Time-variant Graph Learning and Classification

Haishuai Wang

Faculty of Engineering and Information Technology

University of Technology, Sydney

A thesis submitted for the degree of

Doctor of Philosophy

30 October 2016

This thesis is dedicated to my loving parents

CERTIFICATE OF ORIGINAL AUTHORSHIP

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Student: Haishuai Wang
Date: 30.10.2016

Acknowledgements

I benefited and learned a lot from my supervisors, my colleagues, and my friends during the PhD study at University of Technology, Sydney, Australia. I wish to take this opportunity to thank all of them.

Firstly, I would like to express my sincere gratitude to my advisors, Dr. Ling Chen, Dr. Peng Zhang and Prof. Xingquan Zhu, for the continuous support of my Ph.D study and related research, for their patience, motivation, and immense knowledge. Dr. Ling Chen has been supportive and has given me the freedom to pursue various projects without objection. She has also provided insightful discussions about the research. I am deeply indebted to Dr. Peng Zhang for his fundamental role in my doctoral work. Dr. Zhang provided me with every form of guidance, assistance, and expertise that I needed during my Ph.D study. In addition to our academic collaboration, I greatly value the close personal rapport that Dr. Zhang and I have forged over the years. I am also very grateful to Professor Xingquan Zhu for his scientific advice and knowledge and many insightful discussions and suggestions. Their friendship has also been important to me as they have often given me invaluable advice in a personal sense.

I would like to thank Professor Yixin Chen who has provided me with a great opportunity to visit Washington University in St. Louis and has given me a postdoctoral position. I am extremely grateful for his guidance and all the excellent discussions that I have had with him. His deep insights have helped me at various stages of my research. I also give thanks to Professor Chengqi Zhang, Professor Ivor W Tsang, Professor Huan Liu, Professor Xindong Wu for their very helpful comments and suggestions which were aimed at improving my research skills.

I would also like to take this opportunity to thank all my friends in the Quantum Computation & Intelligent Systems Centre at UTS for all the great times that we have shared, in particular, Jia Wu, Shirui Pan, Shaoli Huang, Bo Han, Qin Zhang, Sujuan Hou, Lianhua Chi, Chunyang Liu, Guodong Long, Bozhong Liu, Anjin Liu, Yu Bai, and Tongliang Liu. They are the ones who have given me support during both joyful and stressful times, and to whom I will always be thankful. I am also grateful to Jemima Moore for proof reading my submission drafts.

Finally, I am deeply thankful to my parents and sisters for their endless love, encouragement, support, and various sacrifices. Without them, this thesis would never have been written. I dedicate this thesis to them.

Abstract

Graph classification is an important tool for analyzing data with structure dependency. In traditional graph classification, graphs are assumed to be independent where each graph represents an object. In a dynamic world, it is very often the case that the underlying object continuously evolves over time. The change of node content and/or network structure, with respect to the temporal order, presents a new time-variant graph representation, where an object corresponds to a set of time-variant graphs (TVG). A time-variant graph can be used to characterize the changing nature of the structured object, including the node attribute and graph topological changing over time. Therefore, the evolution of time-variant graphs could be either network structure or node content over time. In this dissertation, we formulate a new time-variant graph learning and classification (TVGLC) task.

To learn and classify time-variant graphs, the vital steps are feature extraction, modeling and algorithm design. However, for time-variant graph classification, frequent subgraph features are very difficult to obtain. Because one has to consider the graph structure space and the temporal correlations to find subgraph candidates for validation, the search space for finding frequent subgraph features is infinite and unlikely to obtain stable structures. Secondly, graph structures that imply subgraph features may irregularly change over time. Thus, to extract effective and efficient features is a great challenge for TVGLC. In addition, carrying out applicable models and algorithms to cater for the extracted features for TVGLC is also a challenge.

Considering the above challenges, this research aims to extract efficient features and design new algorithms to enable the learning of the

time-variant graph. Because time variant graphs may involve changes in the network structures and changes in the node content, which complicate the algorithm designs and solutions, our research employs a divide and conquer principle to first solve a simplified case where (1) network topology is fixed whereas the node content continuously evolves (i.e., networked time series classification). After that, we advance to the setting to (2) evolving network structure and propose solutions to TVGLC with incremental subgraph features. To enhance the subgraph feature exploration for time variant graph classification, we propose (3) graph-shapelet features for TVGLC. Last, but not the least, we study (4) an application of online diffusion provenance detection.

Temporal Feature Selection on Networked Time Series: As the time-variant graph can be graph node content and/or graph structure evolution, we first study a simple case where the structure is fixed but the node content continuously evolves. The problem forms time series data when the node content changes over time, and we combine time series data with a static graph to form a new problem called networked time series. We formulate the problem of learning discriminative features (*i.e.,* segments) from networked time series data considering the linked information among time series (*e.g.,* social users are taken as social sensors that continuously generate social signals (tweets) represented as time series). The discriminative segments are often referred to as *shapelets* of time series. Extracting shapelets for time series classification has been widely studied. However, existing works on shapelet selection assumes that time series are independent and identically distributed (i.i.d.). This assumption restricts their applications to social networked time series analysis. This thesis proposes a new Network Regularized Least Squares (NetRLS) feature selection model, which combines typical time series data and user network graph data for analysis.

Incremental Subgraph based TVGLC: To learn and classify the

time-variant graph with network structure evolve, the key challenges are to extract features and build models. To date, subgraphs are often used as features for graph learning. In reality, the dimension of the subgraphs has a crucial dependency on the threshold setting of the frequency support parameter, and the number may become extremely large. As a result, subgraphs may be incrementally discovered to form a feature stream and require the underlying graph classifier to effectively discover representative subgraph features from the subgraph feature stream. Moreover, we propose a *primal-dual incremental subgraph feature selection* algorithm (*ISF*) based on a max-margin graph classifier. The ISF algorithm constructs a sequence of solutions that are both primal and dual feasible. Each primal-dual pair shrinks the dual gap and renders a better solution for the optimal subgraph feature set. To avoid the bias of the ISF algorithm on short-pattern subgraph features, we present a new *incremental subgraph join feature selection* algorithm (*ISJF*) by forcing graph classifiers to join short-pattern subgraphs and generate long-pattern subgraph features.

Graph-shapelet based TVGLC: As graph structure continuously evolves over time, the search space for finding frequent subgraph features is infinite and unlikely to obtain stable structures. To tackle this challenge, we formulate a new time-variant graph classification task, and propose a new graph feature, *graph-shapelets*, for learning and classifying time-variant graphs. Graph-shapelet is compact and discriminative *graph transformation subsequences*. A graph-shapelet can be regarded as a graphical extension of *shapelets* – a class of discriminative features designed for vectorial temporal data classification. In order to discover graph-shapelets, we propose to convert a time-variant graph sequence as time-series data, and use shapelets discovered from the time-series data to find *graph transformation subsequences* as graph-shapelets. By converting each graph-shapelet as a unique tokenized graph transformation sequence, we can use the editing distance to calculate the distance between two graph-shapelets for time-variant graph classification.

Application of Online Diffusion Provenance Detection: In social network analysis, the information propagation graph (i.e., cascade) is a kind of time-variant graph because the information diffusion forms a graph at a certain time and the graph evolves over time. An important application of information diffusion networks (i.e., time-variant graph) is provenances detection. Existing work on network diffusion provenance identification focuses on offline learning where data collected from network detectors are static and a snapshot of the network is available before learning. However, an offline learning model does not meet the needs of early warning, real-time awareness and real-time response to malicious information spreading in networks. In this part, we study a new problem of online discovering diffusion provenances in large networks. To this end, we propose an online regression model for real-time diffusion provenance identification. Specifically, we first use offline collected network cascades to infer the edge transmission weights, and then use an online l_1 nonconvex regression model as the identification model. The proposed methods are empirically evaluated on both synthetic and real-world networks.

Experiments on synthetic and real-world data validate and demonstrate the effectiveness of the proposed methods for time-variant graph learning and classification.

Contents

List of Figures

LIST OF FIGURES

List of Tables

Chapter 1

Introduction

1.1 Background

Graph classification is an important branch of data mining research, given much structured and semi-structured data can be represented as graphs. Images [23], text [135, 136] and biological data [24] are just a few examples.

However, recently, time-variant graph data generated from dynamic graphs has entered the domain. For example, for the information propagation of graphs (*i.e.,* a series of graphs in time series), when analysing the information propagation in time windows, a sequence of graphs showing information propagation over time, where the target is to predict the class label that describes the outbreak or non-outbreak of an information propagation record, as shown in Figure 1.1. At a specific point in time, the status of the information diffusion is a graph, however the graph is evolutionary over time. Therefore, an information diffusion at different stages forms a time-variant graph.

A time-variant graph can be used to characterize the changing nature of a structured object. As both the node attributes and graph structure keep changing in each time-variant graph, in this thesis, we refer to a sequence of graphs whose node content and/or network structure continuously change as a time-variant graph. We first address a simple case where the graph structure is fixed but node content continuously evolves, which becomes the networked time series problem. In this case, only the attribute of nodes changes over time but the struc-

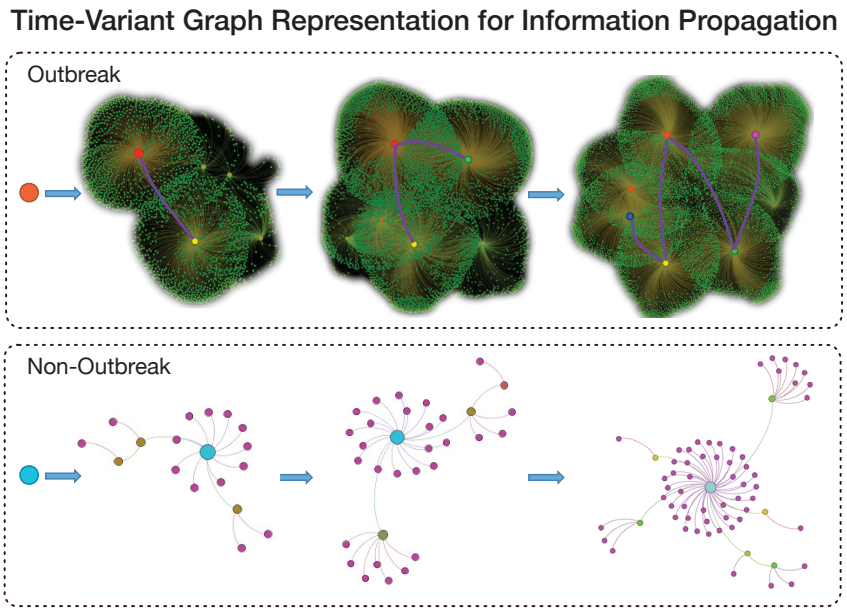


Figure 1.1: An example of a time-variant graph representation for large-scale information propagation. The information propagation can be regarded as a series of graphs. At different time periods, both the node volume and the graph structure are diverse, *i.e.,* the information propagation is a process that takes place on the graph. The information propagation outbreak prediction aims to build a time-variant graph classification solution to accurately identify an outbreak information diffusion graph (above) from a non-outbreak information diffusion graph (below).

ture of the graph is static. The variation of the nodes attribute over time forms the time series. By combining with the static graph structure, we call this kind of time-variant graph a networked time series. For example, users in social networks (i.e., Twitter) have follower/friend relationships, which constructs a graph. The number of tweets of each user in the graph changes over time (i.e., every day), which forms the time series. To predict the class label of users, both the user graph and tweets time series are used, which forms the networked time series. After this, we address another case where the graph structure changes. In this case, both the node volumes and network structure keep changing in each graph. For example, a chemical compound [24] can be represented as a graph. By varying the temperature or other environmental conditions, the structure of the chemical

compound with respect to each changed condition can be recorded, and a set of graphs (*i.e.* a time-variant graph) would capture its evolution. This representation is much more comprehensive and more accurate than simply representing a chemical compound as a single static graph.

1.2 Motivation

The main challenge in graph classification is that graphs do not have vectorial features directly available for classification, and therefore, traditional vector-based classifiers such as support vector machines (SVM) are not applicable. Research efforts in this area have focused on extracting discriminative substructures from graph data, so that graphs can be represented as vectorial features for learning and classification. One of the most widely used methods is to extract frequent subgraph patterns as discriminative features, and many studies have shown that this approach provides good performance [59, 43, 102].

Despite positive results for many applications [61, 158], using discriminative subgraph patterns as features, is based on a strict assumption that graph data is static, and frequent subgraph patterns are obtainable and finite. However, as time-variant graphs keep evolving over time, two essential challenges need to be addressed for feature extraction: the first is that the size of subgraph features may become extremely large, and the second is that the distance between subgraph features may irregularly change over time. Considering the dynamic nature of time-variant graphs, and the inherent complexity of learning tasks, this research has focused on exploring new features and designing algorithms for time-variant graph learning and classification (TVGLC) as well as its applications. The research first conducts a simple case where the network is fixed but the node content continuously evolves and formulates a new problem called networked time series classification. Then we advance to the setting of time-variant graphs of evolving network structure. From the perspectives of mining new features for the time-variant graph, this thesis explores three sub-tasks of features selection for handling the high dimension feature, infinite search space and networked time series: (1) Temporal Feature Selection on Networked Time Series, (2) Incremental Subgraph based TVGLC, and (3) Graph-shapelet Feature based TVGLC. Based

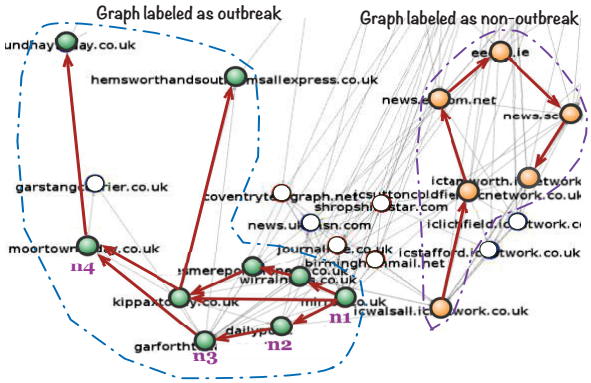


Figure 1.2: An example of an information diffusion network. The information propagation cascade can be regarded as a graph. The cascade on the left (with bold-faced edges and green nodes) quickly grows and propagates to an increasing number of nodes (*i.e.* outbreak), whereas the cascade on the right (with bold-faced edges and yellow nodes) remains steady and is therefore a non-outbreak cascade. Cascade outbreak prediction aims to build a graph classification model to accurately identify outbreak cascades from non-outbreak cascades.

on the explored new features, this research extends traditional graph mining to more complex time-variant graph data. All of these tasks are well motivated in a variety of applications.

- **Cascade Outbreak Prediction**. In cascade outbreak prediction in social networks [22], each cascade data record can be regarded as an acyclic graph that describes a path of information propagation in a social network. In Figure 1.2, two directed networks (with green and yellow nodes, respectively) show two cascades. Although both cascades start from a seed node, the cascade with green nodes propagates to a large number of nodes (*i.e.* graph labeled as outbreak), whereas the cascade with yellow nodes remains steady or may die off (*i.e.* graph labeled as non-outbreak). A cascade propagation with different stages forms a time-variant graph. Graph classification can then be used for cascade outbreak prediction to distinguish outbreak cascades from non-outbreak cascades. The general idea is to extract subcascades (subgraphs) as features by using frequent subgraph pattern mining.

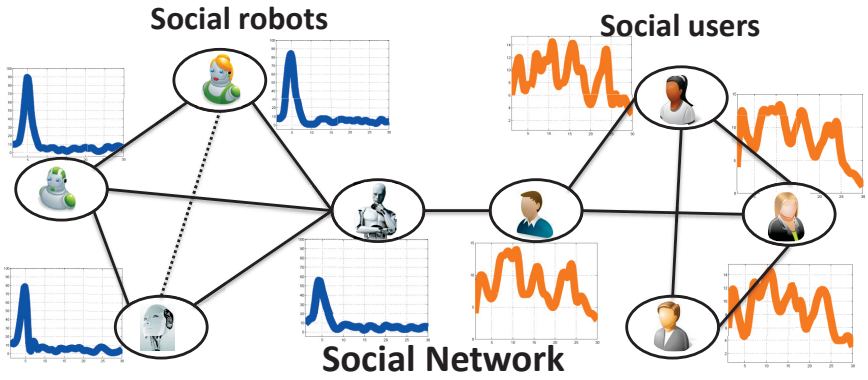


Figure 1.3: An illustration of social robots identification. We aim to identify social robots (Left) from real social users (Right). Each social node can be taken as a **social sensor** [104] that generates continuous social signals (tweets). Each social signal is independent and identically distributed because nodes are mutually connected in social networks.

- **Social Robots Identification**. In emerging applications such as Twitter, there are three types of nodes: 1) Social robots which are zombies controlled by a master node and occasionally distribute spams over the network. Because we have already located the master nodes, we can infer the network link among these zombies nodes. 2) Active social users who are famous/VIP users. These users are very active and update their pages frequently. The links among them are sparser than those of the social robots. 3) Ordinary social users who barely post messages, and the links are the sparsest.The task is to detect social robots that auto-distribute advertisements for viral marketing on social networks, as shown in Figure 1.3. As the number of tweets over time forms time series data, a naive approach would be to analyse the time series data to identify social robots. However, time series data generated by social users are not independent and identically distributed. Users construct a graph with follow/unfollow relationships and the attributes of each node in the graph change over time, which can be regarded as a kind of time-variant graph.

- **Online Diffusion Provenance Detection** In social networks, information propagation (cascade) can be regarded as time-variant graphs because the information diffusion forms a graph at a certain time and the graph

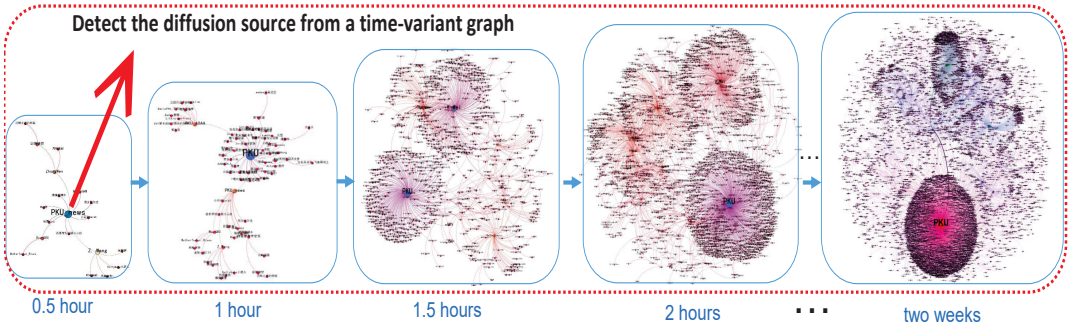


Figure 1.4: An example of time-variant graph provenance detection. In social networks, information propagation forms a cascade (time-variant graph). The cascade in the figure originates from a provenance (i.e., PKU_news) and propagates to a huge number of users over time (outbreak cascade). We aim to identify the provenance from the time-variant graph in a timely fashion to avoid malicious information break outs as early as possible.

evolves over time. An important application of information diffusion networks (i.e., time-variant graphs) is provenances detection, as shown in Figure 1.4. In recent years, information diffusion in large networks has attracted much attention. The spread of malicious information such as viruses, spams and rumors have made various networks vulnerable to privacy attacks and viral advertising etc. To stop the propagation of malicious information, researchers recently proposed several models to identify the diffusion provenances in large networks. However, to the best of our knowledge, most existing work on the diffusion provenance locating problem falls into the category of offline identification, where the data are assumed to be static and available all the time. In fact, for time-critical security monitoring applications, it is necessary to unveil the diffusion provenances as soon as an observation arrives. This way, it is important to detect diffusion provenances as early as possible to enable early warning, real-time awareness, and real-time response to the spread of malicious information.

The time-variant graph learning and classification (TVGLC) tasks are important but, as yet, have not been studied in the literature. The objectives of the dissertation are to model structured data, such as information propagation in social networks, as time-variant graph. Huge number of features can be generated

from a time variant graph. Hence, in order to learn and classify time-variant graphs, we need feature selection techniques to handle huge and variational feature space for different kinds of time-variant graph. In addition, an application of time-variant graph is proposed to real-time detect provenances from an information diffusion network.

1.3 Research Problems

As time-variant graph classification is essential in real-life applications while traditional features are not applicable, our research examines the unique challenge of time-variant graph classification tasks carefully and exploits several research problems accordingly.

1.3.1 Time-variant Graph Feature Extraction

As traditional subgraph features for graph classification are not applicable for the time-variant graph, in this thesis, we first generalize the aforementioned tasks as time-variant graph learning, then propose with three types of features to classify time-variant graph, whereby:

The time-variant can be regarded as the graph streaming form, and can be classified by traditional subgraph features. However, as the time-variant graphs change over time, the size of the graph in the last stage might be huge and the number of mined subgraphs may even become infinite. Therefore, how to deal with the infinite subgraph features for classification is one of the research problems in our research.

As the entire graph data changes over time and we are unlikely to obtain stable subgraph features in time-variant graphs, the conventional graph classification methods, i.e., the subgraph feature based method and distance based approaches (including graph kernel, graph, embedding, and transformation) are not applicable. Thus, new types of graph features are needed for time-variant graph classification.

As the attribute (node content) continuously evolves in the time-variant graph and this can be regarded as a networked time series, some existing time series

classification features can reoccur for attribute changed time-variant graph classification. A line of work has been proposed to extract discriminative features, which are often referred to as shapelets, from the time series. However, existing shapelets mining are only based on univariant/multivariant time series data. Extending the shapelets mining algorithms to the networked time series is another research problem in our research.

1.3.2 Time-variant Graph Classification Algorithms

The infinite number and variational features make existing classifiers unsuitable. Numerous feature selection models have been proposed to select sparse features by filter, wrapper, and embedding approaches. However, these feature selection methods are inefficient in infinite and variational feature space. For example, when using l_1 regularization, we need 1 TB memory to store the feature coefficients of 10^{12} features. Therefore, how to design models and build classifiers from graphs with infinite and variational features, and how to design an efficient algorithm to rapidly solve the graph classifier are also investigated in our thesis.

1.3.3 Evaluation of Proposed Features and Algorithms

Given the unique characteristics of the problem, various data is required to evaluate performance. Time-variant graphs are potential representations of large and dynamic graph data such as social network data. Both real-world and synthetic data sets are required to compare the proposed features and approaches with existing baselines.

In summary, how to evaluate the performance of the proposed features and algorithms is an important research problem in this thesis.

1.3.4 Applications of Time-Variant Graph Learning

As the time-variant graph is popularly employed in social networks, especially information propagation (cascade) in social networks, a case study needs to show the real-world application of the time-variant graph. Because the out-break time-variant network graph in information propagation probably causes malicious in-

formation (e.g., rumor) to be propagated to large populations, identifying a provenance in a timely way can avoid malicious information break outs as soon as possible. Thus, how to build a model to detect a provenance from the time-variant graph in real-time is an application case study of this thesis.

1.4 Thesis Contributions and Road Map

This thesis addresses a number of fundamental problems of time-variant graph learning and classification from four aspects; these are temporal feature selection on networked time series, incremental subgraph feature based TVGLC, graph-shapelets feature based TVGLC, and a case study of the time-variant graph. The main contributions of this study can be summarized in four parts, accordingly.

The thesis is structured into seven parts. We first provide a detailed literature review that surveys existing works on graph learning and classification, graph feature mining, graph stream mining, cascade outbreak prediction, big dimension data learning, feature based time series classification, feature selection in networked data and online diffusion provenances detection in Chapter 2, with major approaches in each learning task being summarized accordingly.

Chapter 3 proposes a Network Regularized Least Square feature selection method (NetRLS in short) to incorporate network information for shapelet selection. We test the model on a real-world Twitter data set to discover shapelets of *social robots* engaging in viral marking in Twitter, and we also test the model on another real-world DBLP data set to classify influential authors in the data mining area. Finally, we use a clinical heart rate data set as a case study to show the performance of the proposed model on medical data. The results demonstrate the advantages of the model.

In Chapter 4, we study the problem of time-graph classification with incremental subgraph features. We first propose a general max-margin graph classifier, based on which we propose a *primal-dual incremental subgraph feature selection* algorithm. The incremental algorithm constructs a sequence of solutions that are both primal and dual feasible. Each primal-dual pair shrinks the primal-dual gap and renders a better solution towards the optimal; We propose a new Incremental Subgraph Join Feature selection algorithm (ISJF for short). ISJF adds

a new constraint on the max-margin graph classifier and forces the classifier to select long-pattern subgraphs by joining short-pattern subgraph features; The performance of the algorithms is validated on four synthetic networks and two real-world networks (DBLP graph data and social network cascade data set with 2.76 million information cascades). The results show that the proposed incremental algorithm enjoys the merit of early prediction which is more than 400 seconds faster than existing models for cascading outbreak prediction.

