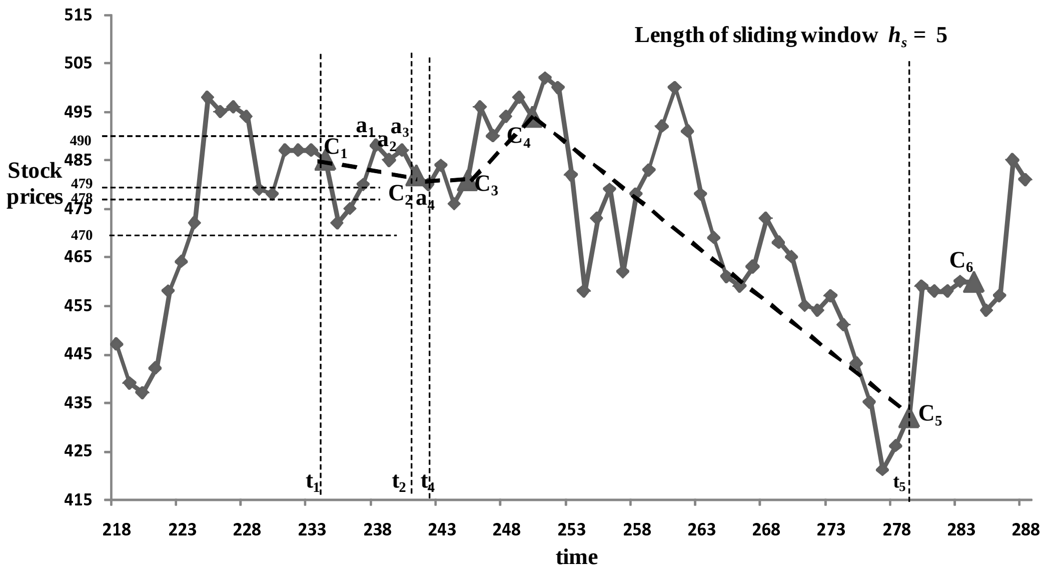


Figure 6.3: Example to identify trends of prices data.

R indicate strong confidence in the slope, and these trends are more trustworthy as a basis for decision.

6.2.1.2 Discreting Trends

The second step to re-describe price series is to assign labels to trends. These labels will be the basis for correlating trends with news articles. Initially, we implement a distance-based hierarchical agglomerative clustering [?] algorithm to automatically cluster stock trends. Because technical analysis of financial data assumes that the amplitude difference is more important than the time difference [?], the clustering algorithm is not necessary for the length of segments.

In detail, we do bottom-up merge of segments according to minimum group average distance; defined as

$$GAD(C_i, C_j) = \sum_{k \in C_i} \sum_{l \in C_j} \frac{dis_{ij}(k,l)}{|C_i||C_j|} \tag{6.5}$$

where $|C_i|$ and $|C_j|$ are the magnitudes of the clusters C_i and C_j respectively; $dis_{ij}(k,l) = \sqrt{(m_k - m_l)^2 + (R_k - R_l)^2}$ is Euclidean distance between the segments k and l which are included in C_i and C_j, respectively.

The clustering procedure terminates when the number of clusters equals to three: *up*, *down* and *steady*. Those segments in the cluster having the maximum average slope are labeled as *up*.

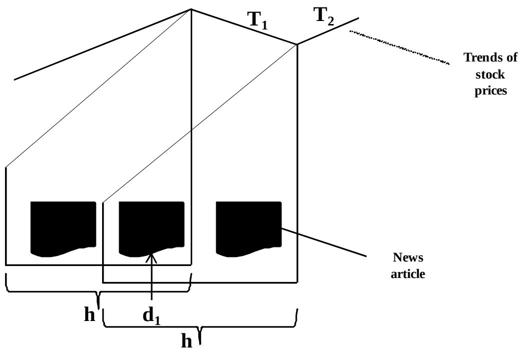


Figure 6.4: Alignment of news articles.

Similarly, those segments in the cluster having the minimum average slope are labeled as *down*. Segments in the remained cluster are labeled as *steady*.

6.2.2 Aligning Trends with News Articles

The news articles are labeled by alignment with trends in stock prices according to when the articles were released and when each trend was promoted. In order to learn models that might assist to suggest future behavior of price series, we associate news articles with trends in stock prices if its time stamp is h time units or less before the beginning of the trend. We define h as the size of influence window of news articles to stock prices.

An article may be associated with more than one trend. Article d_1 in Figure **??** is associated with both trends T_1 and T_2. While this may be contradictory, it is possible for d_1 to influence both trends T_1 and T_2. We can reduce the amount of overlap between time windows associated with trends by decreasing h. However, this can reduce the number of news articles that are associated with each trend, yielding fewer examples with which to train our prediction model. In Section **??**, we will give the evaluation of the prediction by changing the parameter h within $(0, h_t)$. We treat all articles aligned with a trend as having equivalent importance to a trend.

6.3 Preprocess of News Articles

In order to recommend investors profitable action signals (buy or sell), we focus on the trends of *up* and *down* in the forthcoming stock prices. Therefore, we propose a classification method to

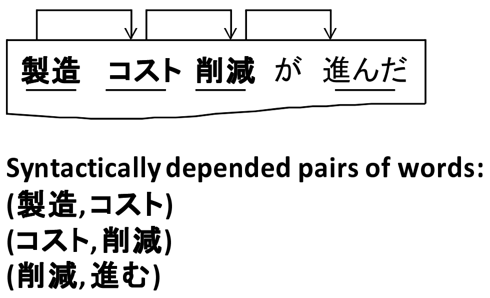

製造 コスト 削減 が 進んだ
Syntactically depended pairs of words:
(製造,コスト)
(コスト,削減)
(削減,進む)

Figure 6.5: An example of terms.

discriminate good news which will be followed by a moving up trend of stock price against the bad ones which will be followed by a going down trend, reversely.

In this section, we discuss about the preprocess of news articles before feeding news articles into classification in two steps: firstly, we choose features for representing news articles; and secondly, we propose a differentiation scheme for weighting terms which are frequently occurred in one class but rarely occurred in the other.

6.3.1 Feature Selection

All of the news articles labeled with *up* (*down*) are grouped together, and we select features for representing news articles. In the related work [?], the authors confirmed that there is a very stronger correlation between the rate of change in stock price and the parsed phases occurred in news articles rather than words. Therefore, we utilize terms which are pairs of syntactically depended words to characterize news articles. An example is illustrated in Figure **??**. We utilize the *cabocha* [1] to extract syntactical dependency among words, and choose noun and verb words for representing terms.

Furthermore, note that news articles may describe the same subject with different vocabulary: it is also necessary to capture semantic coherence between words. In our method, we employ the Latent Semantic Analysis (LSA) method [?]. LSA is a technique that discovers the salient semantic relationships between words by representing the original word-by-document matrix in a low dimensional linear combination of orthogonal variables. A matrix decomposition (i.e., Singular

[1]htt://chasen.org/taku/software/cabocha

Value Decomposition (SVD)) plays a pivotal role to generate a large number of orthogonal singular factors from an inverted index matrix. A small number of the most important singular factors are then selected to approximate the covariance of the original inverted matrix. Then a word is represented as a weighted vector with respect to the singular factors. A hierarchical agglomerative clustering [?] is employed to group words. Therefore, when we count the statistics of terms, we consider that the words in the same semantic group are the same.