Chapter 5 proposes a new class of graph patterns, *graph-shapelet patterns*, as features for time-variant graph classification. Graph-shapelet patterns are compact and discriminative graph transformation subsequences that can be used to classify graph sequences. Technically, in order to solve these challenges, we first convert a graph sequences into a time series by using a fast graph statistics. Then, we extract shapelets from the converted time series by using a traditional shapelets mining algorithms. Next, we locate the graph subsequences that match the shapelets from the original graph sequences. Each *graph subsequence* corresponds to a unique *graph transformation subsequence* based on our new graph transformation rules. At the last step, we extract the most discriminative *graph transformation subsequences* as graph-shapelets based on their graph edit similarity. Experiments on both synthetic and real-world data demonstrate the performance of the proposed algorithms.

Chapter 6 demonstrates a real-world application of the time-variant graph in social networks. We regard the information diffusion graph as the time-variant graph, and study a new problem of online diffusion provenance detection. We present a new online regression learning model to identify diffusion provenances in large networks. The proposed model can handle the issues of the unknown number of diffusion provenances k, the partially activated detectors, the unknown initial propagation time t^*, the uncertain propagation path, and the uncertain propagation time delay. In addition, we present an online stochastic algorithm to solve the proposed online regression learning model. The algorithm uses a stochastic sub-gradient decent algorithm to continuously detect the provenances.

We summarize the whole thesis and point out several future directions of this study in Chapter 7.

In summary, this thesis mainly forces on formulating the novel learning tasks

based on time-variant graph feature selection, model design, and its application. Some of the contents of this thesis come from our published papers [126, 123, 129, 125, 127, 124, 128, 9, 137, 138], with the details provided in the following section. Specifically, our proposed NetRLS algorithm includes network information, in addition to time series data, for temporal feature selection on networked time series data. This method was submitted to TII-2016 and a preprint version is available in [126]. Our proposed incremental subgraph join feature selection algorithm (ISJF) was first published in CIKM-2015 [123], and then the extended version with primal-dual incremental subgraph feature selection algorithm (ISF) was published in TKDE-2016 [129] and IJCNN-2016 [125]. Our proposed graph-shapelet feature for time-variant graph learning and classification is currently under second round review by TSMC and a preprint version is available in [127]. Our non-convex sparse regression model (NSR) for diffusion provenances detection in real time was published in IJCNN-2015 [124] and Journal of Computer Networks [128].

1.5 Publications

Below is a list of the papers associated with my PhD research that have been submitted, accepted, and published:

1. **Haishuai Wang**, Peng Zhang, Xingquan Zhu, Ivor Tsang, Ling Chen, Chengqi Zhang. Incremental Subgraph Feature Selection for Graph Classification. *IEEE Transactions on Knowledge and Data Engineering* (IEEE TKDE), vol 29, no 1, pp 128-142, 2017.

2. **Haishuai Wang**, Shirui Pan, Peng Zhang, Ling Chen. Towards Large-scale Social Networks with Online Diffusion Provenance Detection. *Journal of Computer Networks*, vol 114, pp 154-166, 2017.

3. **Haishuai Wang**, Jun Wu. Boosting for Real-Time Multivariate Time Series Classification. *Association for the Advancement of Artificial Intelligence* (AAAI), Accepted, In-press, 2017.

4. **Haishuai Wang**, Jia Wu, Xingquan Zhu, Chengqi Zhang. Time-Variant Graph Classification. *IEEE Transactions on Systems, Man, and Cybernetics* (TSMC), Major Revision, 2016.

5. **Haishuai Wang**, Peng Zhang, Ivor Tsang, Ling Chen, Chengqi Zhang. Defragging Subgraph Features for Graph Classification. *ACM International Conference on Information and Knowledge Management* (CIKM), 1687-1690, 2015.

6. **Haishuai Wang**, Peng Zhang, Ling Chen, Huan Liu, Chengqi Zhang. Online Diffusion Source Detection in Social Networks. *International Conference on Neural Network* (IJCNN), 1-8, 2015.

7. **Haishuai Wang**, Peng Zhang, Jia Wu, Shirui Pan. Top-k Minimal Redundancy Frequent Pattern Mining Over Uncertain Databases, *International Conference on Neural Information Processing* (ICONIP), 111-119, 2015.

8. **Haishuai Wang**, Peng Zhang, Ling Chen, Chengqi Zhang. SocialAnalysis: A Real-time Query and Mining System from Social Media Data Streams. *Australian Database Conference* (ADC): 318-322, 2015.

9. **Haishuai Wang**, Jia Wu, Chuan Zhou, Zhenyan Ji, Jun Wu. Mining Subcascade Features for Cascade Outbreak Prediction in Big Networks. *International Joint Conference on Neural Network* (IJCNN), 3942-3949, 2016.

10. Jia Wu, Shirui Pan, **Haishuai Wang**, Zhihua Cai. Locally Weighted Learning: How and When Does It Work in Bayesian Networks? *International Journal on Computational Intelligence Systems*, vol 8, no 1, 63-74, 2015.

11. Jun Wu, Zhibo Xiao, **Haishuai Wang**, Hong Shen. Learning with both unlabeled data and query logs for image search. *Computers & Electrical Engineering* vol 40, no 3, pp 964-973, 2014.

12. Yu Bai, **Haishuai Wang**, Jia Wu, Yun Zhang, Jing Jiang and Guodong Long. Evolutionary Lazy Learning for Naive Bayes Classification. *International Joint Conference on Neural Network* (IJCNN), 3124-3129, 2016.

13. Yu He, Jun Wu, **Haishuai Wang**. Refinement for Improved Graph Ranking.*International Joint Conference on Neural Network* (IJCNN), Accepted, In-press, 2017.

14. **Haishuai Wang**, Jia Wu, Peng Zhang, Yixin Chen, Chengqi Zhang. Temporal Feature Selection in Networked Time Series. *IEEE Transactions on Industrial Informatics* (TII), under review.

15. **Haishuai Wang**, Qin Zhang, Jia Wu, Yixin Chen, Ling Chen. Semi-supervised Shapelets Learning. *International Joint Conference on Artificial Intelligence* (IJCAI), under review.

16. **Haishuai Wang**, Zhicheng Cui, Yixin Chen, Michael Avidan, Arbi Ben Abdallahz. Cost-sensitive Deep Learning for Early Readmission Prediction at A Major Hospital. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, under review.

Chapter 2

Literature Review

This chapter presents a discussion of the research background and existing works in connection with this research. This work is closely related to a variety of works from different aspects: temporal features for networked time series classification, incremental subgraph features and graph-shapelet features for time-variant graph learning and classification, and online diffusion provenance detection. In this chapter, we review the related work accordingly.

2.1 Dynamic Graph

Graphs are commonly used to represent objects, such as images and text, for pattern classification. Graph classification is an important tool for social network and biological data analysis, where the objective is to learn and classify objective represented in graph structure. For example, chemical compounds can be represented in graph formats, predicting chemical compound activities in bioassay tests is a known graph classification problem. In social networks, a user is a node in the graph data, and the relationship (like following or fans) is the edge in the graph.

In a dynamic world, an object may continuously evolve over time, which forms dynamic changing graph records. Examples of the dynamic changing graphs are delay-tolerant networks [52], opportunistic-mobility networks [18], and complex networks [69]. As the notion of (static) graph is the natural means for representing

a static network, the notion of dynamic graph is the natural means to represents these highly dynamic networks. Some research efforts carried out a number of approaches for dynamic graph or graph streams learning and classification. The authors in [3] propose a random walk method to classify nodes in a dynamic content-based graph. [1, 19] proposes a hash-based probabilistic method to find discriminative subgraphs for massive graph streams classification. A Nested Subtree Hash Kernel algorithm [70] is proposed to project different subtree patterns to a set of common low-dimensional feature spaces, and construct an ensemble method for graph stream classification. [146] adopts an incremental learning strategy to incorporate the discriminative substructure patterns to improve the algorithm performance.

Even dynamic graph or graph streams are closely related to the problem considered in this paper, however, there are two main differences with our TVGLC problem: 1) dynamic graph and graph streams do not consider the relationship between two graphs at different time. In other words, they classify graphs at a certain time instead of the entire time-variant graph; 2) the graph size or subgraph feature dimension is small, while the subgraph features mined from the entire time-variant graph is usually high-dimensional. For example, it is not reasonable if we mine features at a certain graph in an information propagation network because the features from the entire propagation network are more valuable for classification. While outbreak information propagation usually has huge number of subgraph features because of the large size network. Therefore, it is significantly to propose new features and algorithms for time-variant graph learning and classification.

2.2 Temporal Features for Networked Data

In this section, we give a brief review of state-of-the-art feature learning based time series classification methods and feature selection approaches in networks respectively.

2.2.1 Discriminative Features for Temporal Data

Discriminative features for temporal data classification have been studied extensively [27, 55, 140]. Examples include bursts [58], periods [33], anomalies [134], motifs [72], shapelets [147] and discords [145]. Time series shapelets have recently attracted increasing interest in data mining [42, 80, 141], because shapelets are usually much shorter than the original time series which means we only need one shapelet for classification rather than the entire data set. Shapelets were first proposed by [147] as time-series segments that maximally predict the target variable. However, the runtime of brute-force shapelet discovery is not feasible due to the large number of candidates. Therefore, a series of speed-boosting techniques such as early abandonment of distance computations and entropy pruning of the information gain metric have been proposed [147]. The work in [81] relies on the reuse of computations and pruning of the search space to speed up the shapelets discovery. The work [37] proposes a novel method that learns near-to-optimal shapelets directly, without the need to search exhaustively among a pool of candidates extracted from time-series segments. However, all addressed time series issues fall into either the univariate or multivariate problem categories, and ignore the structural information behind a time series which can be represented with graphs.

2.2.2 Feature Selection in Networked Data

Many supervised feature selection algorithms have been proposed to select informative features from labeled data. A commonly used criterion in feature selection is to score the features. State-of-the-art methods to score the features are the filter-based, wrapper-based, and embedded approaches [41]. The embedded methods combine feature selection with the classifier, and are often considered as more effective than the first two methods [41]. However, traditional feature selection approaches assume that the data is independent and identically distributed, which is not suited for networked data. Based on the embedded method and graph regularization, Laplacian Regularized Least Squares (LapRLS) [13] is proposed for networked data, [40] which combines linear regression and graph regularization to select features in networked data and outperforms traditional feature selection

methods from the networked method.

Since the prior works are either for time series data or networked data, the research for networked time series data is an untouched area. Therefore, none of the aforementioned works can directly address the networked time series classification problem studied in this work.

2.3 Incremental Subgraph base TVGLC

The increasing availability of networked data is creating great potential for knowledge discovery from graph data. Generally, real-world network graphs tend to be big and complex. One difficulty of big graph data is the transfer of graph data into proper formats for learning methods to train classifiers for graph classification.

2.3.1 Graph Classification

Existing **graph classification** methods mainly fall into two categories: 1) Distance based methods, which design a pairwise similarity measure between two graphs, such as the graph kernel [54], graph embedding [99], and transformation [100]. The main shortcomings of this type of method are that it is computationally expensive to calculate graph distance and prediction rules are hard to interpret, because graph features are numerous and implicit. 2) Subgraph feature-based methods, which identify significant subgraphs as signatures for one particular class. [34] proposed to extract subgraph structural information for classification. [94] formulated subgraph selection as a combinatorial optimization problem, used heuristic rules, and combined a frequent subgraph mining algorithm, gSpan [45], to find subgraph features. [60] presented a semi-supervised subgraph feature selection method which uses unlabeled graphs to boost subgraph feature selection for classification. Several boosting methods [62, 46] use individual subgraph features as weak classifiers to build an ensemble for graph classification. None of the existing subgraph-based methods consider high dimension of subgraph features as incremental.

2.3.2 Incremental Feature

To handle high dimensional features, many research efforts have been made to address the **incremental feature** challenge. The work in [91] proposed a grafting algorithm based on a stage-wise gradient descent approach for incremental feature selection. However, grafting is ineffective in dealing with incremental features with unknown feature size because choosing a suitable regularization parameter inevitably requires information about the global feature set. The work in [155] studied stream-wise feature selection and proposed two algorithms based on steam-wise regression, information-investing and alpha-investing. However, this method only considers adding new features and never evaluates the redundancy of selected features after new features are added. The authors in [47] presented a framework based on feature relevance. The work [119] used a new, adaptive complexity penalty, the Information Investing Criterion (IIC), to dynamically adjust the threshold on the entropy reduction required for adding a new feature. The work in [139] involved the tolerance of information for selecting incremental features. These methods require prior knowledge about the structure of the feature space in order to heuristically control the choice of candidate feature selection. In real-world applications, obtaining such prior knowledge is difficult.

The **primal-dual** approach provides a powerful tool for designing approximate online and incremental algorithms. Typical primal-dual methods include primal-dual online linear programming [17], primal-dual online set cover [76], and primal-dual online SVMs [114]. By following the weak duality theorem [111], online learning algorithms quickly converge to approximate solutions by continuously updating both primal and dual feasible solutions which generate tight competitive bounds with respect to off-line algorithms.

2.3.3 Cascade Outbreak Prediction

Cascade outbreak prediction has been studied extensively. Most existing research efforts can be categorized into two classes: how to detect an outbreak cascade with minimum detection time or minimum affected population [68], and how to predict outbreaks at an early stage according to the cascading behaviors of a given network and dynamic cascades over the network [22]. The former

assumes that a portion of nodes in a given cascade can be accessed and used to select some of them as sensors for outbreak detection, whereas the latter aims to predict whether an arbitrary given cascade may outbreak or not. However, both of them use the nodes in a network as features for classification and prediction, ignoring the fact that cascades consist of sequential paths, while monitoring nodes in a network is costly because a user may post messages intensively.

2.3.4 High Dimensional Data Learning and Data Stream Mining

High dimensional data learning. High dimensionality is important and challenging because the immense growth of feature dimensionality in data analytics has exposed the inadequacies of many existing methodologies [149]. Directly learning a classifier from high dimensional subgraphs is infeasible. So far, many feature selection methods have been proposed to transform high dimensional data into a lower space representation while preserving the intrinsic data structure. The existing feature selection methods are often categorized as filter, wrapper, and embedding approaches [74]. Among the above sparsity induced methods, l_1 regularization has been popularly used in the literature [116]. However, l_1 regularization is inadequate in this work because it is inefficient when the data dimension is ultra-high. For example, 1TB memory is needed to store the weight vector w when the data dimension is 10^{12}. Moreover, feature selection bias inevitably exists in the l_1 norm, and different levels of sparsity can be achieved by changing the regularization parameter C [117].

Data stream mining is one of the important data mining tasks. Existing data stream mining algorithms [4, 79, 89, 161, 152] focus on the challenges of high dimensional stream records, concept-drift, concept-evolution, and feature-evolution. However, none of these stream algorithms touches the problem of incremental subgraph features.

Frequent subgraph mining approaches, such as AGM [48], Gaston [82] or gSpan [142], have been applied to enumerate frequently appearing subgraph patterns. Then, a graph is represented as a vector of binary indicators, and the traditional methods such as streaming feature selection are applicable to graph

data. In contrast to incremental feature selection and subgraph mining methods, we have combined streaming feature selection and subgraph extraction to tackle high dimensional subgraph features for graph classification.

2.4 Graph-shapelet based TVGLC

The increasing availability of structured data holds great potential for knowledge discovery from graph data. Generally, real graphs tend to be time-variant graphs with time changing. However, one of the difficulties when classifying time-variant graphs is converting the graph data into a format that allows extraction of effective features for graph classification.

2.4.1 Graph Features

Exploring new graph features is a persistent research focus in the graph data mining community, and many feature extraction methods have been proposed [61, 158]. Existing graph classification approaches can be roughly categorized into two groups: 1) distance based methods that include a pairwise similarity measure between two graphs, such as graph kernel [8], graph embedding [98, 78, 101, 26], graph matching [108, 105], and transformation [51, 71, 130]; and 2) subgraph feature based methods that identify significant subgraphs as signatures for one particular class. For example, the work in [35] proposed to extract subgraph structural information for classification. The work in [95] formulated the subgraph selection problem as a combinatorial optimization problem and they used heuristic rules and a combination of a frequent subgraph mining algorithm such as gSpan [142] to find subgraph features. Some boosting methods, such as [103, 87] use individual subgraph features as weak classifiers to build an ensemble for graph classification. Yet none of these methods consider the dynamic property of graphs. Since the structure of time-variant graph may change continuously over time, the subgraph features mined by these existing methods fail to consider the variation between graphs at different time. If subgraph mining algorithms are applied to the entire time-variant graph directly, huge number of features can be generated from a time variant graph. This will be time-consuming and

inefficiently because of the large size of time-variant graph (such as outbreak information propagation networks). Hence, we need feature selection techniques to handle huge and variational feature space.

2.4.2 Graph Stream Mining

Some graph stream mining methods have been recently proposed that address the streaming feature challenge. The work [92] proposed a grafting algorithm based on a stage-wise gradient descent approach for streaming feature selection. The work [156] studied streaming feature selection and proposed two novel algorithms based on steam-wise regression, information investing and alpha investing. The work in [120] developed a new, adaptive complexity penalty, the Information Investing Criterion (IIC), for the model to dynamically adjust the threshold of the entropy reduction required for adding a new feature. In [90], a graph ensemble boosting approach was proposed to handle the imbalance in noisy graph stream classification. Again, these graph streaming methods ignore the dynamic nature behind graph sequences.

2.5 Online Diffusion Provenances Detection

Malicious information such as rumors and viruses has recently been observed propagating in networks, incurring privacy and security concerns [86, 64] and motivating the research of diffusion provenance detection. To date, existing works on diffusion provenance detection have focused on offline detection, where a snapshot of a large network or data harvested from detectors are assumed to be available in advance. In order to design an online detection algorithm, three technical questions need to be answered: 1) how to design stochastic propagation models, 2) how to design an objective function for online detection, and 3) how to design an online algorithm as the solution. We survey related work regarding these three aspects.

In terms of stochastic propagation models in diffusion provenance locating, existing works simulate the spreading by using infection models such as the *Susceptible-Infected-Recovered* (SIR) [157] model, the *Susceptible-Infected* SI [109]

model, and others [57, 122]. On the other hand, recent works [36, 31, 154] have claimed that modeling propagation cascades and information diffusion using continuous-time diffusion networks can provide accurate models.

Based on the stochastic models, several learning models were proposed to infer the provenances. For example, a recent work [109] provided a systematic survey of locating rumor provenances in a network, and presented a *rumor centrality estimator* to estimate the rumor provenances by assigning a score to each infected node. The work [30] studied the problem of a single rumor provenance locating with *priori knowledge*. Most existing estimators are based on either topological centrality measures [160] or distance measures between observed data. Owing to this, maximum likelihood estimator can be used to infer the provenances.

A limitation of the above works is that they all assume that the infection status of the nodes (i.e., labels) is known a priori. For example, the work [77, 109] considered the multiple infection provenances estimation problem and assumed that the number of infection provenances is unknown in advance. Some work [53] assumed that not all nodes are infected and only a subset of detectors are used for the provenance estimation.

For online algorithms, online learning has been extensively studied in machine learning. Typical methods include the Passive-Aggressive (PA) [21] and *truncated gradient* algorithms [63]. However, these online learning algorithms are based on linear and convex optimization, which do not fit our non-convex online learning problem.

Despite the complexity of inferring the diffusion provenances in a network, a simple heuristic is to say that the provenance is the center of the network [110]. There are many notions of network centrality, but a very common one is known as distance centrality, e.g., betweenness centrality [12], closeness centrality [84] and Bonacich centrality [15]. Betweenness centrality measures a node's centrality in a network. The infection closeness centrality heuristic claims the node with the maximum infection closeness is the source. Bonacich Centrality is a measure of the influence of a node in a network. One may argue that the most influential nodes are more likely to be the provenances of the diffusion. However, in actuality, an almost isolated node that has few connections to the most influential nodes is most likely to be the source. Therefore, traditional central-based algorithms are

hardly applicable.

In summary, none of the aforementioned works can be directly used to address the online diffusion provenance detection problem studied in this work.

Chapter 3

Temporal Shapelet Feature for Networked Time Series

3.1 Introduction

A simple case of the time-variant graph is that the network is fixed but node content/attributes continuously evolve. When the node attributes of graphs change over time, the sequence can be regarded as time-series, and some time-series features and classification approaches can be recurred to time-variant graph learning and classification. To date, there exist two categorises of time series classification approaches: distance-based methods, and feature-based methods. The former measures the similarity between two time series (*e.g.,* dynamic time warping (DTW) [56]), while the later considers time series as a feature vector so that a traditional feature-based classifier (*e.g.,* SVM or logistic regression) can be applied. For the feature-based methods, the features can be simple statistics (*e.g.,* mean and variance) or a subsequence of the time series (*e.g.,* shapeltes). Shapelets are discriminative segments of time series that best predict class labels [148]. We have also observed a line of work extracting accurate and interpretable shapelets for time series classification, such as the decision tree based shapelets extraction [148][97], regression-based shapelets learning [38], and the time series transformation method [73].

On the other hand, a recent work [39] proposes a new time series shapelets

learning approach. Instead of searching shapelets from a candidate pool, they use regression learning and aim to learn shapelets from time series. This way, shapelets are detached from candidate segments and the learnt shapelets may differ from all the candidate segments. More importantly, shapelets learning is fast to compute, scalable to large datasets, and robust to noise.

In this chapter, we propose a Network Regularized Least Square feature selection method (NetRLS in short) to incorporate network information for shapelet selection. We test the model on a real-world Twitter data set to discover shapelets of *social robots* engaging in viral marking in Twitter, and we also test the model on another real-world DBLP data set to classify influential authors in data mining area. Finally, we use a clinical heart rate data set as case study to show the performance of the proposed model on medical data. The results demonstrate the advantages of the model.

The remainder of this chapter is organized as follows. Section 3.2 gives the preliminaries and problem definition. We introduce the proposed network regularized least squares shapelets learning model in Section 3.3. In Section 3.4, we conduct the experiments on real-world data sets and compare the proposed method with benchmark approaches. Finally, we draw a conclusion and point out the future work in Section 3.5.

3.2 Preliminaries and Problem Definition

In the problem setting, we have two types of data: a network $G = (V, E)$ where nodes $|V| = n$ and edges $|E| = l$, and a set of time series data denoted by a matrix $\boldsymbol{X} \in \mathbb{R}^{q \times n}$ where the j-th column vector $\boldsymbol{x_j} = [x_j^{(1)}, x_j^{(2)}, \cdots, x_j^{(q)}] \in \mathbb{R}^q$ represents time series generated by node $v_j \in V$. There are totally c class labels denoted by a label matrix $\boldsymbol{Y} \in \{0, 1\}^{c \times n}$ where each row $\boldsymbol{y_j} \in \mathbb{R}^c$ is a unit vector denoting the label of node v_j and $\boldsymbol{x}_j \in \boldsymbol{X}$.

3.2.1 Time Series Segments

Consider a sliding window of length t, when the window slides along a time series, a set of segments can be obtained. For time series $\boldsymbol{x}_j \in \boldsymbol{X}$, we can

generate totally $q-t+1$ segments by sliding the window from $\boldsymbol{x}_j^{(1)}$ to $\boldsymbol{x}_j^{(q-t+1)}$. Thus, for the entire time series $\boldsymbol{X}$, there are totally $(q-t+1)\times n$ segments, i.e., $\boldsymbol{\Omega} = [\varphi_1, \cdots, \varphi_{(q-t+1)\times n}]$ where each $\varphi_j \in \boldsymbol{\Omega}$ denotes a segment. Each element $\boldsymbol{s}_{(j,k)}$ is the distance between time series $\boldsymbol{x}_j$ and segment φ_k. It can be defined as the differential minimum distance that approximately denotes the minimum distance between the time series and the segment. Note that the segment length $t \ll q$, the number of segments $(q-t+1)\times n$ is very large.

3.2.2 Shapelets

Shapelets are defined as the most discriminative time series segments [148]. Therefore, time series segments are shapelet candidates, and we can use $\boldsymbol{\Omega}$ as the feature space for shapelets selection. To represent each time series $\boldsymbol{x}_j \in \boldsymbol{X}$ in the space $\boldsymbol{\Omega}$, we use a column vector $\boldsymbol{s}_i = [s_{i,1}, \cdots, s_{(i,(q-t+1)\times n)}]$ to record $\boldsymbol{x}_j's$ feature values, where each element $s_{i,j}$ depends on a distance function between $\boldsymbol{x}_j$ and segment $\varphi_j \in \boldsymbol{\Omega}$, i.e., $s_{i,j} = d_{(\boldsymbol{x}_i,\varphi_j)}$ (We will discuss this distance function later). This way, the time series data set $\boldsymbol{X}$ can be represented by a data matrix $\boldsymbol{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_n] \in \mathbb{R}^{(q-t+1)\times n, n}$, where each column vector $\boldsymbol{s}_j$ represents a time series $\boldsymbol{x}_j$ in space $\boldsymbol{\Omega}$. Note that each $\boldsymbol{s}_j$ is a ultra-high dimensional vector.