After finding out the terms, each article in a particular class (*up* or *down*) is represented by a normalized vector-space model:

$$d = (w_1, w_2, ...w_n) \tag{6.6}$$

where the element w_T corresponds to the score of term T in the article d, and for the initialization, it is calculated by the standard *tf∗idf* scheme:

$$w_T = tf_{d,T} \times \log \frac{N}{df_T} \tag{6.7}$$

where $tf_{d,T}$ is the frequency of the term T in the article d; df_T is the number of articles containing the term T; and N is the total number of articles labeled according to a particular class (*up* or *down*).

6.3.2 Filtering and Weighting Scheme

We are interested in filtering out terms which frequently occur in one class of articles but rarely occur in the other. Incremental K-means is used for splitting the weighted articles of each class into two clusters. The centroid of the cluster C_i is defined as:

$$C_i = \frac{1}{|N_i|} \sum_{d \in D_i} d \tag{6.8}$$

where N_i is the set of articles within the cluster C_i and $|D_i|$ is the number of articles in this set. The similarity between the article d_i and the centroid C_j is determined by the cosine measure which is recommended by most researchers in document categorization:

$$\cos(d_i, C_j) = \frac{d_i * C_j}{|d_i| * |C_j|} \tag{6.9}$$

where $|d_i|$ and $|C_j|$ are the magnitude of the article d_i and the cluster C_j respectively.

The above procedure is taken out by both articles of *up* class and *down* class. Thus, on completion, four clusters of articles exist: two clusters of *up* class articles, C_{up_1} and C_{up_2}, and two clusters of *down* class articles, C_{down_1} and C_{down_2}. Using the following formula:

$$\cos(C_i, C_j) = \frac{C_i * C_j}{|C_i| * |C_j|} \tag{6.10}$$

we say that C_{up_i} (for $i = 1, 2$), which has the higher average similarity with the two C_{down_j} (for $j = 1, 2$), is insufficient to differentiate the *up* trend. Thus, it will be removed. Suppose we removed C_{up_1}, then just all of the news articles within C_{up_2} will be used in classification. Similarly, the C_{down_j} (for $j = 1, 2$), which has the higher average similarity with C_{up_i} (for $i = 1, 2$), will also be removed. Filtering is thus achieved.

In order to differentiate the terms appearing in one class of articles but not in the other class, we propose a new weighting scheme with respect to class relevance and discrimination by introducing two coefficients: inter-class discrimination coefficient (*CDC*) and intra-class similarity coefficient (*CSC*):

$$CDC = \frac{n_{i,T}}{N_T} \tag{6.11a}$$

$$CSC = \frac{n_{i,T}}{n_i} \tag{6.11b}$$

where $n_{i,T}$ is the number of articles in the i class (*up* or *down*) containing the term T; N_T is the total number of articles containing the term T; and n_i is the total number of different terms appearing within the i class.

Note that both *CDC* and *CSC* are always between 0 and 1. For *CDC*, the higher the value, the more powerful discrimination of the term T across classes. For *CSC*, the higher the value, the more the specified term is contained within the articles of the class. The weight of each term in each article is finally calculated as follows:

$$w(T, d) = tf_{T,d} \times CDC \times CSC \tag{6.12}$$

where $tf_{T,d}$ is the frequency of the term T in the article d.

6.4 SVM-based Classification of News Articles

We use Support Vector Machine (SVM) [?] to learn the patterns of terms in news articles that are highly associated with particular trends in stock prices. SVM obtains very accurate result in text classification, and outperforms any other techniques such as neural network and Naive Bayes. Since SVM is a binary classifier, we need to have a pair of classifiers. One classifier is responsible for checking if a piece of news will trigger the up event, while the other classifier is responsible for the event of down.

For the classification, we simply pass the newly collected news article to the pair of classifiers and decide which class the article should belong to. For example, an article signals an *up* event if the output value of the up classifier is positive. If the output values of both classifiers are negative, we will classify that article is not recommended, as it belongs to neither of the trends. If the output values of two classifiers are positive, then it certainly leads to be ambiguous, and therefore it is not classified as recommendation as well.

6.5 Regression-based Stock Price Prediction

In addition, we present an SVM regression method to predict stock movement. We do the process on each class of news articles respectively. We assume that the stock movement data are y_t $(t = 1, 2, \ldots, N)$, for the corresponding N time units. Consequently, we present articles influencing the prices as D_t. In our experiments on real data, D_t is choosen as daily close price data. Note that the weight of each term T_i in D_t is calculated by adding up all the weights of T_i in each news article. SVM regression function $f(D)$ is trained on $(D_1, y_1), \ldots, (D_N, y_N)$. It is tested on the samples $((D_{N+1}, y_{N+1}), \ldots, (D_M, y_M))$ incrementally to get the prediction prices $(\hat{y}_{N+1}, \ldots, \hat{y}_M)$. We use the Root Mean Square Error (RMSE) defined as

$$E = \sqrt{\frac{1}{(M-N)} \sum_{j=(N+1)}^{M} (y_j - \hat{y}_j)^2}. \tag{6.13}$$

to evaluate the error of prediction.

6.6 Extension to Dynamic Prediction of Stock Data

In the above design of our prediction precess, we label the news articles for learning according to the trends of stock prices in sliding windows. Therefore, selecting an appropriate length of sliding windows for identifying trends is a very important step for prediction. In the real world, investors may have some ideas for the choices of parameter h_t. However, considering the influence of some events to the stock market may be sensitive. In this case, it maybe necessary to analyze the trend of stock price in dynamic sliding windows. In this section, we extend our method to the dynamic determination of sliding windows.

We represent l-th article as $d_l = (w_{l1}, w_{l2}, \ldots, w_{lm})$, where w_{lk} is the weight of k-th term; m is the total number of terms. We define the *hotness* $H_{ij}(T_k)$ of term T_k in a period from j-th time unit to i-th time unit (where $i - j = h_t$) as

$$H_{ij}(T_k) = \frac{W_i(T_k) - W_j(T_k)}{W_i(T_k)} \tag{6.14}$$

where $W_i(T_k)$ is the sum of weight w_{lk} in all of the news articles released in the i-th time unit. Then we regard the term, which has the maximum hotness that also must be larger than the average hotness of all the m terms; as a *hot topic*. Then we reset h_t as the length of the period from the time unit when this hot topic occurred to that when the fluctuation of stock price became maximum. Until the hot topic disappears, h_t is reset to the initial value.

6.7 Experiment Results

In this section, we demonstrate several experimental results to evaluate our proposed methods. We carry out evaluations on two major processes. Firstly, we attempt to evaluate the effectiveness of our proposed method for grasping trend of stock prices based on estimating variance of data densities. Then we discuss the discriminating power of our ideas for representing news articles.

6.7.1 Experimental Setup

In our analysis we focused on a set of 120 stocks over the period from January, 2005 to December, 2008. The stocks are selected, based on the average amount of news about that stock. We generate the daily closing prices of these stocks, resulting in 986 or more data samples for every stock. The

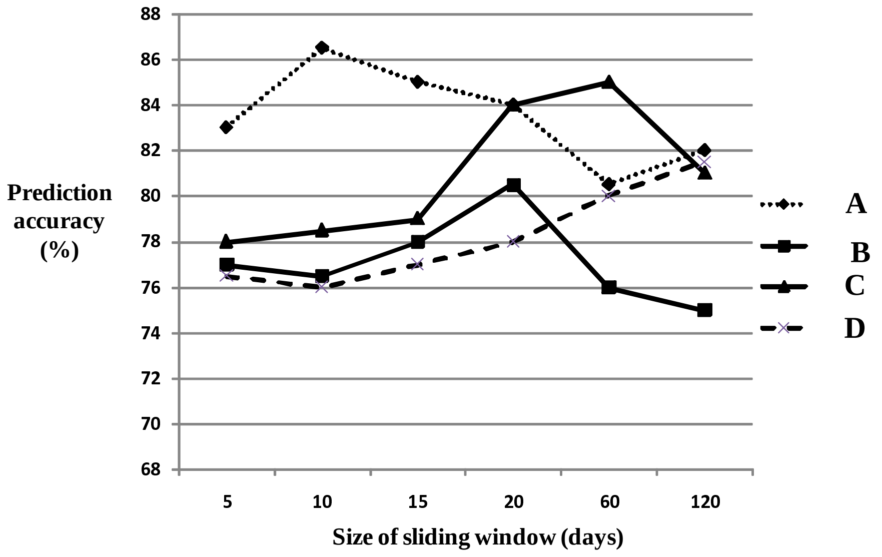


Figure 6.6: Effect of sliding window size on accuracy.

price data is re-described into trends as explained in Section ??. Our news collection contains 28042 or more news articles, gathered from the *Nikei Shimbun* over the same period of time. Each article contains a reference to at least one of the stocks that we are tracking.

We discuss two factors about the results of prediction: the direction (*up* or *down*), and the magnitude of the forthcoming trend in price series. We determine the direction of trend by classification. We evaluate the classification accuracy: the proportion of the number of news articles classified correctly to the number of total news articles for testing. On the other hand, we use the RMSE defined in Equation (??) to evaluate the power of prediction.

6.7.2 Experiments on Representation of Movement in Price Series

6.7.2.1 Choice of Sliding Window

In order to choose an effective news classifier, it is necessary to determine an effective value as the length of sliding window for grasping trends in stock prices. For this purpose, the news articles are labeled using various sliding window sizes. In Figure ?? the effect of increasing the sliding window size is investigated on the accuracy of classification in terms of four different companies. The most accurate results (i.e., those with the highest accuracy) for these companies are different. For the company A, we can see the best observation of movement in stock price is about two weeks. For

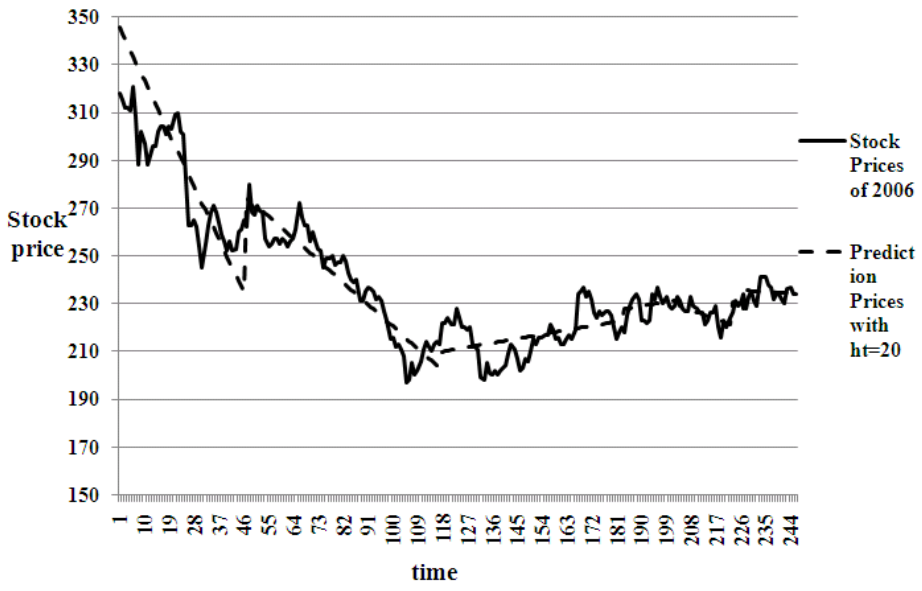
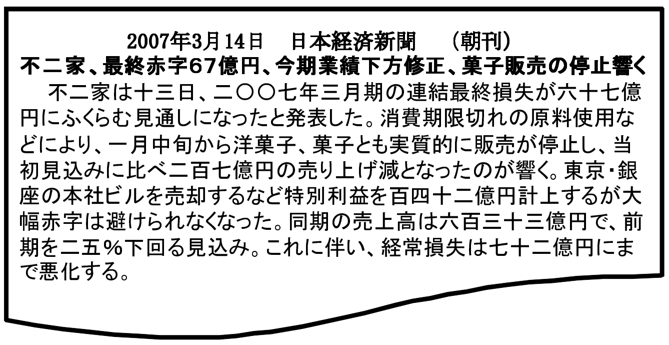


Figure 6.7: Real stock price of company E in 2006 and the prediction price with respect to $h_t = 20$.

the company D, the best choice is about half a year. The best values are about one month and three monthes in the cases of company B and company C, respectively. We consider that these results correspond to the type of each company's business. For example, we can see that there are many news reports about new products in daily use of company C before a new season comes in a year.

6.7.2.2 Choice of Influence Window

We assume that a news article affects stock price only after it becomes publicly available and then we do experiments for a set of alignments. Here, we test the results of average accuracy of classifications for all the stocks against the null hypothesis, that the results of the classification are the results of random guessing. Results of Chi-square hypothesis tests against the null hypothesis are shown in terms of z-scores. We do experiments on different window bounding offsets in days for the alignments. We find that the most significant classification results and best relative prediction accuracy occurred for alignment $h = 2$ days. Furthermore, we find that as the window of influence is extended, classification results become less and less significant. This suggests that we have found a strong correlation between news articles and the daily behavior of stock prices 2 days after news articles become publicly available.


2007年3月14日　日本経済新聞　（朝刊）

不二家、最終赤字67億円、今期業績下方修正、菓子販売の停止響く

　不二家は十三日、二〇〇七年三月期の連結最終損失が六十七億円にふくらむ見通しになったと発表した。消費期限切れの原料使用などにより、一月中旬から洋菓子、菓子とも実質的に販売が停止し、当初見込みに比べ二百七億円の売り上げ減となったのが響く。東京・銀座の本社ビルを売却するなど特別利益を百四十二億円計上するが大幅赤字は避けられなくなった。同期の売上高は六百三十三億円で、前期を二五%下回る見込み。これに伴い、経常損失は七十二億円にまで悪化する。

Figure 6.8: Big news released on March 14, 2007, which trigger sudden drop in stock market.

6.7.2.3 Dynamic Choice of Sliding Windows

We illustrate an example to evaluate the effectiveness of our mechanism of dynamic choice of sliding window size. From the movement of stock market of company E in 2006 as illustrated in Figure **??**, we can grasp the trends in stock price correctly with the parameter $h_t = 20$ according to the method proposed in Section **??**. However, after the release of news article as shown in Figure **??** at March 14, 2007, stock prices of company E dropped suddenly as the blue line as shown in the Figure **??**. The words like ”売却 ”, ”損失 ”appeared frequently in news articles for a consecutive period. According to our proposed dynamic mechanism in Section **??**, we detect these events, and find the first maximum drop of price on March 20, which is 5 days after the release of the event. Then our method changes to observe the movement of price series with respect to $h_t = 5$. In Figure **??**, the green line is the prediction of price after the adjustment of sliding windows; however, the red line is the prediction results of the previous parameter. Here, we can see that the accuracy is higher than the previous one. Figure **??** shows the ratio of RMSE with different value of h_t to the real price data. We can see that the dynamic adjustment of sliding windows is effective.