3.2.3 Goal

The purpose is to select the most discriminative segments as shapelets. Consider a multi-class problem with c class labels, denote a mapping matrix $\boldsymbol{W} \in \mathbb{R}^{q\times c}$ where the $j-$th column stores the classier $\boldsymbol{w}_j$ that identifies the $j-$th class from the remaining $c-1$ classes. We expect to obtain a sparse matrix $\boldsymbol{W}$ with only a few non-zero row vectors by minimizing the $L_{2,0}-$norm $\|\boldsymbol{W}\|_{2,0}$. The $L_{2,0}$-norm of $\mathbf{W}$ is defined as $\|\mathbf{W}_{2,0}\| = card(\|w_1\|_2, \cdots, \|w_c\|_2)$. w_j shrinks to zero if the j-th feature is not discriminative. Therefore, the features corresponding to zero column of $\mathbf{W}$ will be discarded when performing feature selection. This way, a few segments (row vectors in $\boldsymbol{W}$) are selected as the shapelets for building the classifiers.

3.3 Network Regularized Least Squares Shapelets Learning

Network regularization. Network information can help identify the classifiers $\boldsymbol{W}$. The idea behind this is to use network regularization based on the rule that, if two nodes are linked together, then they are likely to share the same class label. Technically, consider an undirect network with the adjacent matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ derived from the edge set E, the network regularization term $R_G(\boldsymbol{W})$ can be formulated as:

$$
\begin{aligned}
R_G(\boldsymbol{W}) &= \frac{1}{2} \sum_{k=1}^{c} \sum_{i,j} (\boldsymbol{w_k^T s_i} - \boldsymbol{w_k^T s_j})^2 \boldsymbol{A}_{ij} \\
&= \sum_{k=1}^{c} \sum_{i,j} \boldsymbol{w_k^T s_i A}_{ij} \boldsymbol{s_i^T w_k} - \sum_{k=1}^{c} \sum_{i,j} \boldsymbol{w_k^T s_i A}_{ij} \boldsymbol{s_j^T w_k} \\
&= \sum_{k=1}^{c} \sum_{i} \boldsymbol{w_k^T s_i D}_{ii} \boldsymbol{s_i^T w_k} - \sum_{k=1}^{c} \sum_{i,j} \boldsymbol{w_k^T s_i A}_{ij} \boldsymbol{s_j^T w_k} \\
&= \sum_{k=1}^{c} \boldsymbol{w_k^T S}(\boldsymbol{D} - \boldsymbol{W}) \boldsymbol{S^T w_k} \\
&= tr(\boldsymbol{W^T S L S^T W}),
\end{aligned}
\tag{3.1}
$$

where $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ is an undirected graph Laplacian [16] and $\boldsymbol{D}$ is a diagonal matrix called degree matrix with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$. Eq. (3.1) can be easily extended to a directed network by replacing the undirected graph laplacian with a directed graph laplacian $\boldsymbol{L} = \boldsymbol{\Pi} - \frac{1}{2}(\boldsymbol{\Pi P} + \boldsymbol{P}^T \boldsymbol{\Pi})$, where $\boldsymbol{\Pi}$ is a diagonal matrix and $\boldsymbol{P}$ is the transition matrix of random walk on the directed network [20].

3.3.1 Shapelets Selection

We use the embed-based feature selection [40]. Specifically, we propose to use a network regularized least squares learning (NetRLS for short) for shapelets selection. The NetRLS aims to learn c linear classifiers and select the top-k most

discriminative shapelets as shown in Eq. (3.2),

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \|\boldsymbol{Y} - \boldsymbol{W}^T\boldsymbol{S}\|_F^2 + \alpha\|\boldsymbol{W}\|_F^2 + \beta tr(\boldsymbol{W}^T\boldsymbol{S}\boldsymbol{L}\boldsymbol{S}^T\boldsymbol{W}) \\ s.t.: \quad & \|\boldsymbol{W}\|_{2,1} \leq \lambda, \quad \alpha, \beta > 0. \end{aligned} \tag{3.2}$$

The first two terms in the objective function are the regularized least squares and the third term is the network regularization. The constraint $\|\boldsymbol{W}\|_{2,1} \leq \lambda$ is a relaxation of the $L_{2,0}$ norm $\|\boldsymbol{W}\|_{2,0} \leq \lambda$ and $\|\boldsymbol{W}\|_{2,1}$ is defined as the sum of the l_2 norm of all the column vectors $\boldsymbol{w}_j \in \boldsymbol{W}$, i.e, $\|\boldsymbol{W}\|_{2,1} = \sum_j \|\boldsymbol{w}_j\|_2$. $\|\boldsymbol{W}\|_{2,1} \leq \lambda$ guarantees that at most λ rows in $\boldsymbol{W}$ are selected.

3.3.2 Challenges and Convexity

The matrix $\boldsymbol{S}$ is intimidatingly large because the segment space $\boldsymbol{\Omega}$ is ultra-high. Therefore, Eq. (3.2) cannot be solved directly on $\boldsymbol{S}$. In the sequel, we propose to trim matrix $\boldsymbol{S}$ by using the correlation of segments.

Keogh et al. [28] conducted an experimental comparison of time series representations and distance measures, where they compared eight representation methods and nine similarity measures and their variants and tested their performance on 38 time series data sets. They claimed that the Euclidean distance is surprising competitive with other more complex approaches, although it is very sensitive to misalignments. Because the focus of this chapter is not introducing new representation/distance methods, we simply use the Euclidean distance to measure the similarity between segments. Note our method can be extended to other representation/distance methods discussed in their work [28].

Define a diagonal matrix $\boldsymbol{M} = diag(0, 1, \cdots, 1, 0)$, where $rank(\boldsymbol{M}) = \rho \ll (q - t + 1) \times n$ which means only ρ element one in $\boldsymbol{M}$, and the problem turns to calculating M.

Based on the Euclidean distance, we define a distance matrix $\widetilde{\boldsymbol{U}}$ as in Eq.

(3.3). Note that the matrix is symmetric and non-negative.

$$\widetilde{\boldsymbol{U}} = \begin{pmatrix} d_{(\varphi_1,\varphi_1)} & \cdots & d_{(\varphi_1,\varphi_{(q-t+1)\times n})} \\ d_{(\varphi_2,\varphi_1)} & \cdots & d_{(\varphi_2,\varphi_{(q-t+1)\times n})} \\ \vdots & \ddots & \vdots \\ d_{(\varphi_{(q-t+1)\times n},\varphi_1)} & \cdots & d_{(\varphi_{(q-t+1)\times n},\varphi_{(q-t+1)\times n})} \end{pmatrix} \tag{3.3}$$

Based on the matrix, we define a diagonal matrix $\widetilde{\boldsymbol{P}} = \widetilde{\boldsymbol{D}} - \widetilde{\boldsymbol{U}}$, where $\widetilde{\boldsymbol{D}}_{ii} = \sum_j \widetilde{\boldsymbol{U}}_{ij}$. Then, selecting the optimal $\widetilde{S}$ is equivalent to selecting the maximum triangle elements, i.e.,

$$\begin{aligned} &\widetilde{\boldsymbol{M}} = \arg\max_{\boldsymbol{M}} tr(\widetilde{\boldsymbol{P}}\boldsymbol{M}) \\ &s.t. : rank(\boldsymbol{M}) = \rho. \end{aligned} \tag{3.4}$$

Then, we can obtain a lower space segment space $\widetilde{\boldsymbol{S}}$ based on $\boldsymbol{S}$, and $\widetilde{\boldsymbol{W}}$ based on $\boldsymbol{W}$, as shown in Eq. (3.5).

$$\widetilde{\boldsymbol{S}} = \boldsymbol{S}\widetilde{\boldsymbol{M}} \in \mathbb{R}^{\rho\times n}, \quad \widetilde{\boldsymbol{W}} = \boldsymbol{W}\widetilde{\boldsymbol{M}}. \tag{3.5}$$

Eq. (3.4) aims to find the top-ρ segments $\widetilde{\boldsymbol{S}}$ from all the segment candidates $\boldsymbol{S}$. The constraints denotes only ρ segments from $\widetilde{\boldsymbol{P}}$ are selected. The objective function denotes we want to obtain which segments have the maximal distances to all the other segments. Eq. (3.4) is easy to solve, because it is equivalent to selecting the maximum values on the diagonal matrix $\widetilde{\boldsymbol{P}}$.

Convexity. In Eq. (3.5), we have reduced the high dimension $\boldsymbol{W}$ and $\boldsymbol{S}$ into low dimension space $\widetilde{\boldsymbol{W}}$ and $\widetilde{\boldsymbol{S}}$. Once we replace $\boldsymbol{S}$ with $\widetilde{\boldsymbol{S}}$ in Eq. (3.2), we want to show if the problem is convex. If so, then we can use gradient-based algorithms as the solution.

Theorem 1. *The problem in Eq. (3.2) is convex* w.r.t. $\widetilde{\boldsymbol{W}}$ *and gradient-based algorithms can achieve a global optimum.*

Proof. Due to $\|\widetilde{\boldsymbol{W}}\|_F^2 = tr(\widetilde{\boldsymbol{W}}^T\widetilde{\boldsymbol{W}})$, Eq. (3.2) can be converted to the following optimization problem with parameters $\alpha, \beta > 0$,

$$\min_{\|\widetilde{\boldsymbol{W}}\|_{2,1}\leq\lambda} tr(\widetilde{\boldsymbol{W}}^T[\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^T + \alpha I + \beta\widetilde{\boldsymbol{S}}\widetilde{\boldsymbol{L}}\widetilde{\boldsymbol{S}}^T]\widetilde{\boldsymbol{W}} - 2tr(\widetilde{\boldsymbol{S}}^T\boldsymbol{Y}\widetilde{\boldsymbol{W}}^T) \tag{3.6}$$

Algorithm 1 NetRLS for Shapelets Selection on Networked Time Series.

Require:
Time series $\boldsymbol{X} \in \mathbb{R}^{q \times n}$, Network $G = (V, E)$, window length t, # of classes c, # of segments η, # of shapelets ρ;

Ensure:
Shapelets $\widetilde{\boldsymbol{S}}^*$;

Initialize α, β, θ_0, $\widetilde{\boldsymbol{W}}_1$, $\gamma_1 = 1$
Generate a segment space $\boldsymbol{\Omega} = [\varphi_1, \cdots, \varphi_{(q-t+1)\times n}]$
Generate an Euclidean distance matrix $\widetilde{\boldsymbol{U}}$ based on $\boldsymbol{\Omega}$
Generate a diagonal matrix $\widetilde{\boldsymbol{P}}$ based on $\widetilde{\boldsymbol{U}}$
Generate a selection matrix $\boldsymbol{M} = diag(0, .., 1), rank(\boldsymbol{M}) = \rho$
Solve $\widetilde{\boldsymbol{M}} = \arg\max_{\boldsymbol{M}} \ tr(\widetilde{\boldsymbol{P}}\boldsymbol{M}) \ \ s.t. : rank(\boldsymbol{M}) = \rho.$
Generate the candidate shapelet matrix $\widetilde{\boldsymbol{S}} = \boldsymbol{S}\widetilde{\boldsymbol{M}}$
Prune matrices in Eq. (3.2), $\widetilde{\boldsymbol{L}} = \boldsymbol{L}\widetilde{\boldsymbol{M}}$, $\widetilde{\boldsymbol{W}} = \boldsymbol{W}\widetilde{\boldsymbol{M}}$
repeat
  while $F(\widetilde{\boldsymbol{W}}_{\boldsymbol{k}}) > G_{\theta_{t-1}}(\widetilde{\boldsymbol{W}_{\boldsymbol{k+1}}}, \widetilde{\boldsymbol{W}}_k)$ **do**
    Set $\theta_{t-1} = \tau\theta_{t-1}$
  end while
  Set $\theta_k = \theta_{k-1}$
  $\widetilde{\boldsymbol{W}_{\boldsymbol{k+1}}} = \arg\min_{\widetilde{\boldsymbol{W}}} \ G_{\theta_k}(\widetilde{\boldsymbol{W}}, \widetilde{\boldsymbol{J}_{\boldsymbol{k}}})$
  $\gamma_{k+1} = \frac{1+\sqrt{1+4\gamma_k^2}}{2}$
  $\widetilde{\boldsymbol{J}_{\boldsymbol{k+1}}} = \widetilde{\boldsymbol{W}_{\boldsymbol{k}}} + \frac{\gamma_k - 1}{\gamma_{k+1}}(\widetilde{\boldsymbol{W}_{\boldsymbol{k+1}}} - \widetilde{\boldsymbol{W}_{\boldsymbol{k}}})$
until *Convergence*
$Score(i) = \sqrt{\sum_j \widetilde{\boldsymbol{W}}_{i,j}^2}$
Output: Segments $\widetilde{\boldsymbol{S}_{\boldsymbol{k}}}$ with the largest Scores

Let $\boldsymbol{\Lambda} = \widetilde{\boldsymbol{S}}\widetilde{\boldsymbol{S}}^T + \alpha I + \beta\widetilde{\boldsymbol{S}}\widetilde{\boldsymbol{L}}\widetilde{\boldsymbol{S}}^T$. Obviously, $\boldsymbol{\Lambda}$ is always non-negative and thus it is a positive semi-definite matrix. The constraint $\|\widetilde{\boldsymbol{W}}\|_{2,1} < \lambda$ is also convex. Therefore, the optimization problem is convex. □

3.3.3 Networked Time Series Classification Algorithm

We use a recently proposed gradient-based algorithm, the Accelerated Proximal Gradient Decent (APG) algorithm [40], as the solution. The computational complexity of APG algorithm can be found in [118]. The convergence rate of APG is very fast of $O(\frac{1}{\sqrt{(\epsilon)}})$. Because Eq. (3.2) is convex, APG can achieve a global

optimum.

Recall that the purpose of APG is to find a sequence of variables $\{\cdots, \widetilde{\boldsymbol{W}}_{\boldsymbol{k+1}}, \cdots\}$ such that the objective function converges to a global minimum. Eq. (3.2) can be relaxed to $F(\widetilde{\boldsymbol{W}}) = f(\widetilde{\boldsymbol{W}}) + \lambda\|\widetilde{\boldsymbol{W}}\|_{2,1}$, where $f(\widetilde{\boldsymbol{W}})$ is the objective function in Eq. (3.2) and $\lambda\|\widetilde{\boldsymbol{W}}\|_{2,1}$ is a relaxation of the constraint. According to the Taylor series expansion, $F(\widetilde{\boldsymbol{W}})$ approximately equals to $G_{\theta_k}(\widetilde{\boldsymbol{W}}, \widetilde{\boldsymbol{W}}_k)$ as follows,

$$\begin{aligned} G_{\theta_t}(\widetilde{\boldsymbol{W}}, \widetilde{\boldsymbol{W}}_k) = f(\widetilde{\boldsymbol{W}}_k) + &< \nabla f(\widetilde{\boldsymbol{W}}_k), \widetilde{\boldsymbol{W}} - \widetilde{\boldsymbol{W}}_k > \\ &+ \frac{\theta_k}{2}\|\widetilde{\boldsymbol{W}} - \widetilde{\boldsymbol{W}}_k\|^2 + \lambda\|\widetilde{\boldsymbol{W}}\|_{2,1} \end{aligned} \tag{3.7}$$

where $\nabla f(\widetilde{\boldsymbol{W}}_k)$ is the first order derivative of $f(\widetilde{\boldsymbol{W}})$ at $\widetilde{\boldsymbol{W}}_k$.

Now, the iterative step $\widetilde{\boldsymbol{W}}_{\boldsymbol{k+1}}$ can be obtained by minimizing $G_{\theta_k}(\widetilde{\boldsymbol{W}}, \widetilde{\boldsymbol{W}}_k)$, i.e.,

$$\begin{aligned} \widetilde{\boldsymbol{W}}_{\boldsymbol{k+1}} &= \arg\min_{\widetilde{\boldsymbol{W}}} \; G_{\theta_k}(\widetilde{\boldsymbol{W}}, \widetilde{\boldsymbol{W}_{\boldsymbol{k}}}) \\ &= \arg\min_{\widetilde{\boldsymbol{W}}} \frac{1}{2}\|\widetilde{\boldsymbol{W}} - \widetilde{\boldsymbol{V}}_{\boldsymbol{k}}\|_F^2 + \frac{\lambda}{\theta_k}\|\widetilde{\boldsymbol{W}}\|_{2,1} \end{aligned} \tag{3.8}$$

where $\widetilde{\boldsymbol{V}}_k = \widetilde{\boldsymbol{W}}_k - \frac{1}{\theta_k}\nabla f(\widetilde{\boldsymbol{W}}_k)$. It is not difficult to prove that the solution of Eq. (3.7) can reduce the objective function $F(\widetilde{\boldsymbol{W}})$ and the algorithm is convergent.

Eq. (3.7) can be further broken down into c separate subproblems, each of which has a closed form solution given in Eq. (3.9), where $\widetilde{\boldsymbol{w}}^i_{k+1}, \widetilde{\boldsymbol{w}}^i$ and $\boldsymbol{v}^i_k$ are the i-th rows of $\widetilde{\boldsymbol{W}}$, $\widetilde{\boldsymbol{W}}$ and $\boldsymbol{V}_k$ respectively.

$$\boldsymbol{w}^i_{k+1} = \begin{cases} (1 - \frac{\lambda}{\theta_k\|\boldsymbol{v}^i_k\|})\boldsymbol{v}^i_k, & if \ \|\boldsymbol{v}^i_k\| > \frac{\lambda}{\theta_k} \\ 0, & otherwise. \end{cases} \tag{3.9}$$

Moreover, we construct a linear combination of $\widetilde{\boldsymbol{W}}_{\boldsymbol{k}}$ and $\widetilde{\boldsymbol{W}}_{\boldsymbol{k+1}}$ to update $\widetilde{\boldsymbol{J}}_{\boldsymbol{k+1}}$ as follows,

$$\widetilde{\boldsymbol{J}}_{\boldsymbol{k+1}} = \widetilde{\boldsymbol{W}}_{\boldsymbol{k}} + (\gamma_k - 1)(\widetilde{\boldsymbol{W}}_{\boldsymbol{k+1}} - \widetilde{\boldsymbol{W}}_{\boldsymbol{k}})(\gamma_{k+1}), \tag{3.10}$$

where the sequence of γ_k is conventionally set to be $\gamma_{k+1} = \frac{1+\sqrt{1+4\gamma_k^2}}{2}$. The

Figure 3.1: The network (Left) has 200 nodes and 210 edges, social robots are densely connected while the remaining two groups of real users are sparsely connected. The generated time series (Mid.) of the three classes of nodes, social robots and active VIP users have discriminative patterns while ordinary users tend to have flat curves. The shaplets (Right) of a typical social robot is concave ([a, b, c]), while shapelets of an active VIP user is convex ([a,b,c]). Time series of ordinary users have heavy noise and it is hard to capture a shapelet.

Table 3.1: The data sets summarization.

Data set	Classes	Size	Length
Twitter	3	200	30
DBLP	2	700	20
Medical data	2	2565	50

algorithm is summarized in Algorithm 1.

3.4 Experiments

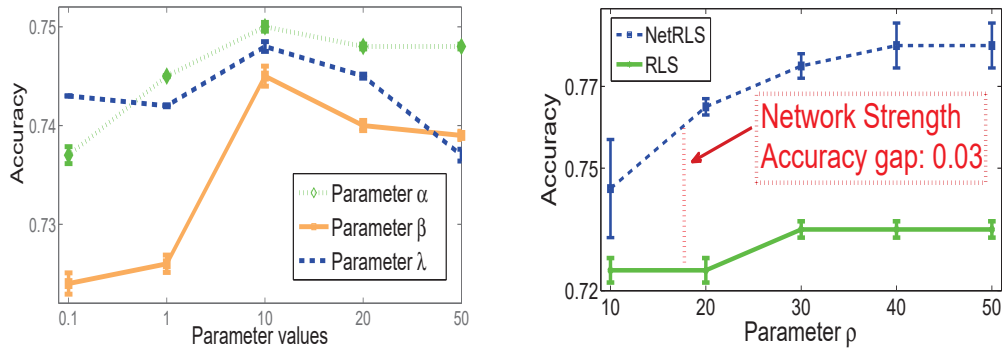
We design experiments to validate that the proposed NetRLS model, which combines both time series data and network data, can obtain better performance than using only time series data. All the experiments are conducted on a Linux Ubuntu server with 16*2.9GHZ CPU and 64G memory. The experiments are implemented in Matlab.

3.4.1 Data Sets

We collect a Twitter data set and a DBLP data set for testing, and we use a clinical heart rate data set for a case study to show the performance of the proposed model on medical data. The detailed information is summarized in Table 3.1.

3.4.1.1 Twitter

The task is to detect social robots that auto-distribute advertisements for viral marketing on social networks. We located and collected 200 social time series in the last 30 days from three types of nodes: 1) Social robots which are zombies controlled by a master node and occasionally distribute spams over the network. Because we have already located the master nodes, we can infer the network link among these zombies nodes. 2) Active social users who are famous/VIP users. These users are very active and update their pages frequently. The links among them are sparser than that of the social robots. 3) Ordinary social users who



(a) Parameter tests on α, β and λ.

(b) NetRLS VS. RLS.

Figure 3.2: Parameter test (a) and model comparison (b).

barely post messages, and the links are sparsest.

The network information is shown in Fig. 3.1. Fig. 3.1 gives a small portion of the time series data. We calculate the total number of tweets for each node in each day and obtain time series of length 30. There are 20 social robots, 30 active social users, and 150 ordinary social users. Intuitively, we can observe that the social robots have a very short yet sharp time period of distributing information. But in the rest time, these nodes are in sleep. The active social users have a more regular and frequent information distribution while the ordinary users are in low-frequency. These basic features can guarantee satisfactory shapelets for classification.

3.4.1.2 DBLP

The task is to classify if an author is an influential author in a given research field. We retrieve around 700 authors from DBLP [1] in data mining area, *i.e.*, authors in the conferences of ICDM, PKDD/ECML and KDD. There are 300 influential authors (*e.g.*, Jiawei Han, Philip S. Yu, *etc.*) and the remaining are normal authors in data mining area. Note that the normal authors do not represent small quantities papers in each year or small quantities papers in a certain data mining conference. We label the influential or normal users based on prior knowledge (*i.e.*, some already well-known researchers) and their frequent

[1] http://dblp.uni-trier.de/db/

co-authors appeared on top-5 data mining conferences. There are some authors who have duplication names (denoted as noise authors), we manually remove the noise authors and mislabeled authors. We crawl the number of papers in each year to form time series data of each author between 1996 to 2015. Then, we based on the coauthorship to construct the network information, *i.e.,* there exist an edge if two authors have coauthorship relation and set the weight is 1, otherwise 0.

Baseline Methods. To show the power of the network information in building a classifier, we compare the proposed NetRLS model with a regularized least squares (RLS) model which only uses time series data for classification. Also, we compare NetRLS with a state-of-the-art shapelet learning method (denoted as LTS) [38]. The LTS performance has been demonstrated better than other state-of-the-art methods [38], *e.g.*, the Fast Shapelets (FSH) which is a fast random projection technique on SAX representation [28], and the Dynamic Time Warping (DTW) [97].