6.7.2.4 Comparison with Related Work

We compared the accuracy of classification and RMSE error of our proposed method with those of method based on $\%b$ series proposed in [?] in Figure **??** and Figure **??**. From the results, we can see that our proposed method outperforms the traditional method by yielding about 78% accuracy in predicting the magnitude of movement and 83% accuracy for the direction of movement in stock

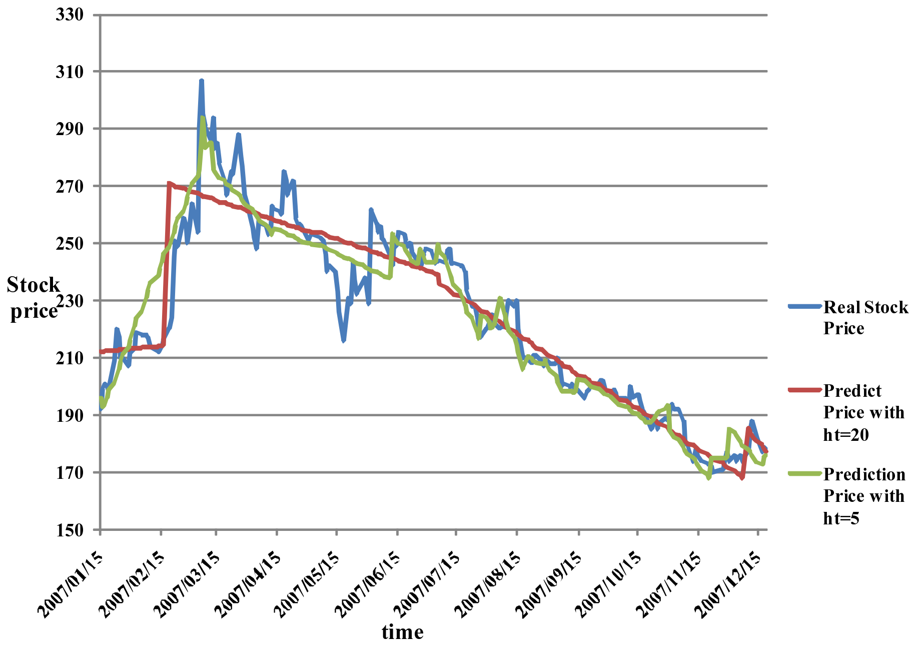


Figure 6.9: Prediction price after dynamic adjustment of sliding window for observing stock market.

price.

6.7.3 Experiments on Preprocessing News Articles

In order to evaluate the effectiveness of our preprocessing of news articles, we compare the results of our method with other two methods. One just uses single words for characterizing news articles, and the other presents news articles as term vectors in our method but weighted with tf*idf values. As shown in Table **??**, our proposed method achieves the best performance.

6.8 Summary

We proposed classification and regression methods to forecast stock prices based on news articles. In our proposed framework, it is flexible for investors to grasp short-term or long-term trends of stock prices by identifying trends in sliding windows. Additionally, we proposed ideas to determine h_t dynamically according to the contents of news article sequences. Moreover, our proposed ideas for processing news articles also contribute to good performances. Experiments on real stock data and news articles are performed. Our results yield about 83% accuracy in predicting the

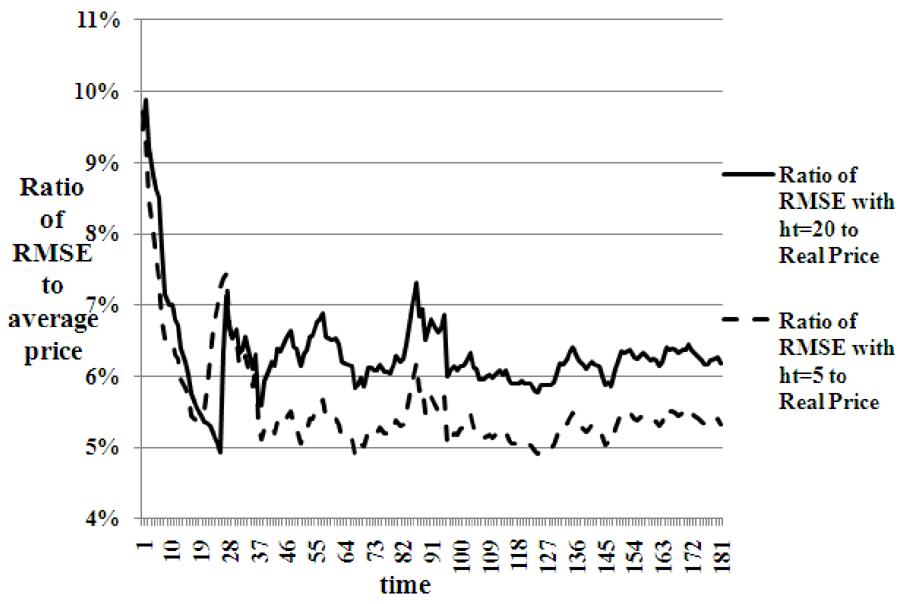


Figure 6.10: Comparison of RMSE of prediction after dynamic adjustment of sliding window.

Table 6.1: Experiment on different preprocessing method of news articles

	Feature of Single Word		Terms Weighted with tf*idf		Our Proposed Method	
	up	*down*	*up*	*down*	*up*	*down*
Precision	0.989	0.885	0.923	0.914	0.972	0.925
Recall	0.794	0.733	0.912	0.847	0.930	0.922
F-value	0.790	0.818	0.920	0.879	0.980	0.917
Accuracy	0.787		0.904		0.952	

magnitude of forthcoming trend and 83% for the direction of forthcoming trend in stock price. The mechanism for dynamically choosing sliding windows for identification of trends is also proven to be effective for higher prediction of stock prices.

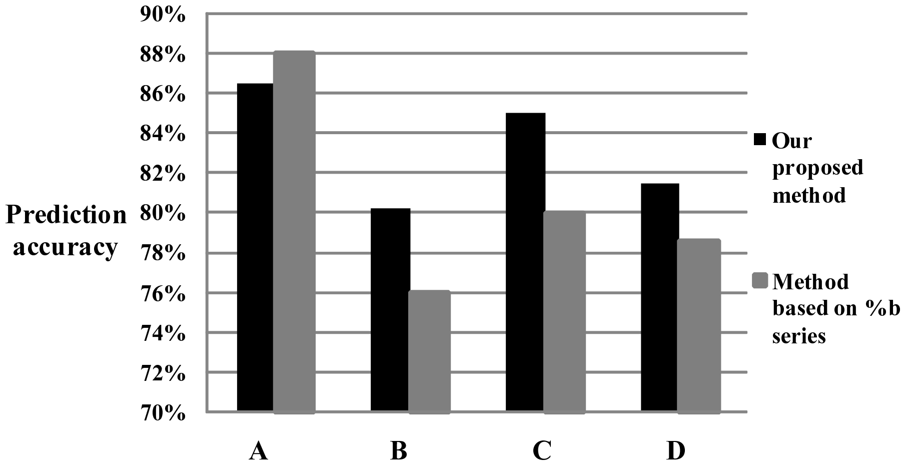


Figure 6.11: Comparison of prediction accuracy.

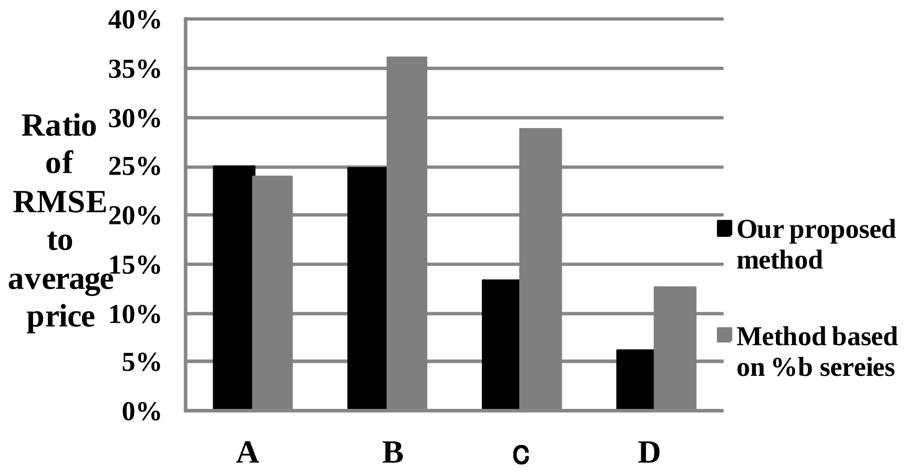


Figure 6.12: Comparison of RMSE ratio exhibited by companies.

Chapter 7

CONCLUSIONS

Techniques of data stream mining discover up-to-date patterns which are invaluable for strategic decisions, but the mining process has to be done accurately, quickly with limited computation resources, as well as adaptively considering concept drifts.

Besides the needs for knowledge discovery from single data sources, in the era of information overload, it becomes more and more important to mine interesting correlations among multiple streaming cross-domain data sources to support people's decision making. The objective of this dissertation is do further explorations of mining correlations among multiple streaming cross-domain data sources. In terms of different types of streaming data, we categorize the correlations into two basic correlations: *discrete correlation* and *continuous correlation.* The discrete correlation corresponds to the applications assuming that the data samples are independent with each other, and continuous correlation is defined as the cross-relationship among the multiple continuous time-series data streams. We propose novel and efficient algorithms to discover these two kinds of correlations in different applications of single data source. Moreover, this dissertation extends the study to mine complex correlations in cross-domain data sources by combining the investigations of these two kinds of basic correlations, taking the example of dynamic stock prediction based on analysis of online news articles.

In the applications of discovering discrete correlation from streaming quantitative transaction data, we proposed efficient algorithms for discovering *Quantifiable Correlated Patterns* in Chapter 3. The result is the first algorithm that we discover correlations in streaming transaction data with only one single scan of data. On one hand, the property of correlation in our results is very useful means for the discovery of rarely occurring but important incidents, such as network intrusions; on the other hand, the quantitative ratio associations among the itemsets are also discovered. The stronger expressive power of quantitative associations allows us to obtain much richer knowledge.

In Chapters 4 and 5, we targeted to the massive streaming time-series data to discover continuous correlations. In order to discover the cross-relationship among the streams, as one of the solutions, we proposed was to incrementally learn the inherent correlations among streams and find the underlying hidden variables in each correlated-cluster. This algorithm satisfies the requirement of scalability on the number of streams, full automation and adaptivity taking into account of the data evolution in streaming data. We proved that the discovered hidden variables can be used to immediately spot potential anomalies, and do efficient forecasting in sensor network. Another solution is to support flexible timeline clustering techniques to the massive data streams. This method realizes group streams in terms of arbitrary interested periods of time. Therefore, it is possible to track the cluster transaction at multi-solutions.

In Chapter 6, we proposed a framework for investigating the correlation among news articles and the evolving stock prices. We realized the automatic and dynamic classification of news articles for predicting forthcoming trends of stock prices. Given a news article, we decide whether it is a piece of good news followed by a moving up trend in stock market, or a piece of bad news reversely. Furthermore, we proposed the regression method to estimate how much would the fluctuation of stock price be influenced by the news article. In our proposed framework, we combine the investigations of the discrete correlation as well as the continuous correlation, in order to achieve dynamic analysis of the correlation between online news articles and stock price series.

Future work There are many researches to be done to enhance the applicability and efficiency of the methods in this dissertation as listed in Table **??**.

Meanwhile, in many business applications, business people cannot effectively take over and interpret the identified patterns for business use [**?**, **?**]. Therefore, it is also important to develop effective approaches for discovering patterns which are not only technical significant, but also satisfy business expectations, and further indicate the possible actions that can be explicitly taken by business people.

Table 7.1: Main contribution, future works and possible applications.

Contribution	Future work	Possible Applications
Quantifiable correlated patterns mining	1. Other quantities to estimate correlation coefficient, such as Bayesian approach	Web usage analysis, Mobile communication analysis
Correlated-clusters mining	1. Modeling and tracking transitions of clusters	Sensor network monitoring
Flexible timeline clustering	1. Backward segmentation for improving the accuracy of approximation with an overall view of the whole time series 2. Modeling and tracking transitions of clusters	Sensor network monitoring
Dynamic prediction of stock prices based on analysis of news articles	1. Explanation of similarity among different stock prices 2. Discovery of abnormal behaviors of stocks comparing to the key trends in the whole market	Information integration

Bibliography

[1] Cranor, C., Johnson, T., Spataschek, O. and V. Shkapenyuk (2003) "Gigascope: a stream database for network applications". *In proceedings of the 22nd ACM International Conference on Management of Data (SIGMOD)*, San Diego, pp. 647-651.

[2] Abadi, D.J., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N. and Zdonik, S. (2003) "Aurora: a new model and architecture for data stream management". *In The International Journal on Very Large Data Bases (VLDB Journal)*, Vol. 12, No. 2, pp. 120-139.

[3] Rosset, S., Murad, U., Neumann, E., Idan, Y. and Pinkas, G. (1999) "Discovery of fraud rules for telecommunications - challenges and solutions". *In proceedings of the fifth ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, San Diego, pp. 409-413.

[4] Zhu, Y.Y and Shasha, D. (2002) "StatStream: Statistical monitoring of thousands of data streams in real time". *In proceedings of the 28th International Conference of Very Large Data Bases (VLDB)*, Hong Kong, pp. 358-369.

[5] Zhu, Y.Y and Shasha, D. (2003) "Efficient elastic burst detection in data streams". *In proceedings of the ninth ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Washington, D.C., pp. 336-345.

[6] Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J. (2002) "Models and issues in data stream systems". *In proceedings of 21st SIGMOD-SIGACT-SIGART Symposium on Principles of database systems (PODS)*, New York, pp. 1-16.

[7] Aggarwal. C.C. (2007) "Data Streams: Models and Algorithms". Springer Berlin Heidelberg.

[8] Gaber, M., Zaslavsky, A. and Krishnaswamy, S. (2005) "Mining Data Streams: A Review". *SIGMOD Record*, Vol. 34, No. 2. pp. 18-26.

[9] Jiang, N. and Gruenwald, L. (2006) "Research Issues in Data Stream Association Rule Mining". *SIGMOD Record*, Vol. 35, No. 1, pp. 14-19.

[10] Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Dtar, M., Manku, G., Olston, C., Rosenstein, J. and Varma, R. (2003) "Query Processing, Approximation, and Resource Management in a Data Stream Management System". *In proceedings of Conference on Innovative Data System Research*, Asiloma, pp. 245-256.

[11] Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F. and Shah, M. (2003) "TelegraphCQ: Continuous Data Flow Processing for an Uncertain World". *In proceeding of Conference on Innovative Data System Research*, Asiloma, pp. 269-280.

[12] Aggarwal, C.C., Han, J. Wang, P., Yu, S. (2003) "A Framework for Clustering Evolving Data Streams". *In proceedings of the 19th International Conference on Very Large Data Bases (VLDB)*, Berlin, Volumn 3, pp. 81-92.

[13] Wang, H., Fan, W., Yu, P. and Han, J. (2003) "Mining Concept-Drifting Data Streams using Ensemble Classifiers". *In proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Washington, D.C., pp. 226-235.

[14] Manku, G.S. and Motwani, R. (2002) "Approximate frequent counts over data streams". *In proceedings of International Conference on Very Large Data Bases (VLDB)*, pp. 346-357.