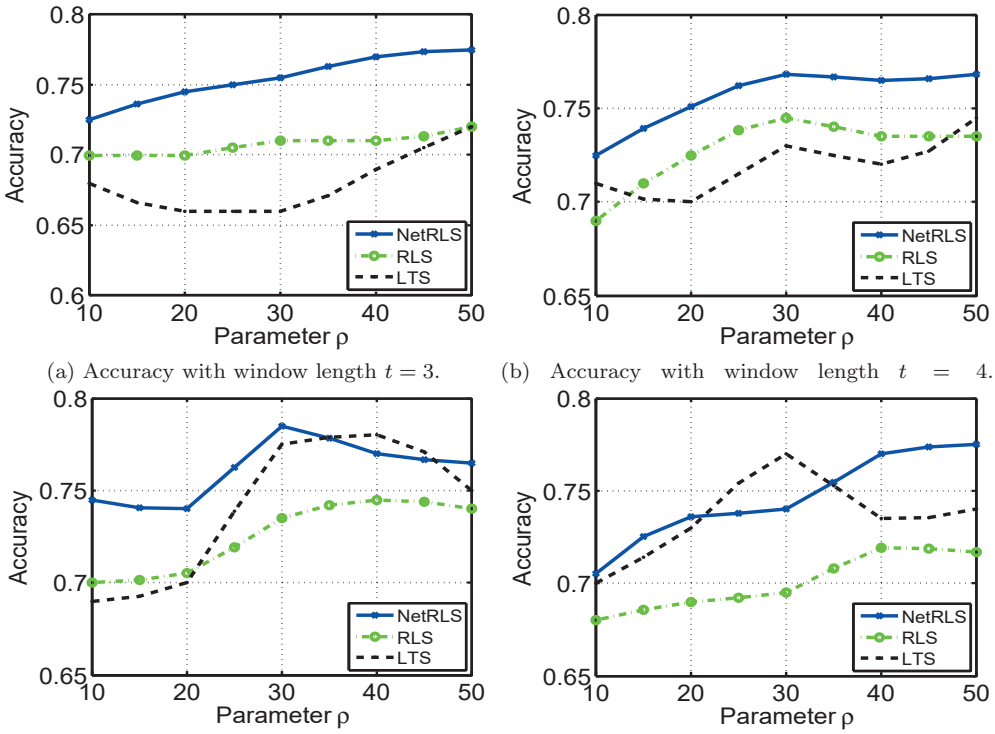
3.4.2 Experimental Measures

We measure the performance by classification accuracy: *tp / (tp+fp+fn+tn)*, where *tp* is true positive, *fp* is false positive, *fn* is false negative, and *tn* is true negative. We also use the AUC value to evaluate the performance. We use 70% for training and the remaining 30% for testing.

Parameter testing. Fig. 3.2(a) shows the parameter tests with respect to α, β, λ on the Twitter data set. The parameters are $\alpha = 1$, $\beta = 1$, $\lambda = 1$ by default. We can observe that when the parameter β is set to a small value of 0.1, the model obtains the worst result, this is because the network information is nearly dropped in the analysis. The sparse terms $\|W\|^2$ and $\|W\|_{2,1}$ obtain the best results when the parameters λ and β equal to 10. Thus, we will set the parameters $\alpha = 10$, $\beta = 10$, $\lambda = 10$.

3.4.3 Experimental Comparisons

Fig. 3.2(b) shows the improvement of NetRLS over RLS on Twitter data set. We can observe that NetRLS steadily lays above RLS. The accuracy gap is 3%



(a) Accuracy with window length $t = 3$.

(b) Accuracy with window length $t = 4$.

(c) Accuracy with window length $t = 5$.

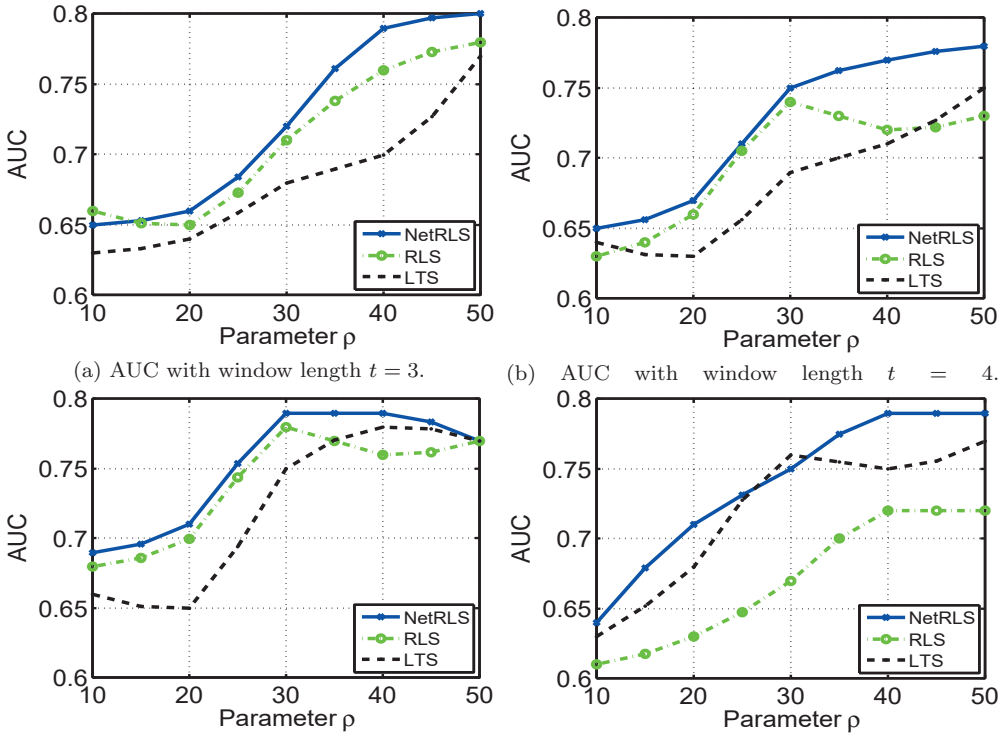
(d) Accuracy with window length $t = 6$.

Figure 3.3: Accuracy comparison on Twitter data set *w.r.t.* various window length and parameter ρ.

on average. The accuracy gap reflects the power of the network data. This is because social time series contain more noise than traditional time series data and network data is useful in improving performance.

Fig. 3.3 and Fig. 3.4 show the results of comparisons between *NetRLS*, *RLS* and *LTS* on the Twitter data set, and Fig. 3.5 and 3.6 show the results on the DBLP data set. We compare the two methods under different pairs of parameters: candidate segment (shapelet) size ρ and window length t.

From the results, we can conclude that: 1) given a shapelet space ρ, increasing the window length t will not guarantee better prediction results. When the length is 5, both NetRLS and RLS obtain relatively better results. 2) given a window length t, increasing ρ will generally improve the performance, but the improvement is insignificant when $\rho > 40$. For example, on the Twitter data set, NetRLS is 0.768 when $\rho = 30$ and $t = 4$, which is the same as $\rho = 50$. Considering that



(a) AUC with window length $t = 3$.

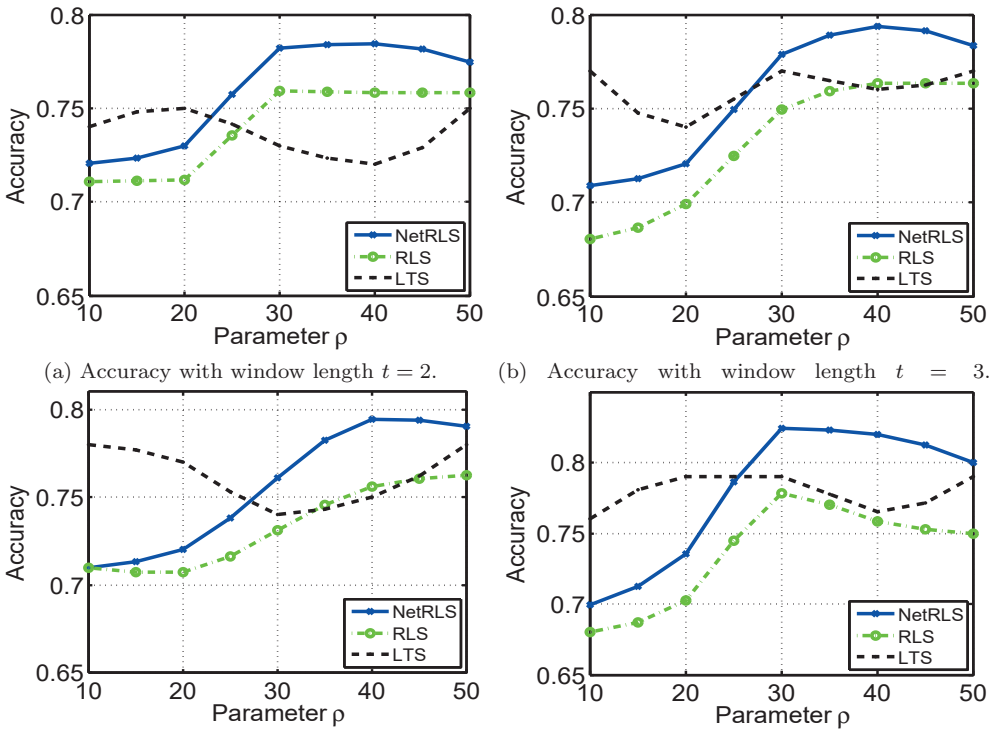
(b) AUC with window length $t = 4$.

(c) AUC with window length $t = 5$.

(d) AUC with window length $t = 6$.

Figure 3.4: AUC comparison on Twitter data set *w.r.t.* various window length and parameter ρ.

increasing ρ will generate more candidate segments and increase memory and computation costs, ρ can be set to between 30 and 50. 3) given the same ρ and t, NetRLS obtains higher accuracy and AUC value than RLS because NetRLS can model both time series and network data and obtain more accurate and robust results. 4) NetRLS performs better than LTS. LTS shows high accuracy on the UCR time series data sets but produces a different result in social network data sets (Twitter and DBLP). This is because there are undistinguishable users and noise when based solely on the time series data. For example, in the Twitter data set, a famous user may have only posted a few tweets during the date ranges fetched in our data set. This could easily be categorized to normal user if the number of tweets was the only consideration. In the DBLP data set, an author may only focus on the top conferences in data mining, and contribute a lot to them, but the total number of papers published per year may limited. Without



(a) Accuracy with window length $t = 2$.

(b) Accuracy with window length $t = 3$.

(c) Accuracy with window length $t = 4$.

(d) Accuracy with window length $t = 5$.

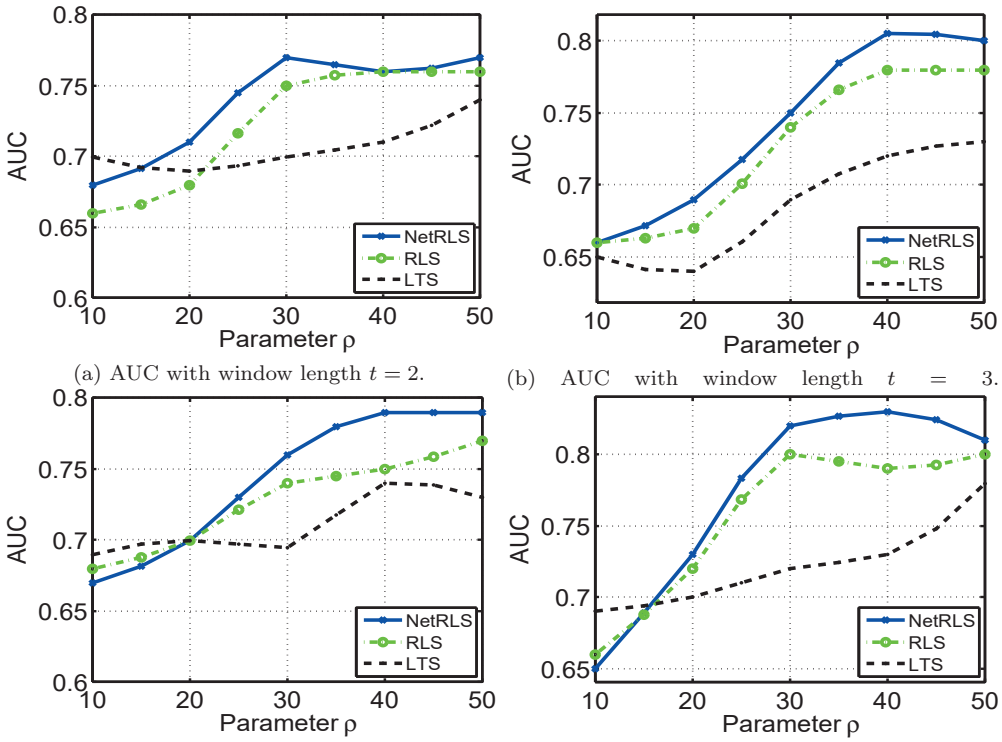
Figure 3.5: Accuracy comparison on DBLP data set *w.r.t.* various window length and parameter ρ.

additional data, this might indicate classification as a non-influential author.

Fig. 3.7 lists the shapelets learnt by NetRLS on the Twitter and DBLP data set. Fig. 3.7 (a) shows the social robots have sharp fluctuant shapelet since they usually post a mass of tweets within a short time, while normal users tend to post tweets with regular fluctuant shapelet. VIP users post tweets with gently fluctuant shapelet because they keep posting plenty of tweets. Fig. 3.7 (b) demonstrates normal authors have subtle changing shapelet because the published papers by normal authors is not significantly changed every year while influential authors tend towards great change due to coauthors or new topics.

Discussions. In this part, we discuss two problems regarding the selected shapelets from networked time series.

First, one may ask if the network information can alter the shapelets? In Eq. (3.2), the optimization problem has only one variable, the classifier weight



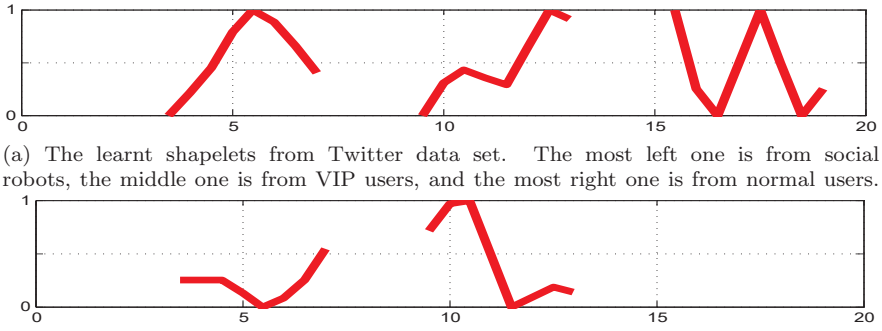
(a) AUC with window length $t = 2$.

(b) AUC with window length $t = 3$.

(c) AUC with window length $t = 4$.

(d) AUC with window length $t = 5$.

Figure 3.6: AUC comparison on DBLP data set *w.r.t.* various window length and parameter ρ.



(a) The learnt shapelets from Twitter data set. The most left one is from social robots, the middle one is from VIP users, and the most right one is from normal users.

(b) The learnt shapelets from DBLP data set. The most left one is from normal authors, while the most right one is from influential authors.

Figure 3.7: An illustration of the shapelets learned by NetRSL on the Twitter and DBLP datasets.

W, so the network information will change only the classifier boundaries but not shapelets themselves. In fact, our solution can be taken a two-step hierarchical

way that first solve Eq. (3.4) to get a trimmed concise segment space and then solve Eq. (3.2) to obtain shapelets. In our experiments, the network links are relatively sparse [1]. This is because social time series data often contain heavy noise and social nodes label are usually incorrect, network link information can somewhat alleviate the noise and mislabel problems, and thus improve the classification accuracy.

Second, in what case the link information can alter the shapelets themselves? We can relax Eq. (3.2) to be a more flexible problem that optimize the objective function between variables $\mathbf{W}$ and $\mathbf{S}$. This way, the third item of network regularization $\mathbf{L}$ will impact the optimal value of $\mathbf{S}$. However, similar to the work of learning shapelets [38], if we allow to optimize *w.r.t.* $\mathbf{S}$, we will arrive higher accuracy at cost of obtaining shapelets that slightly all the segments derived from time series data.

Note that we do not show run-time and memory consumption in the experiments because the data sets we used are not very large and the size of network is small, hence, the difference with the baselines is minor. In fact, the run-time and memory cost of the proposed algorithms mainly relies on the feature learning from time series, since time series is usually long and feature extraction from long time series is time and memory consuming. In contrast, the influence of run-time and memory cost for extracting network features is insignificant. Therefore, comparing with the accuracy improvement, the run-time and memory consumption of the proposed networked time series algorithm is insignificant.

3.4.4 Industry case study on clinical data

In this section, we test the proposed NetRSL method on a real-world clinical data from Saint Louis Children Hospital. This data set includes heart rate and patients age from 2565 patients. The heart rate data was recorded from admission to recovery (denoted as positive class) or transfer to ICU (denoted as negative class) for every minute. Based on the observation from the data, we find that the

[1] Fig. 3.1 shows the 200 nodes and 210 edges. Because we have located all the social robots, so all the links between social robots are captured. On the other hand, the edges between real users, including VIP users and ordinary users, are relatively sparse. Even though the network is sparse, we can still observe performance improvement as shown in Fig. 3.2(b).

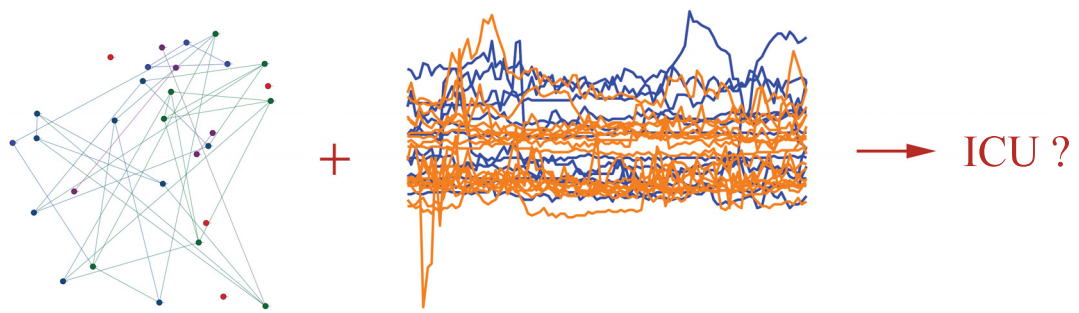
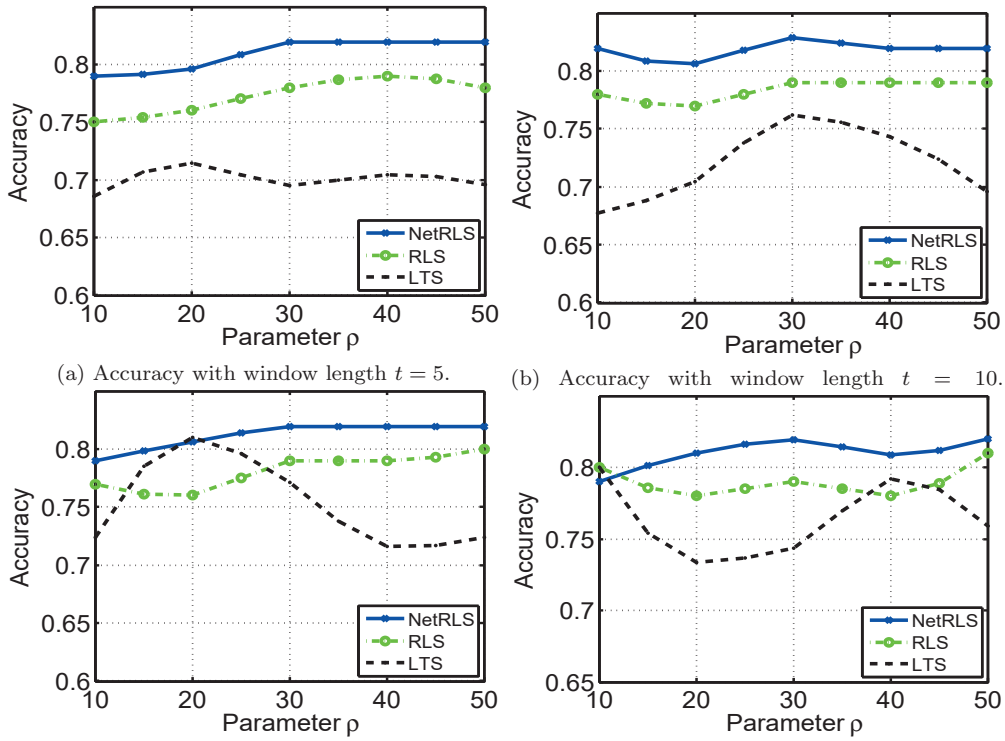


Figure 3.8: A part of the heart rate data. The network (Left) is based on age of patients, and we link two patients if their age difference is within 3 years. The middle time series represents heart rate for one minute interval. The blue time series represents these kinds of patient were transferred to ICU, and the orange one represents the patient was not transferred to ICU. We use both the patients network and heart rate time series to classify if the patient will be transferred to ICU or not. Obviously, only using the time series data with inseparability can make the learning task difficult. Thus, together with the network data we can improve the final classification performance.

age of patients can influence if the patient will be transferred to ICU. Thus, we compare the age of each two patients, there is a link between two patients if their ages differ within 3 years. In this way, we can form an age [1] information graph between each of the two patients, *i.e.*, a node represents a patient, and there exists an edge if the age differ of two patients is less than 3 years. The heart rate recorded in every minute constructs time series data. Fig. 3.8 shows the network structure and time series formed by a part of the clinical data, where the learning task only via time series data can be very difficult to solve. As there are many empty or very short records in this data set, we remove the patients whose data is empty or short. Usually, the last part of heart rate before transferring to ICU or recovery is more significant, so we cut out the length with 50 from the end of each time series. As a result, there are 147 negative class and 1432 positive class. To make the data set more balance, we randomly sample 353 instance from the

[1] In our clinical data set, besides the age features, we also have demographic features such as height and weight etc. However, those features are not very related to the heart rate data and ICU prediction. Thus, the reason why we only used age for constructing patients network is that age is the only one influential feature we can access from the data. The influence of the age network is illustrated in Fig. 3.9. We will include more factors when constructing the patient networks once we have more influential features to the outcome prediction in the future.



(a) Accuracy with window length $t = 5$.

(b) Accuracy with window length $t = 10$.

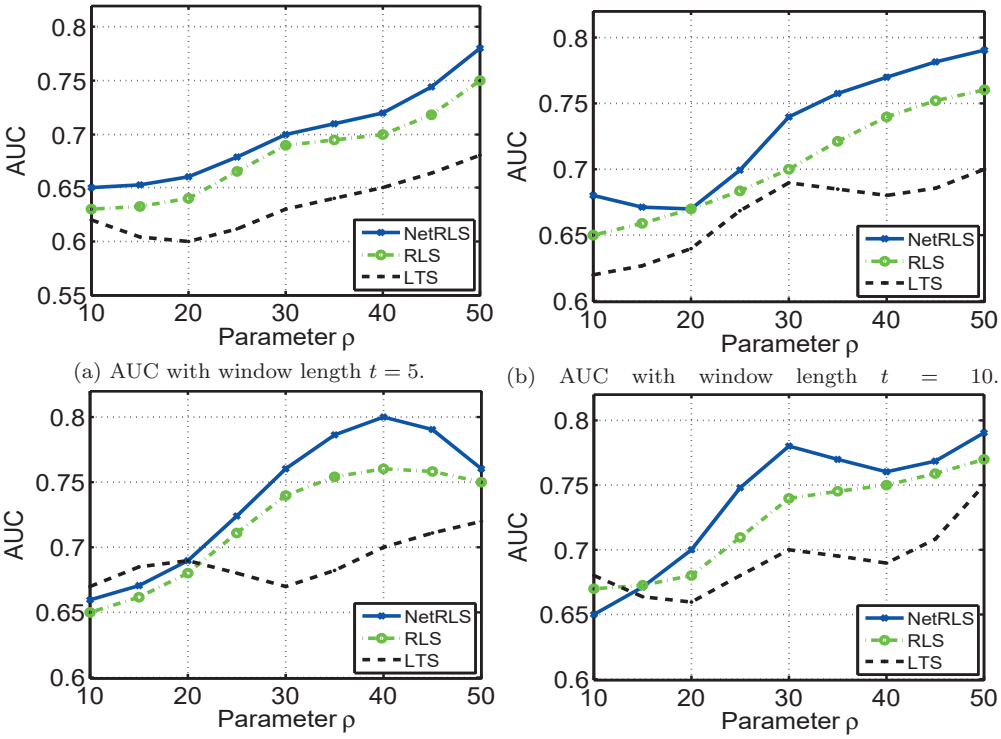
(c) Accuracy with window length $t = 15$.

(d) Accuracy with window length $t = 20$.

Figure 3.9: Performance on the clinical data set *w.r.t.* various window length and parameter ρ.

positive class. The task is to predict if the patient will be transferred to ICU or not.

Fig. 3.9 and Fig. 3.10 show the performance of NetRSL compared with RSL and LTS. We can see from the results that the accuracy and AUC of LTS is the lowest in the most of cases, which indicates that the traditional time series classification is not suitable in the heart rate data set. With the window length increasing, the proposed NetRLS can continuously outperform RLS and LTS. This is because that NetRLS utilizes the user network structure information to enhance the final classification performance.



(a) AUC with window length $t = 5$.

(b) AUC with window length $t = 10$.

(c) AUC with window length $t = 15$.

(d) AUC with window length $t = 20$.

Figure 3.10: AUC comparison on the clinical data set *w.r.t.* various window length and parameter ρ.

3.5 Conclusion

In this chapter, we have explored a new problem of networked time series classification, where the data contain both the typical time series data and network structure data. A network regularized least square feature selection method (NetRLS) is proposed to incorporate the network structure information for shapelet selection, accordingly. Our work drops the independent and identically distributed (i.i.d.). assumption and enables to use rich network structure information to improve the performance. The experiments and comparisons on real-world Twitter and DBLP, and validations on medical data analysis, demonstrate that NetRLS outperforms state-of-the-art time series shapelet learning algorithms and is suitable for a wide range of learning tasks.

Chapter 4

Incremental Subgraph based TVGLC

4.1 Introduction

When mining subgraph as features, the number of frequent subgraphs is sensitive to the setting of the support parameter. In reality, the number of subgraphs crucially depends on the setting of the frequent pattern mining threshold, and with a very small threshold value, the dimension of the subgraphs may be extremely large. As shown in Fig. 4.1, the dimension of the subgraph (subcascade) features increases exponentially as the frequent pattern threshold parameter *Supp* decreases. For example, when the parameter is 50, the number of discovered subgraph features is more than $8 * 10^5$!

The reality of high dimensional or potentially *infinite* subgraph features motivates the need to select a small number of representative subgraph features. In machine learning, high dimensional features are a common challenge and a large number of feature selection models [91, 139, 155] have been proposed to reduce data dimensionality. However, the information propagation network is dynamic with the time variation, and the number of subgraphs might tend to infinite. As the features are subgraphs and features are infinite, these models are incapable of handling infinite subgraph features. In this case, new feature selection techniques need to be proposed to handle huge and variational information subgraph feature

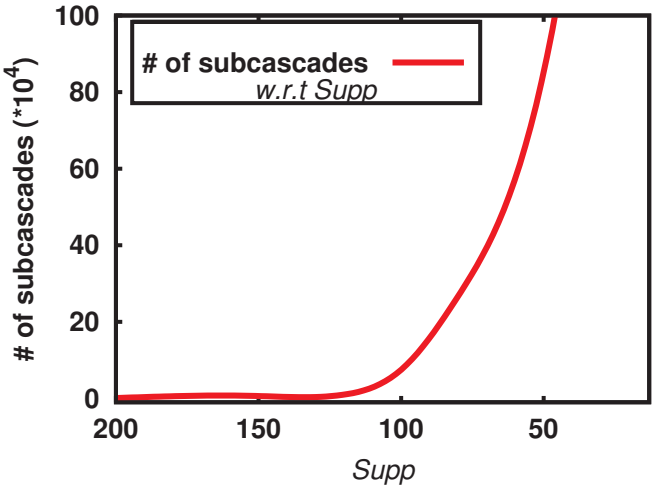


Figure 4.1: The number of frequent subgraphs *w.r.t.* the support threshold in frequent pattern mining. The cascade data, containing about 2.76 million cascades and 3.3 million nodes, are obtained from the SNAP data set (http://snap.stanford.edu/infopath/data.html/). When the parameter *Supp* is 50, the number of discovered subgraph features is more than $8 * 10^5$!

space effectively.

Subgraph feature selection [25, 144, 46] has been proposed in the literature to combine substructure mining and feature learning in graph data. For example, frequent substructure mining methods, such as AGM [49] and gSpan [45], are used to enumerate frequently appearing subgraph patterns, with each subgraph corresponding to a feature, so existing machine learning methods (*e.g.*, SVM [50]) can be applied to the converted feature space. These works, however, can only handle low dimension subgraph features extracted from small graphs. For instance, a recent work [46] used only 4,337 subgraph features in the experiment. Since the number of subgraph features in big graph (outbreak cascade) and under small threshold can be extremely large, traditional graph classifier cannot handle high-dimensional subgraph feature space. As a result, existing methods are inapplicable to big graph data such as social network cascade data where the number of connected subgraphs can be extremely large.

Online incremental feature selection [47] was recently proposed to select features from high dimensional feature streams. For example, a recent work [139] involved tolerance of information for selecting incremental features and presented

a method for selecting strongly relevant and non-redundant features on the fly. This is a two-phase algorithm, including online relevance analysis and online redundancy analysis. However, there is no graph mining involved in this work, which makes it is unable to compute the relevance and redundancy of graph features. Thus, this method is inapplicable to incremental subgraph features where the challenges of subgraph pattern mining and incremental feature selection are tangled. In addition, the method in [139] assumes that the prior knowledge on the structure of the feature space is known a priori, so heuristic rules can be applied. Since the support threshold for subgraph feature mining is a user-defined value, it is hard to know the structure of the feature space as a priori information. Therefore, this strong assumption does not hold in general graph and social network settings, neither.

Motivated by the above observations, our research mainly aim to solve the following challenges:

- How to build a classifier from graphs with an high dimensional number of subgraph features, and how to design an efficient algorithm to rapidly solve the graph classification. The high dimensional number of features makes existing classifiers either inapplicable or ineffective. Numerous feature selection models have been proposed to select sparse features by filter, wrapper, or embedding approaches. However, these feature selection methods are inefficient in high dimensional subgraph feature space. For example, when using l_1 regularization, we need 1 TB memory to store the feature coefficients of 10^{12} features.

- How to join short-pattern subgraphs and design a feature selection model that prefers to select long-pattern subgraph features. We observe that the feature space is dominated by short-pattern subgraphs due to the *downward closure property* of the frequent subgraph mining algorithms [143], *i.e.*, short-pattern subgraph features are always more frequent than long-pattern interesting subgraphs. To discover long-pattern subgraphs buried under a huge number of short-pattern subgraphs, we need to systematically design a new feature selection method. Generally, a small value of the frequency support threshold is employed such that all candidate subgraphs can

be preserved in the feature space. However, this method will unavoidably generate an exponential number of short-pattern subgraphs which flood interesting long-pattern subgraph features. While there are many possible ways to join short-pattern subgraphs, the join rules between subgraph fragments need to be established for efficient computation. New constraints are required to force traditional max-margin based graph classifiers to select long-pattern subgraph features.

- How to evaluate the performance of the proposed method. Both real-world and synthetic data sets are required to compare the proposed method with existing methods.

In this chapter, we aim to address discriminative subgraph features selection from high dimensional subgraph feature space for max-margin graph classification (Section 4.3.1). Our research extends the max-margin graph classifier to handle high dimensional incremental subgraph features using a primal-dual subgraph feature selection which continuously selects subgraph features that are both primal and dual feasible (Section 4.3.2). Because the primal-dual subgraph feature selection algorithm converges quickly and tends to select short-pattern subgraphs, we further propose a long-pattern driven subgraph feature selection model for selecting interesting long-pattern subgraphs (Section 4.3.3). In experiments, we test the methods on both synthetic and real-world data sets. The results show that the proposed algorithms can both solve the incremental subgraph feature problem and select discriminative long-pattern subgraph features.

The major contribution of the chapter is threefold:

- We study the problem of graph classification with incremental subgraph features. We first propose a general max-margin graph classifier, based on which we propose a *primal-dual incremental subgraph feature selection* algorithm. The incremental algorithm constructs a sequence of solutions that are both primal and dual feasible. Each primal-dual pair shrinks the primal-dual gap and renders a better solution towards the optimal.

- We propose a new Incremental Subgraph Join Feature selection algorithm (ISJF for short) [123]. ISJF adds a new constraint on the max-margin graph

classifier and forces the classifier to select long-pattern subgraphs by joining short-pattern subgraph features.

- The performance of the algorithms is validated on four synthetic networks and two real-world networks (DBLP graph data and social network cascade data set with 2.76 million information cascades). The results show that the proposed incremental algorithm enjoys the merit of early prediction which is more than 400 seconds faster than existing models for cascading outbreak prediction.

The rest of the chapter is organized as follows. Section 4.2 introduces the preliminaries including important definitions. Section 4.3 discusses the new classification model and extends the classification model into long-pattern feature mining. We present theoretic analysis in Section 4.4. Section 4.5 conducts experiments and Section 4.6 concludes the chapter.

4.2 Preliminaries

A graph $G = (V, E)$ consists of a set of nodes $V = \{1, \ldots, p\}$ and a set of edges $E \subseteq V \times V$. A directed acyclic graph (DAG) is a graph G whose edges are directed and do not contain directed cycles. We use lower-case bold-faced letters to represent vectors and upper-case bold-faced letters to represent matrices. For example, symbol $\mathbf{e}$ represents a unit vector with all entries equal to 1.

Definition 1. *(**Subgraph**) Consider a directed acyclic graph $G = (V, E)$, $g_i = (V', E')$ is a subgraph of G, i.e., $g_i \subseteq G$, iff (1) $V' \subseteq V$, (2) $E' \subseteq E$. If g_i is a subgraph of G, then G is a supergraph of g_i.*

Definition 2. *(**Subgraphs Join (SgJ)**) Consider two directed acyclic subgraphs g_i and g_j. The vertex sets are $V(g_i) = \{V_\alpha, \cdots, V_\beta\}$ and $V(g_j) = \{V_\alpha{}', \cdots, V_\beta{}'\}$. If $V_\beta = V_\alpha{}'$ or $V_\beta' = V_\alpha$, then $g_i \bigotimes g_j = \{V_\alpha, \cdots, V_\beta, V_\gamma{}', \cdots, V_\beta{}'\}$ is defined as SgJ, where $\bigotimes$ is a concatenation operation.*

Because subgraph mining often outputs many short-pattern subgraphs, Definition 2 joins these subgraphs to generate long-pattern subgraphs. The difficulty

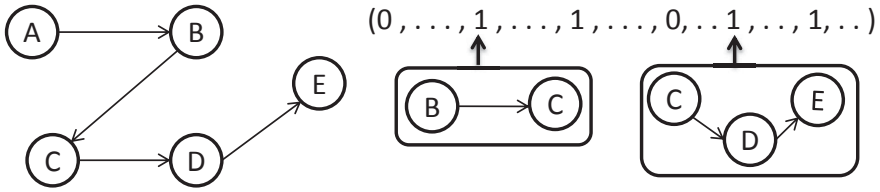


Figure 4.2: Subgraph features. The graph (left) is converted into a binary feature vector (right) by examining the existence of subgraph features. The feature vector can be processed by traditional classifiers such as SVMs.

is that the result may be uncertain due to graph isomorphism [159]. In this chapter, we only consider joining correlated subgraph features, as shown in Fig. 4.3. This is because correlated subgraph features have a high probability of generating interesting long-pattern subgraphs. As shown in Fig. 4.3, the join result is determined based on Definition 2.

In binary classification, the task is to learn a classification boundary from a training graph set $\{(G_k, y_k)\}$, $1 \leq k \leq n$, where each G_k is a training graph with class label $y_k \in \{-1, +1\}$.

Definition 3. *(**Subgraph Features**) Let $S = \{g_1, \cdots, g_m\}$ be a set of subgraphs in a training graph and $|S| = m$. Each graph G_k is encoded as an m-dimensional vector $\boldsymbol{x}_k$ with $\boldsymbol{x}_k^u (1 \leq u \leq m)$ denoted by*

$$\boldsymbol{x}_k^u = I(g_u \subseteq G_k), \ \forall g_u \in S,$$

where $I(\cdot)$ equals 1, if the condition is satisfied; otherwise, it is 0.

We use a simple example in Fig. 4.2 to explain the generation of the subgraph feature space. Consider a graph $A \rightarrow B \cdots \rightarrow E$ which contains subgraphs $B \rightarrow C$ and $C \rightarrow D \rightarrow E$, the corresponding elements in the subgraph feature space are set to 1.

Given a set of training graphs, the subgraph feature space increases exponentially *w.r.t.* graph size. Therefore, it is impractical to use all subgraphs as features. We use frequent subgraphs to prune trivial subgraph patterns.

A handful of algorithms have been proposed to mine frequent subgraphs $\mathcal{F}_s$ from a set of graphs, such as the Depth-First-Search (DFS) algorithm gSpan [45].

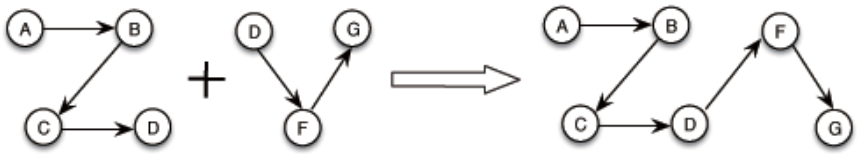


Figure 4.3: Joining correlated subgraph fragments.

The key idea of gSpan is that each subgraph has a unique DFS Code, which is defined by a lexicographic order during the search process. Two subgraphs are isomorphic iff they have the same minimum DFS Code. By employing a depth first search strategy on the DFS Code tree (where each node is a subgraph), gSpan can enumerate all frequent subgraphs efficiently.

4.3 Graph Classification

We present a *primal-dual incremental subgraph feature selection* algorithm based on a max-margin graph classifier. First, we assume that the subgraph feature space is finite, based on which we present a *max-margin graph classifier*. Then, we extend the classifier to handle high dimensional incremental features using the primal-dual subgraph feature selection.

4.3.1 Max-margin Graph Classifier

We introduce a feature scaling vector $\mathbf{d} \in [0,1]^m$ with $||\mathbf{d}||_1 = \sum_{i=1}^m d_j \leq B$ to encourage sparsity. This way, at most B subgraphs are selected. Given a graph G_i, we impose $\sqrt{\mathbf{d}} = [\sqrt{d_1}, \cdots, \sqrt{d_m}]^T$ on its features [121, 115] to a re-scaled example $\hat{\mathbf{x}}_i = \mathbf{x}_i \bigodot \sqrt{\mathbf{d}}$, where $\mathbf{x}_i \odot \sqrt{\mathbf{d}}$ represents the element-wise product between vectors $\mathbf{x}_i$ and $\sqrt{\mathbf{d}}$. Let $\mathcal{D} = \{d \in \mathbb{R}^m \mid ||\mathbf{d}||_1 \leq B, d_j \in [0,1], j = 1, \cdots, m\}$ be the domain of $\mathbf{d}$, the max-margin graph classifier can be formulated as follows,

$$\begin{aligned} \min_{\mathbf{d}\in\mathcal{D}} \min_{\mathbf{w},\xi,b} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^n \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1, \ldots, m. \end{aligned} \tag{4.1}$$

where $\mathbf{w} \in \mathbb{R}^m$ and b determine the classification boundary, ξ_i is the empirical error of $\mathbf{x}_i$, and C is a trade-off parameter. The problem is non-convex *w.r.t.* w and d.

When $\mathbf{d}$ is fixed, Eq. (4.1) degenerates to a standard SVM model. By introducing the Lagrangian multiplier $\alpha_i \geq 0$ to each constraint $y_i(w^T(x_i \odot \sqrt{d}) + b) \geq 1 - \xi_i$, and setting the derivatives of the Lagrange function to be 0 with respect to parameters $\mathbf{w}, \xi$ and b, we obtain

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}), \ \sum_{i=1}^{n} \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq C.$$

Plugging the above results back into Eq.(4.1), we obtain the dual form of the original problem as follows,

$$\min_{\mathbf{d} \in \mathbf{D}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} \quad -\frac{1}{2} \| \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \|^2 + \mathbf{e}^T \boldsymbol{\alpha} \tag{4.2}$$

where $\mathcal{A} = \{\boldsymbol{\alpha} | \sum_{i=1}^{n} \alpha_i y_i = 0, \boldsymbol{\alpha} \geq 0\}$. Based on the minimax saddle-point theorem [112], we can interchange the order of $min_{\mathbf{d} \in \mathbf{D}}$ and $\max_{\boldsymbol{\alpha} \in \mathcal{A}}$, and solve the following minimax problem instead,

$$\min_{\boldsymbol{\alpha} \in \mathcal{A}} \max_{\mathbf{d} \in \mathbf{D}} \quad \frac{1}{2} \| \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \|^2 - \mathbf{e}^T \boldsymbol{\alpha} \tag{4.3}$$

Apparently, the primal problem in Eq.(4.1) can be equivalently formulated as its dual in Eq. (4.3). Eq. (4.3) has only two variables $\mathbf{d}$ and $\boldsymbol{\alpha}$, and it is linear with respect to $\mathbf{d}$ and convex with respect to $\boldsymbol{\alpha}$, which can be solved by the *block coordinate descent algorithm* that alternates between the optimization of $\mathbf{d}$ and $\boldsymbol{\alpha}$.

Given a fixed $\mathbf{d}$, the optimization problem in Eq. (4.3) is reduced as follows,

(**Optimization 1**: fix $\mathbf{d}$ and solve $\boldsymbol{\alpha}$)

Algorithm 2 The max-margin graph classifier.

Require:
Graph $\mathbf{G}$, parameters C, B
Ensure:
Graph classifier f
$\boldsymbol{\alpha}^0 \leftarrow 1/C \cdot \mathbf{1}$, $U \leftarrow \emptyset$, $t \leftarrow 0$
repeat
  Calculate $\mathbf{d}_t$ based on $\boldsymbol{\alpha}_t$ using Eq.(4.5)
  // Optimization 2
  $U \leftarrow U \cup \mathbf{d}_t$
  Calculate $\boldsymbol{\alpha}_{t+1}$ based on U using Eq.(4.4)
  // Optimization 1
  $t \leftarrow t + 1$
until *Convergence*
Output $f \leftarrow SVM(U, C)$

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\|\sum_{i=1}^{n} \alpha_i y_i \hat{\mathbf{x}}_i\|^2 - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0,\ \alpha_i \geq 0,\ i = 1, \ldots, n. \end{aligned} \tag{4.4}$$

where $\hat{\mathbf{x}}_i$ is re-scaled by the given $\mathbf{d}$, *i.e.*, $\hat{\mathbf{x}}_i = \mathbf{x}_i \bigodot \sqrt{\mathbf{d}}$. Due to the sparsity of the scaler $\mathbf{d}$, the above problem can be solved by using standard quadratic programming with a small set of features $\hat{\mathbf{x}}_i$ (the formal statement is given in Appendix *A*).

When the variable $\boldsymbol{\alpha}$ is determined, we select the number of features B as in Eq. (4.5).

(**Optimization 2**: fix $\boldsymbol{\alpha}$ and solve $\mathbf{d}$)

$$\begin{aligned} \max_{\mathbf{d}} \quad & \|\sum_{j=1}^{n} \alpha_j y_j (\mathbf{x}_j \odot \sqrt{\mathbf{d}})\|^2 \\ \text{s.t.} \quad & \sum_{j=1}^{m} d_j \leq B,\ 0 \leq d_j \leq 1,\ j = 1, \cdots, m. \end{aligned} \tag{4.5}$$

To solve Eq. (4.5), we define a score function to denote the weight of each

feature, *i.e.*,

$$\mathbf{c}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \in \mathbb{R}^m. \tag{4.6}$$

Based on the above definition, we have

$$\| \sum_{j=1}^{n} \alpha_j y_j (\mathbf{x}_j \odot \sqrt{\mathbf{d}}) \|^2 = \sum_{j=1}^{m} [c_j(\boldsymbol{\alpha})]^2 d_j.$$

The optimization problem in Eq. (4.5) can then be converted to a linear programming problem with respect to $\mathbf{d}$ as follows:

$$\begin{aligned} \max_{\mathbf{d}} \quad & \sum_{j=1}^{m} [c_j(\alpha)]^2 d_j \\ \text{s.t.} \quad & \sum_{j=1}^{m} d_j \leq B,\ 0 \leq d_j \leq 1,\ j = 1, \cdots, m. \end{aligned} \tag{4.7}$$

Eq. (4.7) can be solved analytically. First, we construct a feasible solution by finding the top B largest scores $[c_j(\alpha)]^2$. Then, we set the number of B scaler d_j to 1 and the remaining $(m - B)$ d_j to 0. Clearly, such a feasible solution is also the optimal solution. The algorithm is given in Algorithm 2.

4.3.2 Incremental Subgraph Features

Eq. (1) provides a basic classifier for solving high-dimensional but finite subgraph feature space. To process high dimensional subgraph feature streams, we introduce a *primal-dual incremental subgraph feature selection* algorithm.

Our method is based on the online linear programming of the feature selection function given in Eq. (4.7). We assume that the constraint matrix is revealed column by column along with the objective function, *i.e.*, the features are processed column by column in a one-scan manner.

In incremental feature selection scenarios, a feature set is split and loaded into memory in a mini-batch manner. At each time window $t = \lceil m\epsilon \rceil$, the incremental algorithm learns both primal and dual feasible solutions. The primal problem is formulated as in Eq. (4.8) which allows $(1 - \epsilon)$ approximation to the

offline solution given by Eq. (4.9) [5]. Eq. (4.10) is clearly a small linear program problem defined on the $\lceil m\epsilon \rceil$ features.

(Primal)

$$\begin{aligned} \max_{\mathbf{d}} \quad & \sum_{j=1}^{t} [c_j(\alpha)]^2 d_j \\ \text{s.t.} \quad & \sum_{j=1}^{t} d_j \leq (1-\epsilon)\frac{t}{m} B \\ & 0 \leq d_j \leq 1, \ j = 1 \cdots, t \end{aligned} \tag{4.8}$$

For Eq. (4.8), we use $p \in \mathbb{R}^m$ to denote the dual variable. The dual problem of Eq.(4.8) is as follows:

(Dual)

$$\begin{aligned} \min_{p} \quad & \sum_{i=1}^{m} b_i p_i (1-\epsilon)\frac{t}{m} + \sum_{i=m+1}^{m+t} p_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} p_i + p_{i+j} \geq [c_j(\alpha)]^2, \ j = 1, ..., t \\ & p_i \geq 0, 1 \leq i \leq m. \end{aligned} \tag{4.9}$$

The dual problem converts a high dimensional problem in Eq. (4.8) into a big constraint problem with respect to dual vector p. For any given p, we define the function $x_t(p)$ denoting dual feasibility as follows,

$$x_t(p) = \begin{cases} 0 & \text{if } b_i \leq p^T \\ 1 & \text{if } b_i > p^T \end{cases} \tag{4.10}$$

The incremental algorithm constructs a sequence of solutions that are both primal and dual feasible. Each primal-dual pair shrinks the dual gap and renders a better solution towards the optimal.

In the online problem, at time t, the coefficient $c_j(\alpha)$ is updated, and the algorithm makes a decision x_t. Given the previous decisions $x_1, \cdots, x_{t-1}$, and $c_j(\alpha)^2$ till time t, the t^{th} decision is to select an x_t such that $\sum_{j=1}^{t} x_j \leq B$, $0 \leq x_j \leq 1$. The goal of the online algorithm is to choose x_t such that the objective function $\sum_{t=1}^{m} c_j(\alpha)^2 x_j$ is maximized.

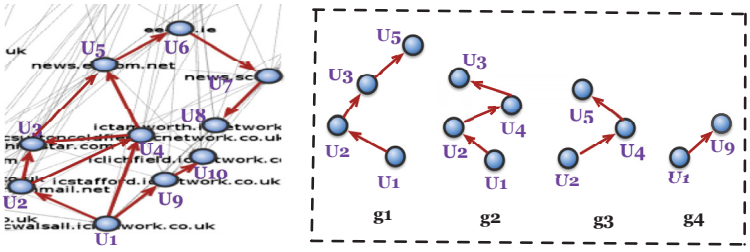
To evaluate the performance of an online algorithm, one approach is based on

Algorithm 3 Primal-dual incremental subgraph feature selection for graph classification

Require:
Graph G, parameters C, B, mini-batch size K
Ensure:
classifier c
$\alpha^0 \leftarrow 1/C \cdot \mathbf{1}$, $S \leftarrow \emptyset$, $t \leftarrow 0$
repeat
  $S \leftarrow$ a mini-batch of K features
  Calculate top-k candidate features based on α_t using Eq. (4.9) // *mini − batch scores*
  for each feature x_t in top-k **do**
    if $x_t(p) \leq b_i - \sum_{j=1}^{t-1} d_j x_j$ **then**
      $x_t = x_t(p)$
    else
      $x_t = 0$
    end if
  end for
  Calculate α_{t+1} based on d_t using Eq. (4.4)
  // *Optimization* 1
  $t \leftarrow t + 1$
until *no feature left*
Output $c \leftarrow SVM(S, C)$

its performance on the worst-case input, *e.g.*, completely robust to make input uncertainty [17]. This approach leads to gloomy bounds for the online problem: no online algorithm can achieve better than $O(1/m)$ approximation of the optimal offline solution [7]. In our problem settings, subgraph features arrive in a random order, and the total number of features m is known a priori. We consider the average behavior of the online algorithm over random permutations, and can use m to decide the length of history used to learn the dual bounds in the algorithm. In this case, the total number of m is a known priori [5], we can relate the approximate knowledge of m within at most $1 \pm \theta$ multiplicative error without affecting the final results.