[15] Beyer, K., Goldstein, J., Ramakrishnan, R. and Shaft, U. (1999) "When is nearest neighbors meaningful?". *In proceedings of International Conference on Database Theory*, pp. 217-235.

[16] Sakurai, Y., Papadimitriou, S., and Faloutsos, C. (2005) "BRAID: Streaming mining through group lag correlations". *In proceedings of 2005 ACM International Conference on Management of Data (SIGMOD)*, pp. 599-610.

[17] Yeh, M.Y., Dai, B.R. and Chen, M.S. (2007) “Clustering over Multiple Evolving Streams by Events and Correlations”. *In IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 10, pp. 1349-1362.

[18] Beringer, J. and Hullermeier, E. (2005) “Online Clustering of Parallel Data Streams”. *In Journal of Data and Knowledge Engineering*, Vol. 58, No. 2, pp. 180-204.

[19] Rodrigues, P.P., Gama, J. and Pedroso, J.P. (2006) “ODAC: Hierarchical Clustering of Time Series Data Streams”. *In proceedings of the sixth SIAM International Conference of Data Mining*, pp. 499-503.

[20] Fan, W., Watanabe, T. and Asakura, K. (2009) “Mining Interesting Ratio Patterns over a Stream Sliding Window”, *In proceedings of International Conference on Machine Learning and Applications (ICMLA)*, pp. 731-734.

[21] Fan, W., Watanabe, T. and Asakura, K. (2009) “Ratio Rules Mining in Concept Drifting Data Streams”, *In proceedings of International Conference on Machine Learning and Data Analysis (ICMLDA)*, Volume. II, pp. 809-814.

[22] Fan, W., Watanabe, T. and Asakura, K. (2010) “Mining underlying correlated-clusters in high-dimensional data streams”, *In International Journal of Social and Humanistic Computing*, Vol. 1, No. 3, pp. 282-299.

[23] Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2008) “An Incremental PCA for Stream Analysis Based on NLMS Adaptive Filter”, *In proceedings of* **電気関係学会東海支部連合大会**, O-511.

[24] Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2009) “Generalized Regression Measure for Local Correlation Tracking in Evolving Data Streams”, *In proceedings of DIEM Forum*, E6-3.

[25] Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2009) “A System for Mining Correlations among Multiple Evolving Data Streams”, *In the 23th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI)*, pp. 1-4.

[26] Fan, W., Watanabe, T. and Asakura, K. (2008) “A framework for flexible clustering of multiple evolving data streams”, *International Journal of Advanced Intelligence Paradigms*, Vol. 1, No. 2, pp. 178-195.

[27] Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2008) “Clustering over Evolving Data Streams Based on Online Recent-biased Approximation”, *In proceedings of Pacific Rim Knowledge Acquisition Workshop*, LNAI 5351 (Springer), pp. 273-287.

[28] Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2008) “Clustering on Multi-granularity Temporal Trend of Multiple Evolving Data Streams”, *In proceedings of Workshop on Informatics*, pp. 215-220.

[29] 范 薇, 渡邉 豊英, 朝倉 宏一. (2010) “Multi-solution Trend Analysis of Stock Price Change by Textual Information”, *In proceedings of FIT*, pp.163-166.

[30] 范 薇, 渡邉 豊英, 朝倉 宏一. (2010) “連続的時系列データストリームからの知識発見”, *In Technical Report of IEICE: DE2010*, pp. 7-12.

[31] Fan, W., Watanabe, T. and Asakura, K. (2010) “News-sensitive Multi-solution Prediction of Stock Price”, *In proceedings of Information Processing Society of Japan 50th Anniversary, 72nd National Convention of IPSJ*, pp. 1-257-1-258.

[32] Guha, S., Mishra, N., Motwani, R. and Callaghan, L. (2000) “Clustering data streams”. *In proceedings of 41st Annual Symposium on Foundations of Computer Science*, pp. 359-366.

[33] Chang, J.K. Lee, W.S. and Zhou, A. (2003) “Finding Recent Frequent Itemsets Adaptively over Online Data Streams”. *In proceeding of 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 487-492.

[34] Aggarwal, C.C., Han, J., Wang J. and Yu. P. (2004) “A framework for projected clustering of high dimensional data streams”. *In proceeding of 30th International Conference on Very Large Data Bases (VLDB)*, pp. 852-863.

[35] Chang, J. and Lee, W. (2004) “ A sliding window method for finding recently frequent itemsets over online data streams”. *In Journal of Information Science and Engineering*, Vol. 20, No. 4, pp. 753-762.

[36] Chi, Y., Wang, H. Yu, P. and Richard, R. (2004) “Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window”. *In proceedings of IEEE International Conference on Data Mining*, pp. 59-66.

[37] Lin, C., Chiu, D., Wu, Y. and Chen, A. (2005) “Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window”. *In proceedings of SIAM International Conference on Data Mining*, pp. 68-79.

[38] Agrawal, R. and Srikant, R. (1994) “Fast Algorithms for Mining Association Rules in Large Databases”. *In proceedings of 20th International Conference on Very Large Data Bases*, pp. 487-499.

[39] Xiong, H., Tan, P.N. and Kumar, V. (2006) “Hyperclique Pattern Discovery”. *In Journal of Data Mining and Knowledge Discovery*, Vol.13, No.2, pp. 219-242.

[40] Brin, S., Motwani, R. and Silverstein, R. (1997) “Beyond Market Basket: Generalizing Association Rules to Correlations”. *In proceedings of ACM International Conference on Management of Data (SIGMOD)*, pp. 265-276.

[41] Lee, Y.K., Kim, W.Y., Cai,Y.D., and Han, J. (2003) “CoMine: Efficient Mining of Correlated Patterns”. *In proceedings of 3rd IEEE International Conference on Data Mining*, pp. 581-584.

[42] Kim, W.Y., Lee, Y.K. and Han, J. (2004) “CCMine: Efficient Mining of Confidence-closed Correlated Patterns”. *In proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pp. 569-579.

[43] Omiecinski, E. (2003) “Alternative Interest Measures for Mining Associations”. *In IEEE Transactions on Knowledge and Data Engineering*, Vol.15, No. 1, pp.57-69.

[44] Ma, S. and Hellerstein, J.L. (2001) "Mining Mutually Dependent Patterns". *In proceedings of 2001 IEEE International Conference on Data Mining*, pp. 409-416.

[45] Ke, Y., Cheng, J. and Ng, W. (2006) "Mic framework: An information-theoretic Approach to Quantitative Association Rule Mining". *In proceedings of the 22th International Conference on Data Engineering*, pp. 112-112.

[46] Ke, Y., Cheng, J. and Ng, W. (2008) "Correlated Pattern Mining in Quantitative Databases". *In ACM Transaction on Database Systems*, Vol. 33, No. 3. August, pp. 1-44.

[47] Xiong, H., Shekhar, S., Tan, P.N. and Kumar, V. (2006) "TAPER: A Two-step Approach for All-strong-pairs Correlation Query in Large Databases". *In IEEE Transactions on Knowledge and Data Engineering*, Vol, 18, No. 4, pp. 493-508.

[48] Srikant, R. and Agrawal, R. (1996) "Mining Quantitative Association Rules in Large Relational Tables". *In proceedings of 1997 ACM SIGMOD International Conference on Management of Data*, pp. 1-12.

[49] Wang, K., Tay, S.H. and Liu, B. (1998) "Interestingness-based Interval Merger for Numeric Association Rules". *In proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 121-128.

[50] Fukuda, T., Morimoto, Y., Morishita, S. and Tokuyama, T. (2001) "Data Mining with Optimized two-dimensional association rules". *In ACM Transations on Database Systems*, Vol. 26, No. 2, pp. 179-213.