The primal-dual incremental subgraph feature selection algorithm is given in Algorithm 2. The primal solution $x_t(p)$ constructed using sample dual solutions in Eq. (4.9) is a feasible solution to the linear program Eq. (4.8) with



(a) A social network cascade data set from http://snap.stanford.edu/infopath

(b) Four graphs of cascade data (g1, g2, g3, g4)

	f1: U1→U2	f2: U2→U3	Label
g1	1	1	+1
g2	1	1	+1
g3	0	0	-1
g4	0	0	-1

(c) Subgraph features f1 and f2

	f3: U1→U2→U3	Label
g1	1	+1
g2	1	+1
g3	0	-1
g4	0	-1

(d) Subgraph feature f3

Figure 4.4: An illustration of a long-pattern subgraph feature buried under two short-pattern subgraph features in the information cascade data. Consider four graphs $g_1, \ldots, g_4$. g_1 and g_2 from class "+1" while g_3 and g_4 from "-1". Assume we have two short-pattern subgraphs $f_1 : U_1 \rightarrow U_2$ and $f_2 : U_2 \rightarrow U_3$, and a long-pattern subgraph $f_3 : U_1 \rightarrow U_2 \rightarrow U_3$ by joining f_1 and f_2. If one feature is allowed to select for classification, then f_1 or f_2 is likely to be selected, instead of the more interesting f_3.

high probability. In fact, the iterative primal solutions constructed using sample dual solutions converge and approach to optimal, *e.g.*, with probability $1 - \xi$, $\sum_{t \in N} x_t(p) \geq (1 - 3\xi)OPT$ given $B \geq \frac{6mlog(n/\xi)}{\xi^3}$, where OPT denotes the optimal objective value for the offline problem [5].

4.3.3 Long-pattern Subgraph Features

In fact, the number and size of subgraph features crucially depend on the threshold parameters of frequent subgraph mining algorithms. Any improper setting of the parameters will generate many trivial short-pattern subgraph fragments which dominate the feature space, distort graph classifiers, and flood interesting long-pattern subgraphs.

The primal-dual incremental feature selection converges rapidly; however, the algorithm often traps in generating a large number of short-pattern subgraph

features which may flood interesting long-pattern subgraph features. Fig. 4.4 shows an example in which a long-pattern subgraph feature f_3 is buried under two subgraph features f_1 and f_2. A classifier is likely to choose the two subgraph features instead of the more informative long-pattern subgraph feature which reflects a latent social group. In this section, we design a long-pattern driven algorithm that prefers long-pattern subgraphs as features, with only one scanning of the feature set.

Intuitively, the size of a subgraph compromises the classification accuracy. In graph classification, based on the minimum description length theory, the relationship between the size of a subgraph and the size of the original graph is compromised when subgraphs are used as features.

Formally, we add extra constraints to allow the algorithm to choose long patterns. The new constraint can be stated as: *if two short patterns p_a and p_b are in the active feature set, i.e., $d_{p_a} = 1$ and $d_{p_b} = 1$, then their derived pattern $p_a \otimes p_b$ will be also selected, i.e., $d_{\ p_a \otimes p_b} = 1$.*

Based on the new constraints, we obtain a new classification model as follows,

$$\begin{aligned} \min_{d \in \mathbf{D}} \min_{w,\epsilon,b} \quad & \frac{1}{2}||w||^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(w^T(x_i \odot \sqrt{d} + b) \leq 1 - \xi_i \\ & \xi \geq 0,\ i = 1, \ldots, m. \\ & d_{\ p_a \otimes p_b} = 1, \forall\ d_{p_a} = 1 \wedge d_{p_b} = 1 \end{aligned} \tag{4.11}$$

The new constraints enforce that if two features can be concatenated into a long cascade, then the corresponding new cascade will be set to 1. The algorithm thus tends to select long patterns.

$$\begin{aligned} \max_{d} \quad & c_j(\alpha)^2 d_j \\ \text{s.t.} \quad & \sum_{j=1}^{m} d_j \leq B, 0 \leq d \leq 1, \\ & d_{\ p_a \otimes p_b} = 1, \forall\ d_{p_a} = 1 \wedge d_{p_b} = 1. \end{aligned} \tag{4.12}$$

Eq. (4.12) can be solved analytically. First, we construct a feasible solution by finding the largest scores $[c_j(\alpha)]^2$. Then, we set the scaler δ_j to 1 and the

Algorithm 4 Long-pattern classifier

Require:
Graph G, parameters C ,B, mini-batch size K
Ensure:
classifier c
1: $\alpha^0 \leftarrow 1/C \cdot \mathbf{1}$, $S \leftarrow \emptyset$, $t \leftarrow 0$
2: **repeat**
3: $S \leftarrow$ a mini-batch of K features
4: Calculate candidate top-k features based on α_t using Eq. (4.9)
// *mini − batch scores*
5: Calculate α_{t+1} based on d_t using Eq. (4.4)
// *Optimization* 1
6: $t \leftarrow t + 1$
7: **until** no feature left
8: Output $c \leftarrow SVM(S, C)$

remaining to 0.

Table 4.1 shows three different types of result when a long-pattern subgraph p_{ab} is generated from two short-pattern subgraph fragments p_a and p_b. For example, consider the four training graphs $g_1, \cdots, g_4$, where g_1 and g_2 belong to the same group, while g_3 and g_4 fall into another group. Assume we have obtained two subgraph fragments $P(a) = A \rightarrow B$ and $P(b) = B \rightarrow C$. We can generate a long-pattern subgraph $A \rightarrow B \rightarrow C$ based on Definition 3. The classifier may have three different types of result, Equal, Improve and Reduce. Therefore, we have the following intuitive conclusion.

Theorem 2. Consider two subgraph fragments $p_a = 1$ and $p_b = 1$, if $p_{ab} = p_a \otimes p_b$, then the generated long-pattern subgraphs p_{ab} can be used to replace the original two patterns p_a and p_b.

Proof: Evidently, if $p_{ab} = p_a \otimes p_b$, then $p_{ab} = 1$ is the solution of Eq. (4.12). □

Based on the above analysis, we design the long-pattern driven incremental feature selection algorithm given in Algorithm 3. The algorithm processes data in mini-batches. The algorithm examines the concatenation of short-pattern subgraphs and generates long-pattern subgraphs that comply with Theorem 2. The complexity of the algorithm is still $o(m)$ in the worst case.

Table 4.1: Analysis of the new constraint

	Equal			Improve			Reduce		
X	p_a	P_b	p_{ab}	p_a	P_b	p_{ab}	p_a	P_b	p_{ab}
x_1	1	1	1	1	1	1	1	0	0
x_2	1	1	1	1	1	1	1	0	0
x_3	0	0	0	1	0	0	0	1	0
x_4	0	0	0	0	1	1	0	1	0

4.4 Analysis

Algorithm 2 is equivalent to the *cutting-plane algorithm* for solving the semi-infinite programming problem. In each iteration, the cutting-plane algorithm removes nonactive constraints that correspond to redundant features in the primal problem. The algorithm iteratively finds active constraints that heavily violate the KKT condition. In the following, we prove the convergence of Algorithm 2.

Theorem 3. *Let* $\{\boldsymbol{\alpha}_t, \boldsymbol{d}_t\}$ *be a sequence of solutions generated by Algorithm 2. If Algorithm 1 stops at iteration* $\{t+1\}$*, then* $\{\boldsymbol{\alpha}_t, \boldsymbol{d}_t\}$ *is the global optimal solution of Eq.(4.3).*

Proof: The objective function of Eq. (4.3) is convex *w.r.t.* $\boldsymbol{\alpha}$ and linear *w.r.t.* $\mathbf{d}$. Thus, Eq. (4.3) has a global optimum. The algorithm, by iteratively solving Eqs. (6) and (7), will converge to the global optimum. □

The stop criterion is based on the bound of $\mathbf{d}$. Specifically, Algorithm 2 stops when $\mathbf{d}$ becomes stable. In each iteration, the algorithm scans the entire feature space and chooses B features. In the worst case, the algorithm iterates m/B times and selects all m features, *i.e.*, the algorithm takes $O(m^2)$ time at worst. The square time complexity is, unfortunately, unaffordable for ultra-high dimensional data, which is often the case in our problem setting.

The incremental feature selection is motivated by the limitations of Algorithm 2. First, the algorithm needs to fully scan all the features in each iteration, which requires $O(m^2)$ complexity in the worst case and is therefore unsuitable for big networks. Second, the number of selected features tB increases as the number of

iterations t continue, which may become high dimensional. Therefore, we propose a new primal-dual incremental feature selection method.

Algorithm 3 is based on the observation that the optimal solution for the offline linear program is almost entirely determined by the optimal dual solution corresponding to the inequality constraints. The algorithm is $1-O(\theta)$-competitive and the results can be stated as follows:

Because optimization 1 can be solved by using convex quadratic programming, there is polynomial time interior point algorithm in Matlab. The primal-dual incremental feature selection selects a mini-batch and only calculates the scores on the mini-batch. The algorithm firstly selects m features, and then calculates B features which has highest score, and it scans the feature set only once. Therefore, the time complexity of the algorithm is *O(m*BlogB)*. Since B is a small value, the time complexity can be a linear time complexity *O(m)*.

4.5 Experiments

We test the proposed algorithms on two real-world networks and four synthetic network data sets. The purpose of the experiments is to: 1) conduct a parameter study for the purpose of choosing optimal parameter settings for our experiments; 2) compare the proposed algorithms with benchmark methods to validate the performance of our method; and 3) test our method on real-world social network applications.

The source codes and data sets are available online[1]. The frequent subgraph mining is implemented in Java, the feature generation is implemented in Python, and the optimization algorithms are implemented in Matlab. All experiments are tested on a Linux Ubuntu server with 16*2.9GHz CPU and 64G memory.

4.5.1 Data Sets

We use two real-world networks and four synthetic data sets for testing. The data sets are summarized in Table 4.2.

[1]https://github.com/BlindReview/streaming for the source codes and the synthetic data sets. http://snap.stanford.edu/infopath/ for the cascade data set.

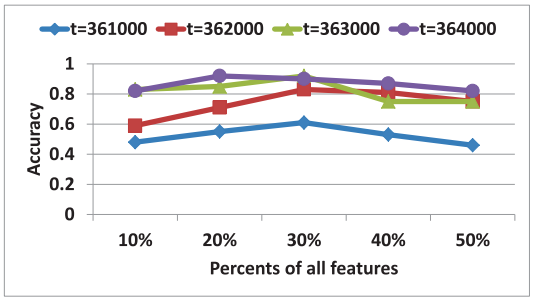
Table 4.2: List of the synthetic and real-world data sets.

Data Set	Nodes	Edges / Cascades	Other parameters
Albert-Barabasi	5000	19990	$n = 4$
Forest Fire	5000	21124	$f = 0.35, b = 0.32$
Small World	5000	19996	$\alpha = 4, p = 0.1$
Erdos-Renyi	5000	6000	$[0.5, 0.5; 0.5, 0.5]$
MemeTracker	3.3 mil.	27559952	\
DBLP	2000	95411	\

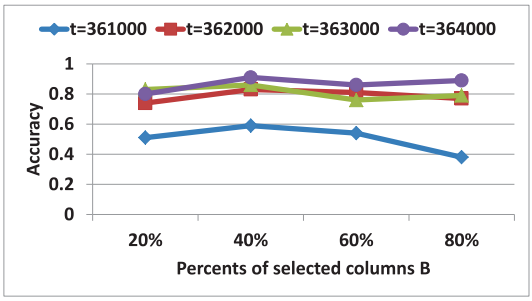
4.5.1.1 Real-world Data

MemeTracker: MemeTracker data set [65] is downloaded from the SNAP cascade data website, and is the topic-based MemeTracker containing 27.6 million news articles and blog posts from 3.3 million online sources with time-stamps over a one-year period, from March 2011 to February 2012. The data format is as follows, $<meme\ id>; <website\ id>, <timestamp>, ..., <website\ id>, <timestamp>$. The time-stamp indicates the information arrival time of a node from its parent nodes. We generate information propagation graphs as shown in Fig. 1. We treat each website as a graph node, there is an edge between two nodes if a website forward articles or blogs from another website. Thus, the propagation network forms a graph at a speicific observation time stamp (*e.g.*, 361000(s)). All the graphs at different time stamps in a cascade have the same label with the cascade (outbreak or non-outbreak). Predicting each graph label forms a graph classification task.

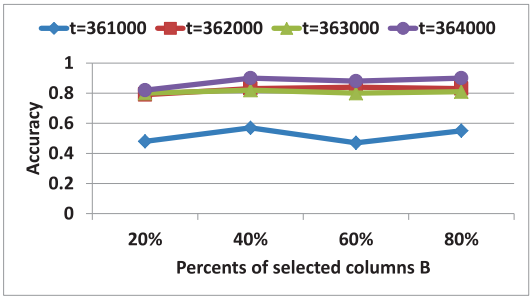
DBLP: DBLP author classification refers to the task of predicting author's research areas based on co-authorship network where two authors are connected if they publish at least one paper together. We retrieve around 2000 authors from DBLP and 95411 coauthors. The training sets are drawn from different domain, *i.e.*, the conferences in each research areas. We mainly crawl the authors from the areas in Software Engineering and Data Mining. We fetch the co-authorship network graphs of each author from 2001 to 2015, and formulate the co-authorship graphs at different year. Specifically, we have three phases' graphs, *i.e.*, 2001 to 2005, 2006 to 2010, and 2011 to 2015. Subgraph features are commonly used in author classification that is based on co-authorships. However, predicting an authors research areas using only a single coauthor or conference is infeasible due to the interdisciplinary nature of research.



(i) ISF: accuracy *w.r.t.* the observed features.



(ii) OSF: accuracy *w.r.t.* the selected columns B.



(iii) ISF: accuracy *w.r.t.* the selected columns B.

Figure 4.5: Parameter study on min-batch size B at each iteration and value k in top_k.

4.5.1.2 Synthetic Data

We use four well-known models to generate synthetic networks for testing and comparison, namely, the *Erdos Renyi [14], Albert Barabasi [10], Forest Fire [66] and Small World [133]* models.

Erdos Renyi [14] generates random graphs with arbitrary degree distributions. Each edge is included in the graph with a probability p independent of other edges. All graphs with N nodes and L edges have equal probability of $p^L(1-p)^{\binom{N}{2}-L}$. The parameter p is a weighting function. In particular, $p = 0.5$ corresponds to the case in which all $2^{\binom{N}{2}}$ graphs on N vertices are chosen with equal probability.

Albert-Barabasi (Scale-free network) [10] generates random scale-free networks using a preferential attachment mechanism. The network begins with an initial connected network containing β_0 nodes. New nodes are added to the network one at a time. Each new node is connected to $\beta \leq \beta_0$ existing nodes with a

Table 4.3: F1 score under the parameter support=30 on the four synthetic data sets.

Data Sets	Time	OSF	RSF	MSF	ISF	ISJF
Albert-Barabasi	100	**0.711±0.146**	0.611±0.070	0.564±0.019	0.682±0.077	0.651±0.126
	300	0.762±0.077	0.700±0.094	0.743±0.026	0.762±0.027	**0.791±0.102**
	500	**0.885±0.037**	0.739±0.094	0.664±0.109	0.785±0.094	0.863±0.082
	700	0.903±0.006	0.667±0.059	0.782±0.011	**0.912±0.028**	0.910±0.006
Erdos Renyi	100	0.750±0.125	0.667±0.178	0.712±0.169	**0.750±0.067**	0.750±0.125
	300	0.824±0.122	0.703±0.067	0.624±0.058	0.817±0.058	**0.824±0.058**
	500	0.807±0.122	0.798±0.044	0.766±0.044	0.815±0.100	**0.889±0.044**
	700	0.889±0.104	0.796±0.114	0.813±0.044	**0.889±0.103**	0.824±0.181
Forest Fire	100	0.891±0.007	0.793±0.027	0.796±0.131	**0.910±0.067**	0.891±0.017
	300	**0.880±0.004**	0.799±0.035	0.761±0.021	0.846±0.001	0.873±0.027
	500	**0.897±0.002**	0.826±0.044	0.801±0.061	0.885±0.030	0.849±0.021
	700	0.879±0.105	**0.916±0.08**	0.862±0.065	0.889±0.117	0.870±0.121
Small-world	100	0.526±0.005	0.404±0.003	0.470±0.004	**0.579±0.008**	0.406±0.013
	300	0.563±0.007	0.484±0.009	0.519±0.006	0.585±0.016	**0.592±0.009**
	500	0.746±0.012	0.498±0.005	0.479±0.007	0.595±0.007	**0.746±0.003**
	700	0.699±0.021	0.552±0.107	0.519±0.003	0.760±0.143	**0.763±0.006**

probability proportional to the number of links that existing nodes already have. For this model, we need to set parameter n, which denotes the number of edges created by each new node.

Small-world (Watts-Strogatz model) [133] is defined as a network in which the typical distance ζ between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes N in the network, that is $\zeta \propto \log N$. We use parameter α to denote that each node is connected to α nearest neighbors in topology, and p denotes the rewiring probability.

Forest Fire model (Scale-free network) [66] is defined as a cellular automaton on a grid with γ^d cells. γ is the side-length of the grid and d is its dimension. In this model, we need to set the parameters of the forward burning probability f, and the backward burning probability b.

To better simulate real-world network diffusion, we generate synthetic networks under a power-law degree distribution exponent $\alpha = 1.5$, which corresponds to the power-law degree exponent of the MemeTracker network.

Synthetic cascades are generated using the following methods. First, we ran-

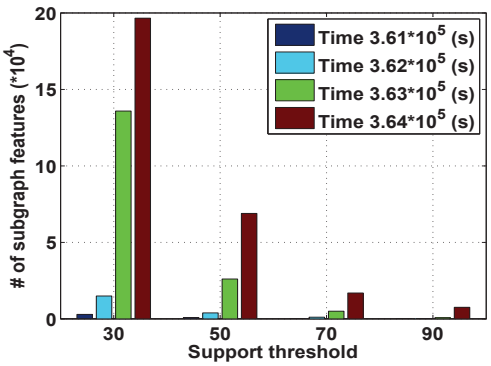
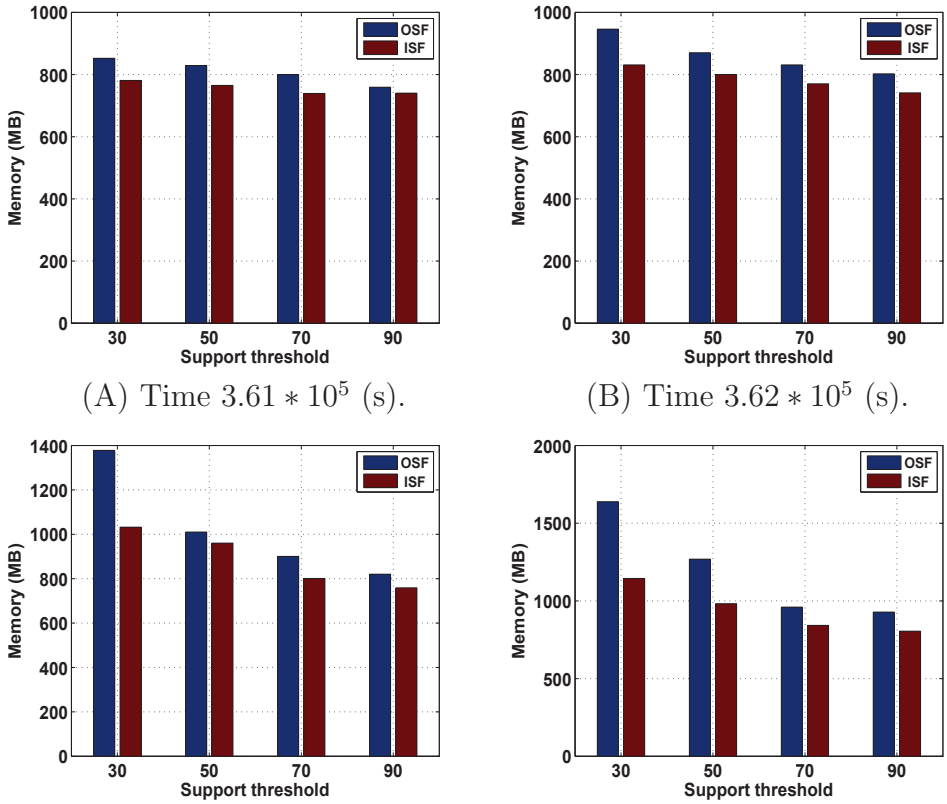


Figure 4.6: # of subgraph features *w.r.t.* support threshold on the MemeTracker data set.

domly select a root node r with a non-zero out-degree. The node r is then added to the initially empty list of the infected nodes I and all outgoing edges (r, s) are added to the initially empty FIFO queue of the infected-susceptible node pairs S. We choose an edge from the candidate set each time and calculate the time delay for the edge until the time delay exceeds a given time window (we use [0,1000] as the time window). The data are generated by repeating the above steps for each source node.

In the synthetic networks, nodes are continuously included in the network, we separate each cascade by different time stamp (*e.g.*, 100, 200), and each network at a specific time stamp (*e.g.*, time is 100) can be treated as a graph. All the graphs in a cascade have the same label with the cascade (outbreak or non-outbreak). We aim to classify outbreak cascades from non-outbreak ones by using graph classification.

Benchmark methods. We compare the proposed *ISF* and the Incremental Subgraph Join Feature selection algorithm (ISJF) with the following three methods: 1) Meta Subgraph Feature selection (*MSF*) which randomly selects meta subgraphs (nodes) as features for graph classification [22]. In MSF, each node of a graph is taken as a feature; 2) Off-line Subgraph Feature selection (*OSF*) [88] which loads all features into memory at one time and selects top_k columns (with top k scores) at each iteration until the condition $\|\alpha_{t+1} - \alpha_t\| < \epsilon$ is met, where $\epsilon = 0.001$ in our experiments; 3) Random Subgraph Feature selection (*RSF*) [142] which randomly selects B features for graph classification.



(A) Time $3.61 * 10^5$ (s). (B) Time $3.62 * 10^5$ (s).

(C) Time $3.63 * 10^5$ (s). (D) Time $3.64 * 10^5$ (s).

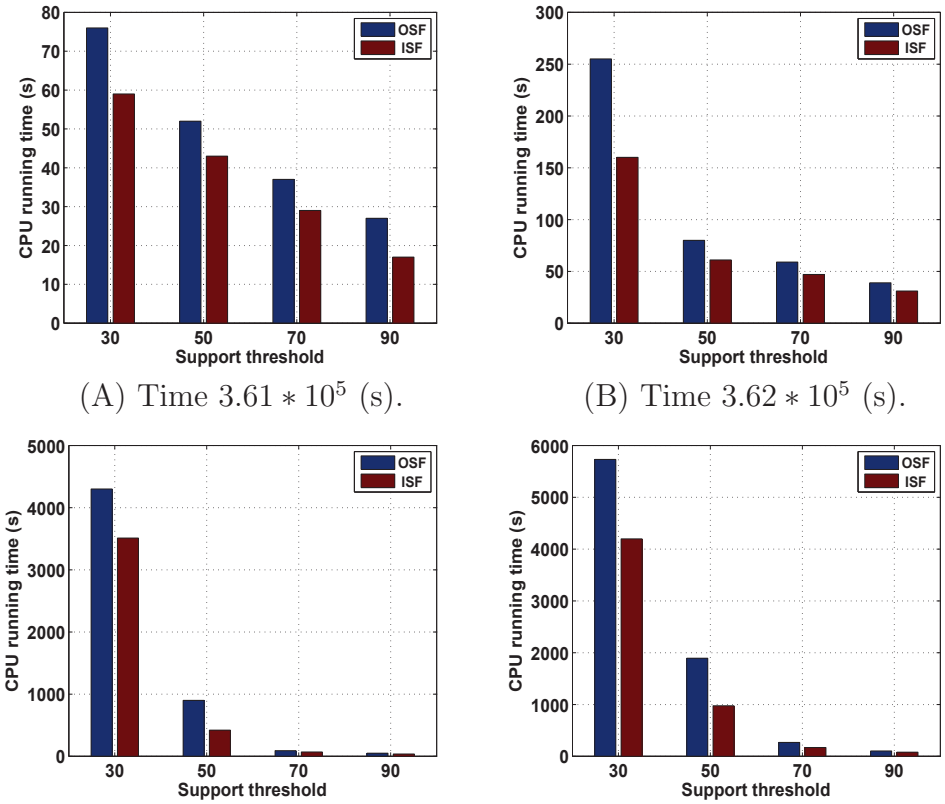
Figure 4.7: Memory cost *w.r.t.* the support threshold *Supp* under different propagation time stamps on the MemeTracker data set.