[51] Zhang, H., Padmanabhan, B. and Tuzhilin, A. (2004) "On the Discovery of Significant Statistical Quantitative Rules". *In proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 374-383.

[52] Chen, Z.Y. and Liu, G.H. (2005) "Quantitative Association Rules Mining Methods with Privacy Preserving". *In proceedings of the 6th International Conference on Parallel and Distributed Computing Applications and Technologies*, pp. 910-912.

[53] Korn, F., Labrinidis, A., Kotidis, Y. and Faloutsos, C. (2000) "Quantifiable data mining using ratio rules", *In Journal of Very Large of Data Base*, Vol. 8, pp. 254-266.

[54] Yan, J., Yang, Q., Zhang, B., Cheng, Q. and Chen Z. (2006) "Mining Adaptive Ratio Rules from Distributed Data Sources", *In Journal of Data Mining and Knowledge Discovery*, Vol. 12, No. 2-3, pp. 249-273.

[55] 濱本 雅史, 北川 博之 (2006) "サポートと確信度をもとにした比率規則による線形関係抽出", 情報処理学会論文誌, Vol. 47, No. 19, pp. 54-71.

[56] Demiriz, A., Cihan, A. and Kula, U. (2009) "Analyzing Price Data to Determine Positive and Negative Product Associations", *In proceedings of the 16th International Conference on Neural Information Processing*, pp. 846-855.

[57] Hidlber, C. (1999) "Online Association Rule Mining". *In proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pp. 145-156.

[58] Giannella, C. Han, J.W., Pei, J. Yan, X.F. and Yu, P.S. (2003) "Mining Frequent Patterns in Data Streams at Multiple Time Granularities". *In proceedings of Next Generation Challenges and Future Directions*, pp. 486-491.

[59] Li, H.F., Lee, S.Y. and Shan, M.K. (2004) "An Efficient Algorithm for Mining Frequent Itemsets over the Entire History of Data Streams". *In proceedings of International Workshop on Knowledge Discovery in Data Streams.*

[60] Yang, L. and Sanver, M. (2004) "Mining Short Association Rules with One Database Scan". *In proceedings of International Conference on Information and Knowledge Engineering.*

[61] Han, J., Pei, J. and Yin, Y. (2000) "Mining frequent patterns without candidate generation", *In proceedings of the 2000 ACM International Conference on Management of Data (SIGMOD)*, pp. 1-12.

[62] Perlman, E. and Java, A. (2003) "Predictive Mining of Time Series Data in Astronomy", *In Journal of Astronomical Data Analysis Software and Systems XII ASP Conference Series*, Vol. 295, pp. 431-434.

[63] Artae, M., Jogan, M. and Leonardis, A. (2002) "Incremental PCA for On-line Visual Learning and Recognition". *In proceedings of 16th International Conference on Pattern Recognition*, Quebec, pp. 781-784.

[64] Guha, S., Gunopulos, D. and Koudas, N. (2003) "Correlating synchronous and asynchronous data streams", *In proceedings of ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pp. 529-534.

[65] Weng, J., Zhang, Y. and Hwang, W.S. (2003) "Candid Covariance-free Incremental Principal Component Analysis". *In IEEE Transactions of the Pattern Analysis and Machine Intelligence*, Vol. 15, No. 8, pp. 1034-1040.

[66] O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S. and MOtwani, R. (2002) "Streaming-Data Algorithms for High-Quality Clustering", *In proceedings of the 20th International Conference on Data Engineering*, IEEE, Boston, USA, pp. 685-694.

[67] Yang, J. (2003) "Dynamic Clustering of Evolving Streams with a Single Pass", *In proceedings of the 21st International Conference on Data Engineering*, IEEE, Bangalore, India, pp. 695-697.

[68] Rafiei, D. and Mendelzon, A.O. (1998) "Efficient Retrieval of Similar Time Sequences Using DFT", *In proceedings of the 5th International Conference of Foundations of Data Organization*, Kobe, Japan, pp. 249-257.

[69] Chan, F.K.-P., Fu, A.W.-C. and Yu, C. (2003) "Haar Wavelets for Efficient Similarity Search of Time-Series: With and without Time Warping", *In IEEE Transactons on Knowledge and Data Engineering*, Vol. 15, No. 3, pp. 686-705.

[70] Kanth, K.V.R., Agrawal, D. and Singh, A.K. (1998) "Dimensionality Reduction for Similarity Searching in Dynamic Databases", *In proceedings of ACM SIGMOD International Conference on Management of Data*, ACM, Washington, USA, pp. 166-176.

[71] Keogh, E., Chu, S., Hart, D., Pazzani, M. (2003) "Segmenting Time Series: A Survey and Novel Approach", *In Data Mining in Time Series Databases*, second edition, World Scientific, Singapore, pp. 1-21.

[72] Appel, U. and Brandt, A.V. (1983) "Adaptive Sequential Segmentation of Piecewise Stationary Time Series", *In Information Science*, Vol. 29, No. 1, pp. 27-56.

[73] Klein, F and Prestbo, J. A. (1974) "News and the Market", Chicago: Henry Regenry.

[74] Peramunetilleke, D and Wong, R. K (2001) "Currency Exchange Rate Forecasting from News Headlines", *In proceedings of 13th Australasian Database Conference*, pp. 131-139.

[75] Antweiler,W. and Frank, M. Z. (2004) "Is All That Talk Just Noise? The Information Content of Internet Stock Message Boards", *In The Journal of Finance*, Vol. 59, No. 3, pp. 1259-1294.

[76] Choudhury, M.D., Sundaram, H., John, A. and Seligmann, D.D. (2008) "Can blog communication dynamics be correlated with stock market activity?", *In proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pp. 55-60.

[77] Pan, Q., Cheng, H., Wu, D., Yu, J. X., Ke, Y. P. (2010) "Stock Risk Mining by News", *In proceeding of 25th Australasian Database Conference*, pp. 179-188.

[78] Seo, Y. W., Giampapa, J. and Sycara, K. (2002) "Text Classification for Intelligent Portfolio Management", *In Technical report CMU-RI-TR-02-14*.

[79] Wuthrich, B. and Zhang, J. (1999) "Text Processing for Classification.", *In Journal of Computational Intelligence in Finance*, Vol. 7, No. 2, pp. 6-22.

[80] Wuthrich, B. (1995) "Probabilistic Knowledge bases", *In IEEE transactions on Knowledge and Data Engineering*, Vol. 7, No. 5, pp. 691-698.

[81] Lavrrenko, V., Schmill, M, Lawire, D., Ogilvie, P., Jensen, D. and Allan, J. (2000) "Language Models for Financial News Recommendation", *In proceedings of ninth international conference on Information and knowledge management*, pp. 389-396.

[82] Lavrrenko, V., Schmill, M, Lawire, D., Ogilvie, P., Jensen, D. and Allan, J. (2000) “Mining of Concurrent Text and Time Series”, *In proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*, pp. 37-44.

[83] Fung, G. Yu, J. X. and Lam, W. (2002) “News Sensitive Stock Trend Prediction.”, *In proceedings of 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pp. 481-493.

[84] Robertson, C., Geva, S. and Wolff, R. C. (2007) “Can the Content of Public News be Used to Forecast Abnormal Stock Market Behavior?”, *In proceedings of Seventh IEEE International Conference on Data Mining*, pp. 637-642.

[85] Gidofalvi, G.: “Using News Articles to Predict Stock Price Movement.”, In Project Report, Department of Computer Science and Engineering, University of California, San Diego. http://www-cse.ucsd.edu/users/elkan/254spring01/gidofalvirep.pdf, 2001-06-15.

[86] Joachims, T. (1998) “Text Categorization with Support Vector Machines: Learning with Many Relevant Features”, *In proceedings of European Conference on Machine Learning*, pp. 137-142.

[87] Mittermayer, M. A. (2004) “Forecasting Intraday Stock Price Trends with Text Mining Techniques”, *In proceedings of 37th Annual Hawaii International Conference on System Sciences*, pp. 10.

[88] 和泉 潔, 後藤 卓, 松井 藤五郎. (2008) “テキスト情報を用いた金融市場分析の試み.”, 第 22 回人工知能学会全国大会.

[89] Motawani, R., Cohen, E. Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Ullman, J.D. and Yang, C. (2001) “Finding Interesting Association without Support Pruning”, *In IEEE Transactions on Knowledge and Data Engineering (speical issue)*, Vol. 13, No. 1, pp. 64-78.

[90] Haykin, S. (1992) “Adaptive Filter Theory”, Prentice Hall.

[91] Chakrabarti, K. and Mehrotra, S. (2000) “Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces”. *In proceedings of 26th International Conference on Very Large Data Bases*, Cairo, pp. 89-100.

[92] Ravi Kanth, K.V, Agrawal, D. and Singh, A. (1998) “Dimensionality reduction for similarity searching dynamic databases”, *In ACM SIGMOD Record*, Vol. 27, Issue 2, pp. 166-176.

[93] Zhang, M., Hsu, W. and Lee, M. (2007) “Mining Prevalence-based Ratio Patterns”, *In proceedings of Tools with Artificial Intelligence of IEEE Conference*, pp. 140-147.

[94] Faloutsos, C., Rangantathan, M. and manalopoulos, Y. (1994) “Fast Subsequence Matching in Time-Series Databases”, *In proceedings of ACM SIGMOD International Conference on Management of Data*, pp, 419-429.

[95] Smyth, P. (1994) “Hidden Markov Models for Fault Detection in Dynamic Systems”, *In Journal of Pattern Recognition*, Vol.7, No. 1, pp.149-164.

[96] Keogh, E. and Smyty. P. (1997) “A Probabilistic Approach to Fast Pattern Matching in Time Series Databases”, *In proceedings of 3rd International Conference of KDD*, pp. 24-40.

[97] Frost, A. J. and Prechter, R. R. (1998) “Elliott Wave Principle”, New Classics Library, first edition.

[98] Jain. A and Dubes, R. (1988) “Algorithms for Clustering Data.”, Prentice Hall.

[99] 張 へい, 松原 茂樹. (2008) “株価データに基づく新聞記事の評価.”, 第 22 回人工知能学会全国大会.

[100] Rosset, S., Murad, U., Neumann, E., Idan, Y. and Pinkas, G. (1999) “Discovery of fraud rules for telecommunications - challenges and solutions”. *In proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, pp. 409-413.

[101] Zhu, Y.Y and Shasha, D. (2003) “Efficient elastic burst detection in data streams”. *In proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, pp. 336-345.

[102] Aggarwal, C.C. and Yu, P.S. (2000) “Finding generalized projected clusters in high-dimensional spaces”. *In proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, pp. 70-81.

[103] Thomas, J. D. and Sycara, K. (2000) “Integrating genetic algorithms and text learning for financial prediction.”, *In Data Mining with Evolutionary Algorithms*, pp. 72-75. Technical Report WS-99-06.

[104] Elkan, C. (1999) “Notes on discovering trading strategies.”.

[105] Wu, H. M., Salzberg, B. and Zhang, D. H. (2004) “Online Event-driven Subsequence Matching over Financial Data Streams”, *In proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pp. 23-34.

[106] Scott, D., Susan T.D., George W.F., Thomas K.L. and Richard, H. (1990) “Indexing by Latent Semantic Analysis”, *In Journal of the American Society for Information Science*, Vol. 41, No. 6, pp. 391-407.

[107] Cao, L., Yu, P., Zhang, C. and Zhang, H. (2008) “Data Mining for Business Applications”, Springer.

[108] Cao, L., Zhao, Y., Zhang, H., Luo, D., Zhang, C., and Park, E.K. (2010) “Flexible Framework for Actionable Knowledge Discover”, *In IEEE transactions on Knowledge Discovery and Engineering*, Vol. 22, No. 9, pp. 1299-1312.

PUBLICATION

学術雑誌論文

1. Fan, W., Watanabe, T. and Asakura, K. (2008) “A framework for flexible clustering of multiple evolving data streams”, *In International Journal of Advanced Intelligence Paradigms*, Vol. 1, No. 2, pp. 178-195.

2. Fan, W., Watanabe, T. and Asakura, K. (2010) “Mining underlying correlated-clusters in high-dimensional data streams”, *In International Journal of Social and Humanistic Computing*, Vol. 1, No. 3, pp. 282-299.

査読付国際会議・ワークショップ

1. Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2008) “Clustering over Evolving Data Streams Based on Online Recent-biased Approximation”, *In proceedings of Pacific Rim Knowledge Acquisition Workshop*, LNAI 5351 (Springer), pp. 273-287.

2. Fan, W., Watanabe, T. and Asakura, K. (2009) “Ratio Rules Mining in Concept Drifting Data Streams”, *In proceedings of International Conference on Machine Learning and Data Analysis*, Volume. II, pp. 809-814.

3. Fan, W., Watanabe, T. and Asakura, K. (2009) “Mining Interesting Ratio Patterns over a Stream Sliding Window”, *In proceedings of Conference on Machine Learning and Applications*, pp. 731-734.

国内研究会・ワークショップ

1. Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2008) “An Incremental PCA for Stream Analysis Based on NLMS Adaptive Filter”, 電気関係学会東海支部連合大会論文集, O-511.

2. Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2008) “Clustering on Multi-granularity Temporal Trend of Multiple Evolving Data Streams”, 情報学ワークショップ論文集, pp. 215-220.

3. 小柳佑介，范薇，朝倉宏一，渡邉豊英 (2008) “類似度によるパターン分類を用いた時系列データ圧縮手法の提案”，情報学ワークショップ論文集, pp. 33-36.

4. Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2009) “Generalized Regression Measure for Local Correlation Tracking in Evolving Data Streams”, 第 1 回データ工学と情報マネジメントに関するフォーラム, E6-3.

5. 小柳佑介，范薇，朝倉宏一，渡邉豊英 (2009) “時系列データ圧縮のための類似部分区間探索手法”, 第 1 回データ工学と情報マネジメントに関するフォーラム，E6-4.

6. Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. (2009) “A System for Mining Correlations among Multiple Evolving Data Streams”, 人工知能学会全国大会, pp. 1-4.

7. Fan, W., Watanabe, T. and Asakura, K. (2010) “News-sensitive Multi-solution Prediction of Stock Price”, 情報処理学会全国大会, pp. 1-257-1-258.

8. 范 薇, 渡邉 豊英, 朝倉 宏一 (2010) “連続的時系列データストリームからの知識発見”, Technical Report of IEICE: DE2010, pp. 7-12.

9. 范 薇, 渡邉 豊英, 朝倉 宏一 (2010) “Multi-solution Trend Analysis of Stock Price Change by Textual Information”, FIT, 第一分冊, pp.163-166.

受賞

1. 范 薇. 平成 20 年度電気関係学会東海支部連合大会奨励賞, 2008.

2. Fan, W., Koyanagi, Y., Asakura, K. and Watanabe, T. 情報学ワークショップ 2008 優秀賞, 2008.

3. 范 薇. 平成 21 年度情報処理学会東海支部学生論文奨励賞, 2009.