Measures. We compare the algorithms with respect to running time, prediction accuracy (*tp / (tp + fp + fn + tn)*), precision(*tp / (tp + fp)*), recall (*tp / (tp + fn)*), and the F1 Score (*2 * (precision * recall / (precision + recall))*) to evaluate classification performance, where *tp* is true positive, *fp* is false positive, *fn* is false negative, and *tn* is true negative.

4.5.2 Parameter study

We first test the parameters in Eq. (4.8) *w.r.t.* the number of features B, and the parameter k of the top_k features in OSF and ISF.

Fig. 4.5 shows the performance of the two algorithms under different parameter settings on the real-world data.



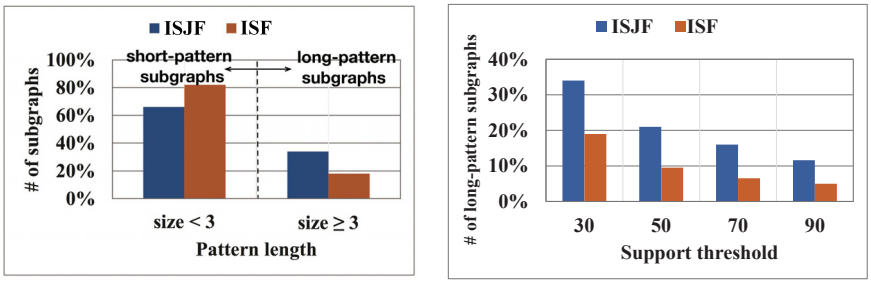
(A) Time $3.61 * 10^5$ (s). (B) Time $3.62 * 10^5$ (s).

(C) Time $3.63 * 10^5$ (s). (D) Time $3.64 * 10^5$ (s).

Figure 4.8: Running time *w.r.t.* the support threshold *Supp* under different propagation time stamps on the MemeTracker data set.

The mini-batch size B: $B > 0$ represents the portion of selected features *w.r.t.* the number of features at each iteration. If B is too large, the algorithm suffers from large computation and memory cost. In the worst case, B equals to the number of all features which degenerate to the off-line method. Fig. 4.5 shows that the best prediction accuracy is achieved when B is 30% of all features.

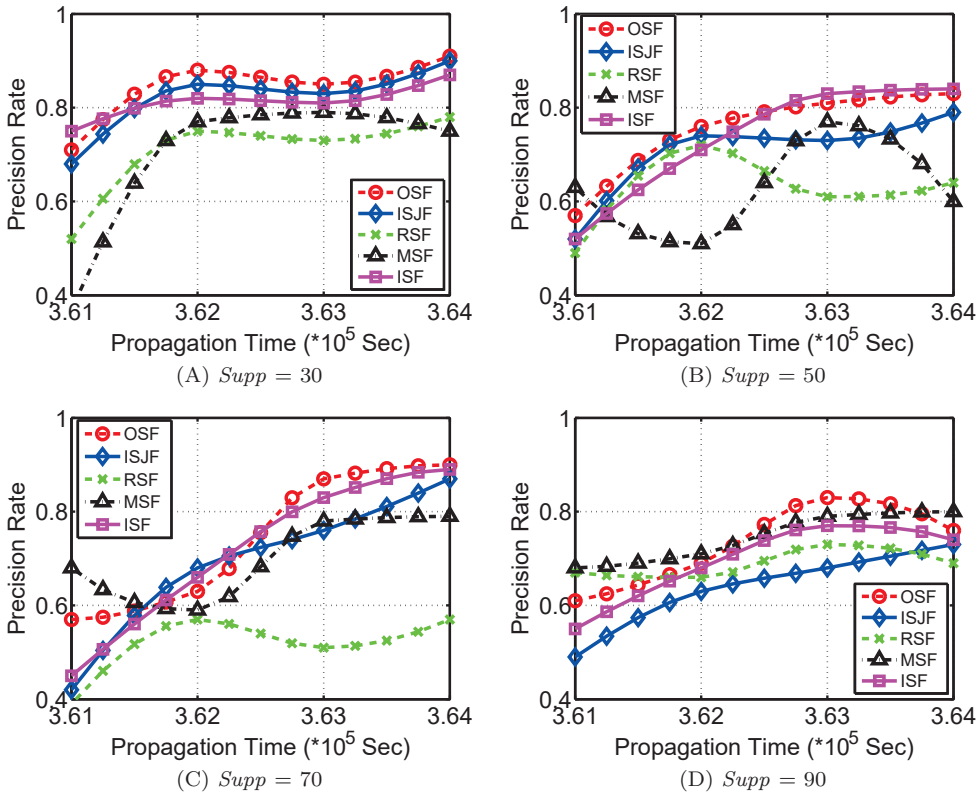
The parameter top_k: We use the optimal value of $B = 30\%$ to study parameter k, which indicates the number of features being selected for the next mini-batch of feature selection. Fig. 4.5 shows that both OSF and ISF have the highest accuracy when k equals to 40% of mini-batch B.



(i) # of subgraph patterns *w.r.t.* pattern length.

(ii) # of long-pattern subgraphs *w.r.t.* support threshold.

Figure 4.9: Percentage of patterns *w.r.t.* the support threshold and pattern length.



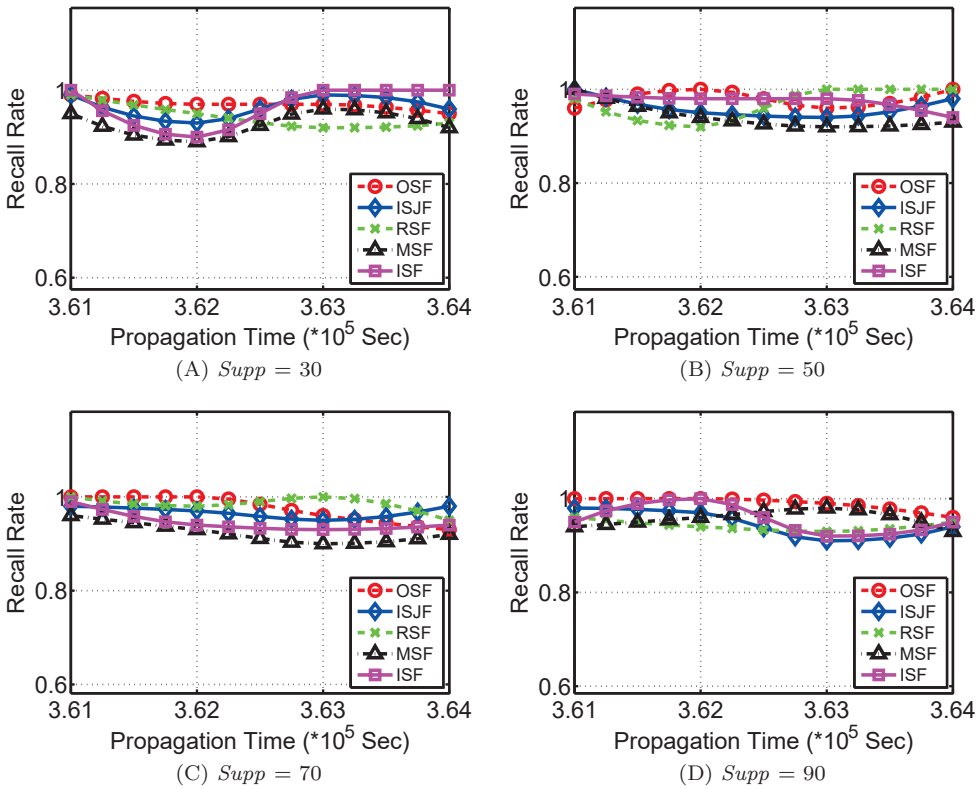
(A) *Supp* = 30

(B) *Supp* = 50

(C) *Supp* = 70

(D) *Supp* = 90

Figure 4.10: Precision comparison under different *Supp* on the MemeTracker data set.



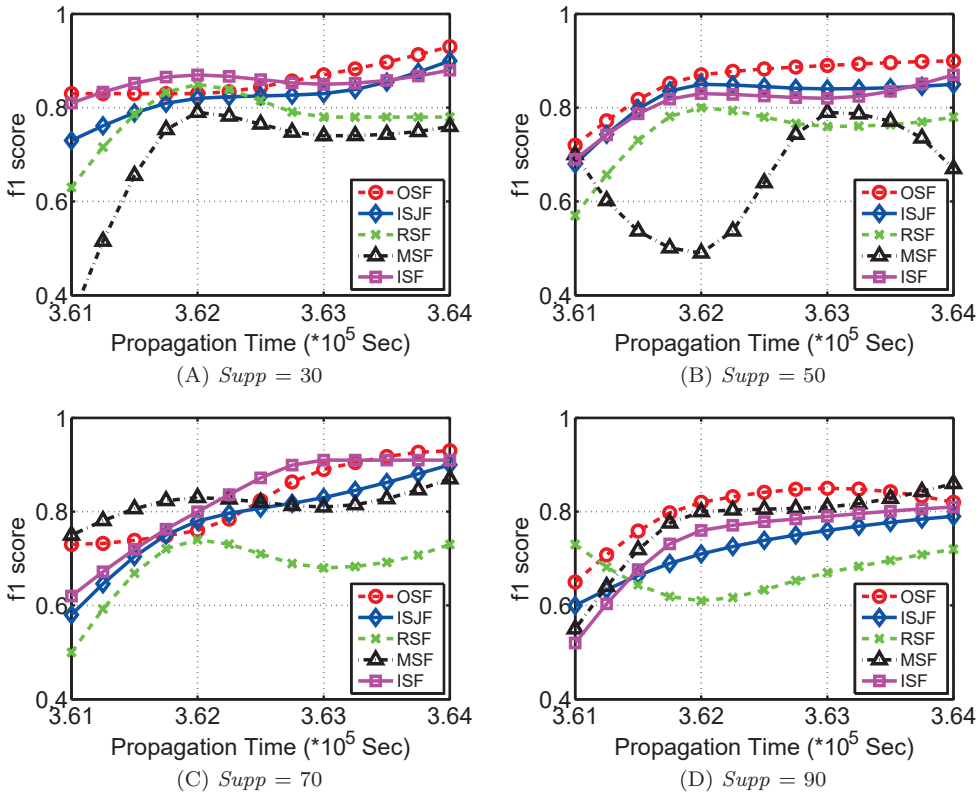
(A) $Supp = 30$ (B) $Supp = 50$

(C) $Supp = 70$ (D) $Supp = 90$

Figure 4.11: Recall comparison under different *Supp* on the MemeTracker data set.

4.5.3 Experimental Results

The dimension of subgraph features and running time on the MemeTracker data set: Fig. 4.6 shows the dimension of subgraph features *w.r.t.* the support threshold *Supp*. Figs. 4.7 and 4.8 show the memory consumption and running time comparisons for ISF and OSF *w.r.t.* the support threshold under different propagation time stamps. The different propagation time stamps and support thresholds indicate the different dimensions of the subgraph features. The number of features approximates to $2 * 10^5$ when the support threshold is 30 and the propagation time reaches 364,000. Figs. 4.7 and 4.8 show that our ISF algorithm can handle high dimensional features faster than OSF. This is because ISF uses a primal-dual subgraph feature selection which continuously selects subgraph features to quickly solve the graph classifier. In contrast, OSF loads all features into



(A) *Supp* = 30 (B) *Supp* = 50

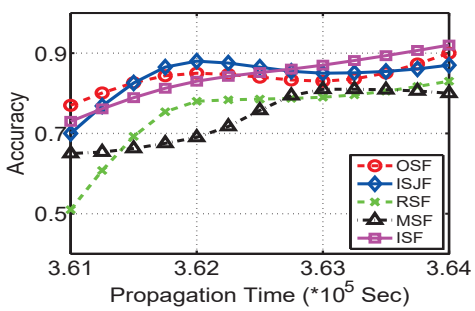
(C) *Supp* = 70 (D) *Supp* = 90

Figure 4.12: F1 score comparison under different *Supp* on the MemeTracker data set.

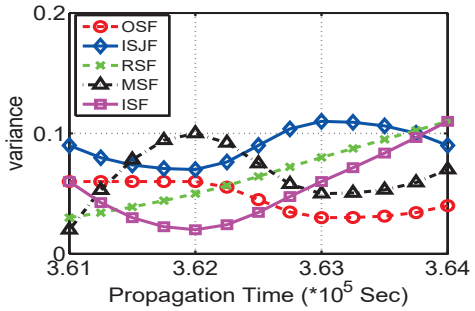
memory at one time which costs heavy in both memory and time.

Prediction Accuracy: The five algorithms are compared under different support thresholds (30, 50, 70 and 90) in the subgraph mining. The settings for the time stamps are 3.61, 3.62, 3.63 and 3.64 * 10^5 seconds. We report the performance of ISF, ISJF and three benchmarks on the MemeTracker data set in Figs. 4.10 - 4.13 and on the synthetic data sets listed in Table 4.3. Fig. 4.9(i) shows the proportion of short-pattern subgraph features (length < 3) and long-pattern subgraph features (length ≥ 3) when the support is equal to 30. Fig. 4.9 (ii) shows the comparison of the long-pattern subgraphs. From these results, we make the following observations.

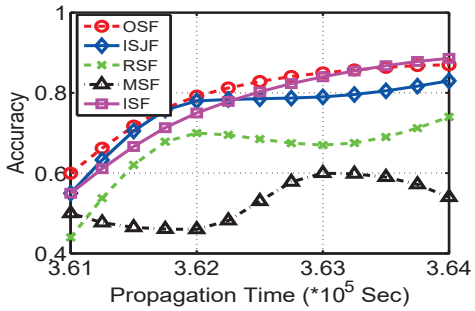
1) The classification precision and F1 score increase with propagation time. This is because more nodes and paths in the cascade graphs are available for classification as time increases. Because the incremental cascade classifier selects



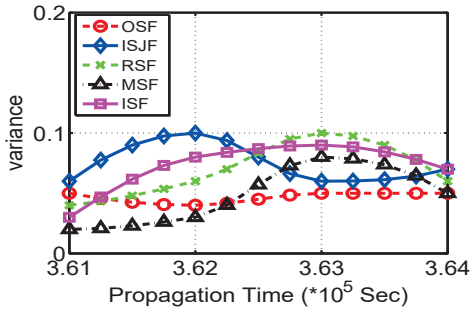
(A) accuracy *w.r.t.* propagation time ($supp = 30$)



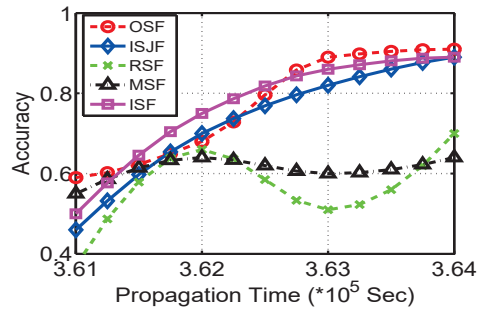
(B) variance *w.r.t.* propagation time ($supp = 30$)



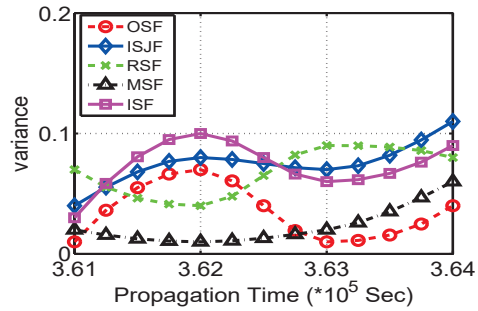
(C) accuracy *w.r.t.* propagation time($supp = 50$)



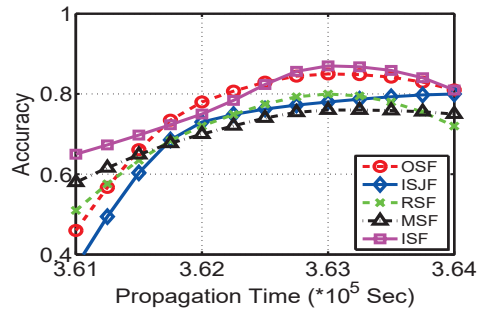
(D) variance *w.r.t.* propagation time ($supp = 50$)



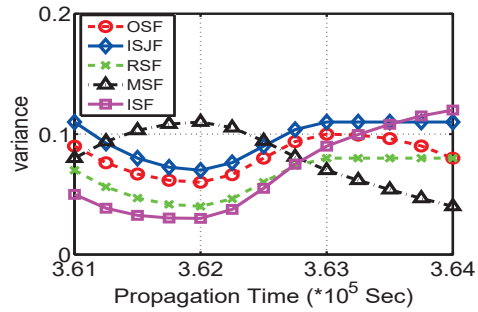
(E) accuracy *w.r.t* propagation time ($supp = 70$)



(F) variance *w.r.t.* propagation time ($supp = 70$)



(G) accuracy *w.r.t.* propagation time ($supp = 90$)



(H) variance *w.r.t.* propagation time ($supp = 90$)

Figure 4.13: Accuracy and variance comparisons *w.r.t.* time stamp on the Meme-Tracker data set.

top_k features at each iteration and uses the dual problem to solve the high dimensional problem, our method is more advantageous than the benchmark methods.

2) The precision, F1 score, and accuracy show that OSF outperforms other methods. This is because OSF always selects the top_k columns from all the features while ISF selects the top_k of $t * B \leqslant N$ features, where t is the number of iterations, B is the number of selected columns, and N is the number of all the used features.

3) In terms of prediction variance, OSF and ISF have lower variance error, which means they are more stable than the benchmark methods.

4) From the four synthetic data sets in Table 4.3, when the time is around 300-500, ISJF is usually better than ISF, because there are more short-pattern subgraphs and useful long-pattern subgraphs when the time is around 300-500. In addition, the long patterns are useful for small world data than scale free networks, as any vertices in the small world be connected by at most 6 vertices [151].

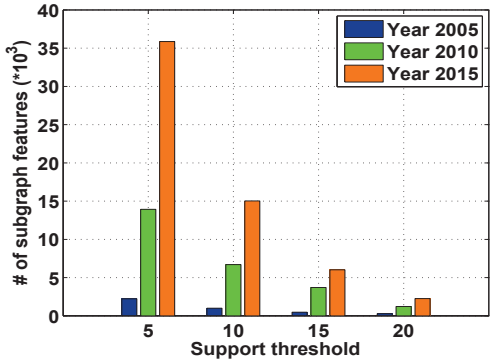
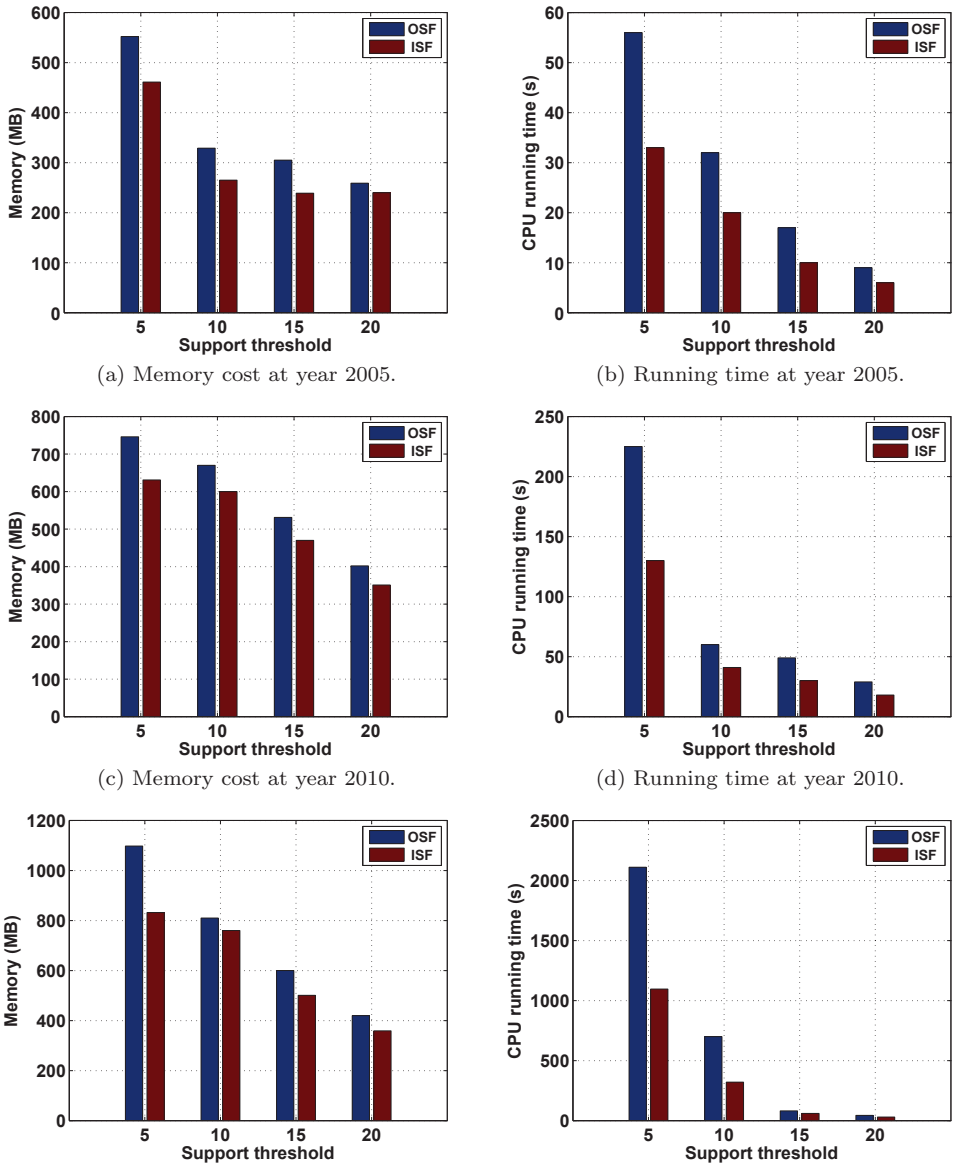


Figure 4.14: # of subgraph features *w.r.t.* support threshold on the DBLP data set.

Experiments on the DBLP data set: The number of subgraph features with respect to support threshold, the memory cost with respect to the support threshold, the running time with respect to the support threshold, and the accuracy comparison in each year bracket are reported in Figs. 4.14 - 4.16.

From the results on the DBLP data set, we make observations similar to the MemeTracker data set. The classification accuracy increases with each year bracket. This is because more nodes and paths in the co-authorship graph are



(a) Memory cost at year 2005.

(b) Running time at year 2005.

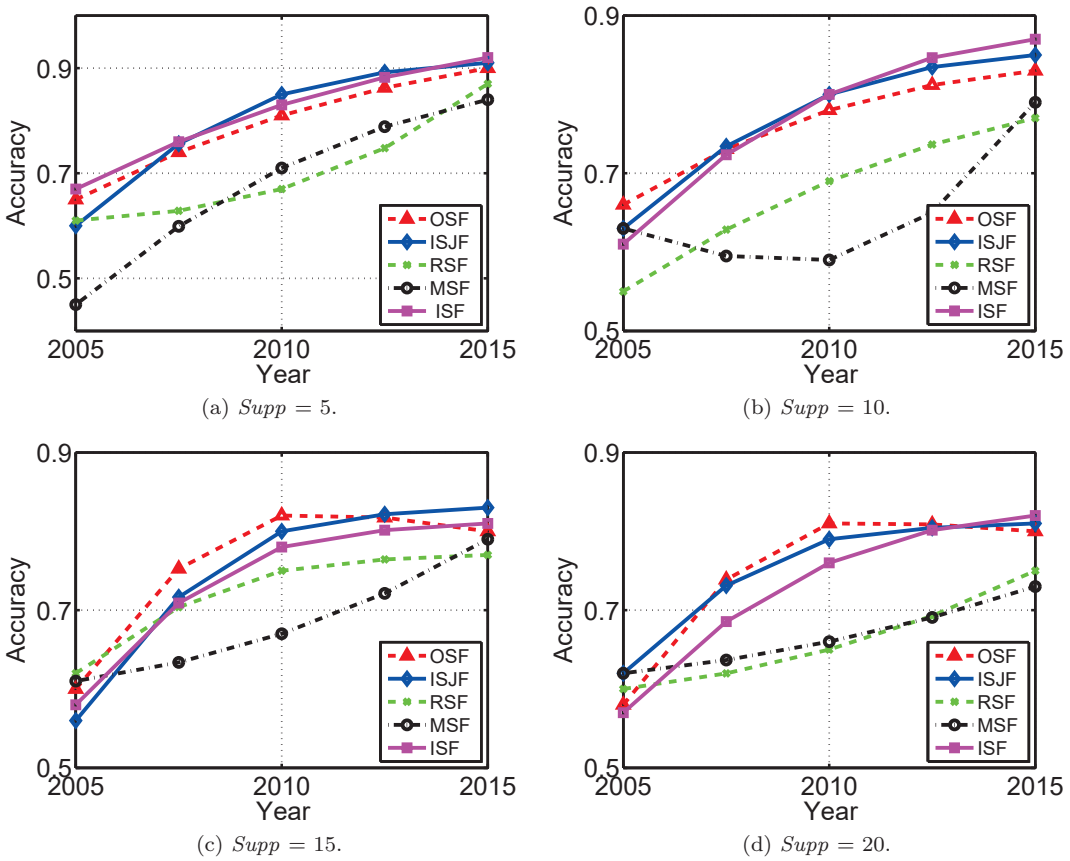
(c) Memory cost at year 2010.

(d) Running time at year 2010.

(e) Memory cost at year 2015.

(f) Running time at year 2015.

Figure 4.15: Memory cost and running time w.r.t. the support threshold at different year on the DBLP data set.



(a) *Supp* = 5. (b) *Supp* = 10.

(c) *Supp* = 15. (d) *Supp* = 20.

Figure 4.16: Accuracy comparison under different *Supp* on the DBLP data set.

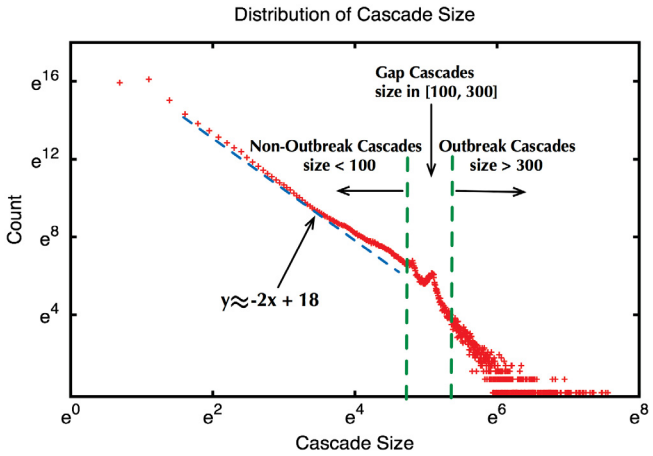


Figure 4.17: The probability distribution of cascades. The dotted line is the linear fitting result to the red curve, showing that the distribution fits the power-law. The two dotted vertical lines indicate the threshold which discriminates outbreaks from non-outbreak cascades. The sizes in [100, 300] are the gap cascades which are not used in our experiments.

available for classification as time goes by. The proposed approach also shows significant improvement with respect to the running time and memory cost, especially when the subgraphs' dimensions are high.

4.5.4 Case Study on Cascading Outbreak Early Prediction

We test the incremental subgraph features method on real-world social networks to predict outbreak cascades. Cascade data represents a new type of graph data that reflect information flows. Because there are no cascade descriptors (features) directly available to reflect the direction of information flows, we resort to subcascades as features. Existing works on cascade outbreak prediction are based on node features and **MSF** can be used as the solution. Fig. 1.2 (Introduction Chapter) samples several information cascades in a social network. Each cascade can be denoted as a graph, and subcascades correspond to subgraphs. For example, the cascade $n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_4$ is a propagation graph which contains a simple subgraph of $n_1 \rightarrow n_2 \rightarrow n_3$. If a cascade becomes an outbreak, we label it as the positive class +1; otherwise, -1. The subgraph features and class labels

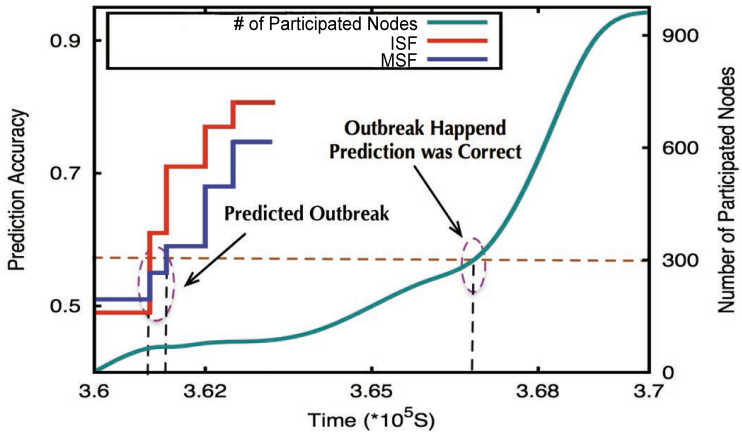


Figure 4.18: Early prediction of information cascade outbreaks. We compare the subcascade-based method (red line) with the node-based method (blue line). The figure shows that the subcascade-based method provides better prediction accuracy than the node-based method.

are used to build classifiers.

The problem of cascade outbreak prediction is defined as: *Given a network graph G, consider a cascade $\boldsymbol{x} = \{xID; \langle V, T \rangle\}$ where V is a network of nodes, T is a time-stamp, and a threshold parameter $\gamma \in (0, 1)$, if $|V| \geq \lceil \gamma |G| \rceil$, then x is labeled as outbreak, i.e., $y = +1$; otherwise, y=-1.*

Fig. 4.17 shows the log-log distribution of the cascade size and its node number. As shown in Fig. 4.17, the size of the MemeTracker cascade data set follows the power-law with long-tails, which indicates that only a small proportion of these cascade become outbreak cascades.

We select cascades having more than 300 nodes as outbreaks (877 positive examples), and cascades having less than 100 nodes as non-outbreaks (27,515,721 negative examples). We randomly select outbreak and non-outbreak examples to construct balanced training sets.

Fig. 4.18 shows that ISF outperforms MSF, which validates the superiority of the proposed subcascade-based outbreak prediction method. This is mainly because node features ignore the path (graph) information. For example, cascade $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$ will be predicted as the same class label because they share the same node features.

4.6 Conclusions

In this chapter, we study graph classification with incremental subgraph features. Based on the observation that subgraph features follow the *downward closure property* and long-pattern subgraph features are often buried underneath short-pattern subgraph features, we propose a *primal-dual incremental subgraph feature selection* algorithm (*ISF*) for mining incremental subgraph features, and a subgraph join feature selection algorithm (*ISJF*) to exact long-pattern subgraphs. The proposed *ISF* and *ISJF* algorithms split feature set and load into memory in a mini-batch manner, which is a significant reduction in memory and running time. Experiments show the *ISF* algorithms can reduce both the running time and memory cost. In the meanwhile, it has the comparable accuracy with offline algorithms. The incremental algorithm learns both primal and dual feasible solutions, and the dual problem converts a high dimensional problem into a big constraint problem with respect to dual vector. *ISJF* adds a new constraint on the max-margin graph classifier and forces the classifier to select long pattern subgraphs by joining short-pattern subgraph features. Compared with baseline method *OSF*, the different is that *OSF* is an offline algorithm and loads all features into memory at one time, while *ISF* does not have access to all features at a time. Experiments on real-world cascade outbreak prediction in social networks demonstrate the effectiveness of the proposed models.

Chapter 5

Graph-shapelets Feature based TVGLC

5.1 Introduction

In time-variant graph classification, frequent subgraph patterns are very difficult to find because it is necessary to consider both the graph structure space and the temporal correlations to find subgraph candidates for validation. The search space becomes infinite and hence unlikely to yield stable structures. Secondly, the distance between subgraph patterns may change irregularly over time, therefore even though a relatively large threshold parameter could be set to mine frequent subgraph patterns, the resulting set may be empty.

Although there are many works on graph stream classification [2, 90], these works typically assume that graphs are independent of one another and flow in a stream fashion. As a result, subgraph features are still applicable because the structure of each graph is static and the static graph distribution assumption and satisfactory empirical results have been reported [107]. However, in time-variant graph classification, the entire graph data changes over time and therefore obtaining stable subgraph features is unlikely. New types of graph patterns are needed as features for time-variant graph classification.

In this chapter, we outline a new class of graph patterns, called **graph-shapelet patterns**, that can be used as features for time-variant graph clas-

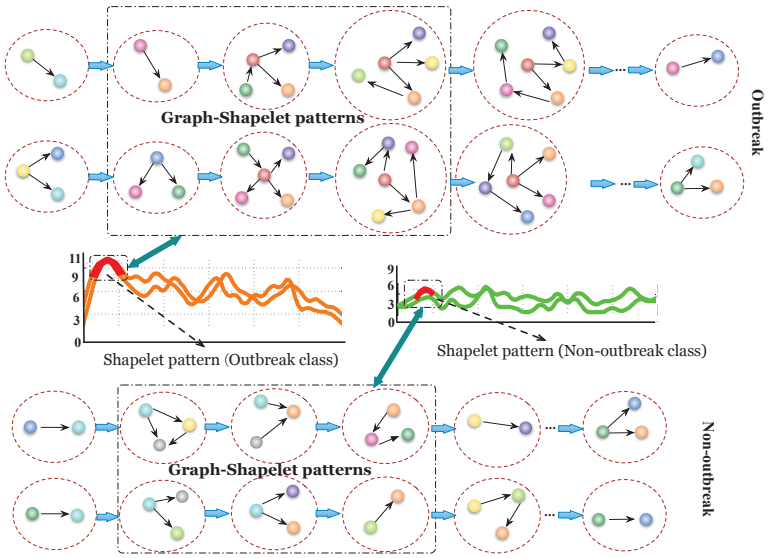


Figure 5.1: An illustration of graph-shapelet patterns. Graph-shapelet patterns are compact and discriminative graph transformation subsequences that describe the graph transformation patterns shared in the same class of time-variant graphs, *e.g.* two time-variant graphs (the two entire rows above) with the outbreak labels. When we explore a univariate time series from a time-variant graph, we can see that the location of a *graph-shapelet pattern* is consistent with that of a *shapelet* in time series (detailed in Section 5.4.1). In this case, graph-shapelet patterns can be used for time-variant graph prediction such as dynamic graph outbreak prediction.

sification. Graph-shapelet patterns are inspired by the *shapelets* in [147], which were proposed as discriminative features for time-variant data stream classification. Shapelets are compact and discriminative subsequences that significantly reduce the time and space required for time-variant data classification [81, 37]. However, all the existing time series issues fall into either univariate or multivariate problems. They ignore the structural information behind time sequences that can be represented as graphs.

Graph-shapelet patterns can be taken as a graphical extension of the traditional shapelets used in time series classification. From the time series classification viewpoint, our study extends the univariate and multi-variant time series data classification into time-variant graph classification where at each time stamp we obtain a graph instead of a single variable or a vector. As shown in

Fig. 5.1, graph-shapelet patterns are compact and discriminative graph transformation subsequences extracted from a sequence of graph data that describes graph transformation patterns. The graph-shapelet patterns look similar to the frequent graph subsequences proposed in graph sequence mining [44], and they are discriminative graph subsequences that can differentiate two classes of graph sequences with high accuracy. In contrast, frequent graph subsequences do not contain the class label information and thus are not helpful for time-variant graph classification.

The design of graph-shapelet patterns requires a graph sequential pattern mining to be grafted onto time series shapelet pattern mining. The main technical challenges include:

- *Challenge #1:* From the time series shapelet pattern mining viewpoint, existing shapelet pattern mining is only based on univariant/multi-variant time series data. Extending existing shapelet pattern mining algorithms to time-variant graphs is the first challenge, and requires downscaling a graph sequence into a univariant/multi-variant time series by designing a graph statistics, which can minimize information loss during the conversion of a graph into a univaraint/multivariant time series.

- *Challenge #2:* From the graph sequence mining viewpoint, existing graph sequence mining aims to find frequent transformation subsequences *directly* from graph sequences by using *graph edit similarity*, which measures the similarity between two neighboring graphs. However, such methods are unscalable to the large sets of graph sequences which commonly occur in dynamic graphs. Moreover, these graph sequence mining algorithms cannot always guarantee that the frequent graph transformation subsequences are *discriminative transformation subsequences* for graph classification. Therefore, how to design a scalable algorithm that can extend existing graph sequence mining algorithms to discover *discriminative transformation subsequences* is the second challenge.

- *Challenge #3:* From the application viewpoint, time-variant graphs have the potential to represent large and dynamic graph data. How to evaluate

the performance of the proposed method by collecting persuasive data sets and designing proper benchmark methods is the third challenge.

In this chapter, we propose a new class of graph patterns, *graph-shapelet patterns*, as features for time-variant graph classification [127]. Graph-shapelet patterns are compact and discriminative graph transformation subsequences that can be used to classify graph sequences. Technically, to solve these challenges, we first convert a graph sequence into a time series using fast graph statistics. Then, we extract shapelets from the converted time series by using a traditional shapelets mining algorithms. Next, we locate the graph subsequences that match the shapelets from the original graph sequences. Each *graph subsequence* corresponds to a unique *graph transformation subsequence* based on our new graph transformation rules. At the last step, we extract the most discriminative *graph transformation subsequences* as graph-shapelets based on their graph edit similarity. Experiments on both synthetic and real-world data demonstrate the performance of the proposed algorithms.

The remainder of the chatper is organized as follows. Section 5.2 presents the preliminaries with key definitions. An overall framework of graph-shapelet based TVGLC is demonstrated in Section 5.3. Section 5.4 introduces the proposed graph-shapelet pattern based classifier. The proposed algorithm is shown in Section 5.5. Section 5.6 reports the experimental results, with discussion being conducted in Section 5.7. We conclude this chapter in Section 5.8.

5.2 Problem Formulation and Preliminaries

5.2.1 Problem Definition

Definition 4. *(*Graph*) For a time-variant graph g_i, each graph $g_i^{(j)}$ is represented as $g_i^{(j)} = (\mathcal{V}, E)$, where $\mathcal{V} = \{v_1, \cdots, v_z\}$ is a set of vertices, $E = \{(v, v') | (v, v') \in \mathcal{V} \times \mathcal{V}\}$ is a set of edges. $\mathcal{V}(g_i^{(j)}), E(g_i^{(j)})$ are sets of nodes and edges of $g_i^{(j)}$.*

Definition 5. *(*Time-Variant Graph*) A time-variant graph (i.e., graph sequence) is represented by $g_i = \langle g_i^{(1)} \cdots g_i^{(j)} \cdots g_i^{(m)} \rangle$, where the superscript represents the*

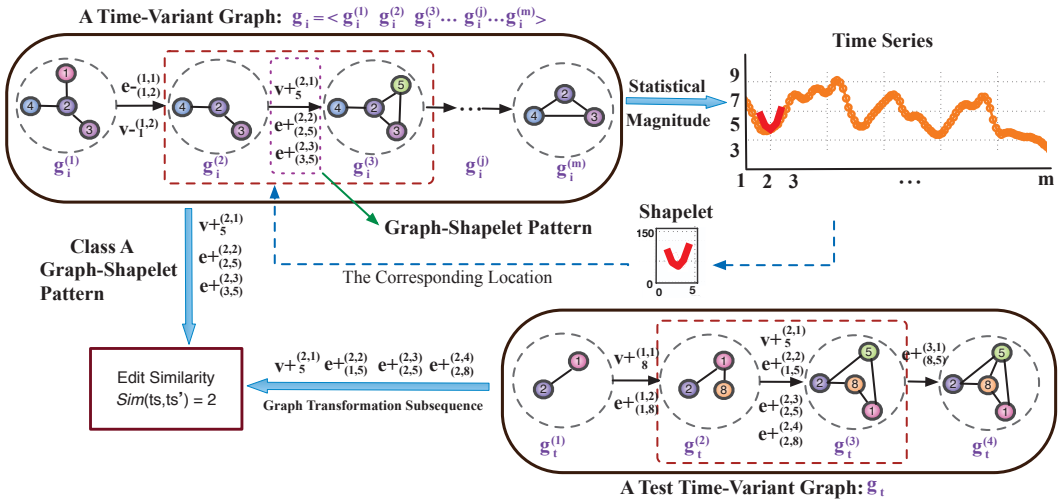


Figure 5.2: A toy example of graph-shapelet pattern mining to explain the definitions of related operations. We first explore univariate time series from a set of time-variant graphs (detailed in Section 5.4.1), where shapelets are discovered using shapelet pattern mining algorithms. The graph-shapelet patterns (discriminative graph transformation sequences) are then relocated by calculating the graph edit similarity (as in Definition 8) between shapelet patterns which match a set of graph subsequences. To make the above illustration clear, we use this example through *Examples* 1 $\sim$ 4.

temporal order of the graph (e.g., $g_i^{(1)}$*) in the sequence, and each time-variant* g_i *is assigned a label* L_i*.*

Definition 6. *(*Sub-Time-Variant Graph*) For a time-variant graph (i.e., graph sequence)* $g_i = \langle g_i^{(1)} \cdots g_i^{(m)} \rangle$*, a sub-time-variant graph* $g_i' = \langle g_i^{(k)} \cdots g_i^{(m')} \rangle$ *is a fragment of a time-variant graph, where* $1 \leq k \leq m' \leq m$*.*

Given a set of time-variant graphs $\mathcal{G} = \{g_1, \cdots, g_n\}$, time-variant graph classification **aims** to learn classification models from $\mathcal{G}$ to accurately predict previously unseen time-variant graphs with maximum accuracy. For simplicity, we only consider binary time-variant graph classification tasks.

5.2.2 Preliminaries

In this subsection, we formally define notations and introduce time series data and graph transformation subsequences. We state the distance between the time-

variant graphs and the two approaches to convert the time-variant graphs into graph transformation sequences. Fig. 5.2 shows a toy example of the given definitions and the proposed approaches. The main operations used in the graph dynamic changes are defined in Table 5.1.

In a time-variant graph g_i, we assume that nodes v and v' are exclusive to each other in $g_i^{(j)}$. We define a set of unique IDs $ID(\mathcal{V})$ and pairs of unique IDs $ID(E)$ as:

$$ID(\mathcal{V}) = \{id(v)|v \in \cup_{g_i^{(j)} \in g_i} \mathcal{V}(g_i^{(j)})\},$$
$$ID(E) = \{(id(v), id(v'))|(v, v') \in \cup_{g_i^{(j)} \in g_i} E(g_i^{(j)})\}.$$

Table 5.1: Operation Definitions

$v+_u^{(j,k)}$	$\neg\exists v \in \mathcal{V}(g_i^{(j)})\ s.t.\ id(v) = u$ and $\exists v' \in \mathcal{V}(g_i^{(j+1)})\ s.t.\ id(v') = u\}$
$v-_u^{(j,k)}$	$\exists v \in \mathcal{V}(g_i^{k})\ s.t.\ id(v) = u,$ $\neg\exists (v_1, v_2) \in E(g_i^{(k)})\ s.t.\ (id(v_1) = u) \vee (id(v_2) = u)$ and $\neg\exists v' \in \mathcal{V}(g_i^{k+1})\ s.t.\ id(v') = u\}$
$e+_{(u_1,u_2)}^{(j,k)}$	$\neg\exists (v_1, v_2) \in E(g_i^{(j)}), \exists v_1 \in \mathcal{V}(g_i^{(j)}),$ $\exists v_2 \in \mathcal{V}(g_i^{(j)})\ s.t.(id(v_1) = u_1) \wedge (id(v_2) = u_2)$ and $\exists (v_1', v_2') \in E(g_i^{(j+1)}) s.t.(id(v_1' = u_1) \wedge (id(v_2' = u_2)$
$e-_{(u_1,u_2)}^{(j,k)}$	$\neg\exists (v_1, v_2) \in E(g_i^{(j)}), s.t.(id(v_1) = u_1) \wedge (id(v_2) = u_2)$ and $\neg\exists (v_1', v_2') \in E(g_i^{(j+1)}), \exists v_1' \in \mathcal{V}(g_i^{j+1}),$ $\exists v_2' \in \mathcal{V}(g_i^{(j+1)})\ s.t.(id(v_1' = u_1) \wedge (id(v_2' = u_2)$

$*$: k is the index of operation in a transformation sequence.

A transformation from $g_i^{(k)}$ to $g_i^{(k+1)}$ is represented by $op_{g_i}^k$, and $op_{g_i}^k = \langle op_{g_i}^{(k,1)}, \cdots, op_{g_i}^{(k,l_k)} \rangle$ with l_k denoting the length of $op_{g_i}^k$, is a series of transformations such as insertions and deletions of nodes and edges between neighboring graphs, *e.g.*, $v+, e+, e-, v-$.

Definition 7. *(*Graph Transformation Sequence and Graph Transformation Subsequence*) Given a time-variant graph $g_i = \langle g_i^{(1)} \cdots g_i^{(m)} \rangle$, the corresponding graph transformation sequence can be denoted as $ts(g_i) = \langle op_{g_i}^1 \cdots op_{g_i}^{(m-1)} \rangle$, with $ts(g_i') =$*

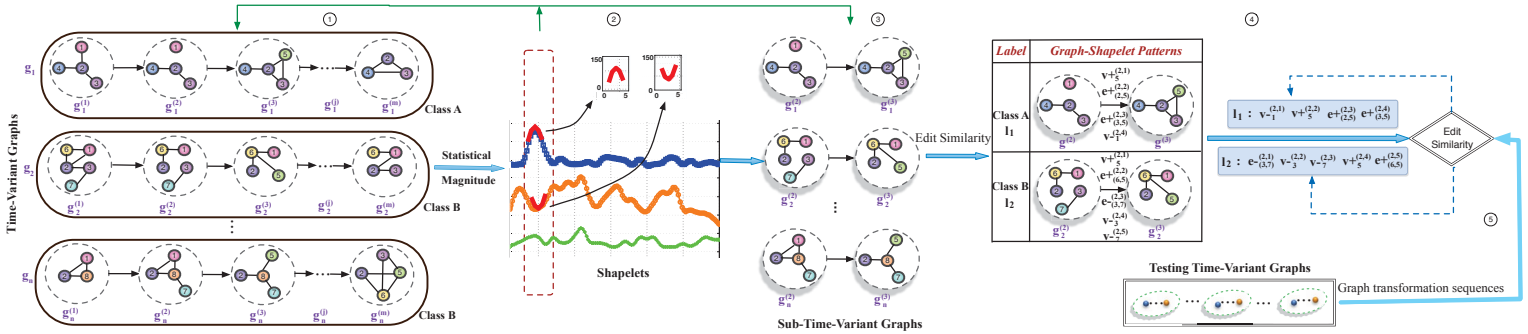


Figure 5.3: A concept view of the proposed time-variant graph classification framework. We first explore univariate time series from time-variant graphs via a sample kernel method in each graph (step ①, Section 5.4.1). Then, we find shapelet patterns from the time series by using shapelet pattern mining (step ②, Section 5.4.2). Next, we locate the sub-time-variant graphs that match the shapelet patterns from the original time-variant graphs (step ③, Section 5.4.3). Note that each sub-time-variant graph corresponds to a unique graph transformation subsequence by the proposed time-variant graph representation approach. At the last step, we calculate the graph edit similarity between graph transformation subsequences and find the most discriminative transformation subsequences as graph-shapelet patterns (step ④, 5.5.1). Step ⑤ shows the process of time-variant graph prediction.

$\langle op^1_{g'_i} \cdots op^{(m'-1)}_{g'_i} \rangle$ *denoting a subsequence of* $ts(g_i)$. $ts(g'_i) \subseteq ts(g_i), m' \leq m$, *iff* $\forall op^{k'}_{g'_i} \in ts(g'_i), 1 \leq k' \leq m'-1, \exists op^k_{g_i} \in ts(g_i), 1 \leq k \leq m-1; op^{k'}_{g'_i} = op^k_{g_i}$ *holds.*

We can use time series subsequences to locate sub-time-variant graphs which are potential graph-shapelets. To predict a new testing time-variant graph, we can also convert it to a transformation sequence to compare the distances of time-variant graphs by calculating the distance of their corresponding transformation sequences.

In Fig. 5.2, for a time-variant graph g_i, when its individual graph $g_i^{(1)}$ changes to $g_i^{(2)}$, the corresponding transformation sequence is $op^1_{g_i} = \langle e -^{(1,1)}_{(1,2)} v -^{(1,2)}_{1} \rangle$, where superscripts $(1,1)$ and $(1,2)$ represent the first and second graph operations changing from graph $g_i^{(1)}$ to $g_i^{(2)}$. $e-$ and $v-$ represent the removal of an edge and a node. The order of the operations subscripts $(1,2)$ and 1 represent that we remove an edge from the node of ID 1 to ID 2, and remove a node of ID 1. Similarly, $op^2_{g_i} = \langle v +^{(2,1)}_{5} e +^{(2,2)}_{(2,5)} e +^{(2,3)}_{(3,5)} \rangle$.

Example 1: A sub-time-variant graph $\langle g_i^{(1)} g_i^{(2)} g_i^{(3)} \rangle$ generates a graph transformation sequence $\langle e -^{(1,1)}_{(1,2)} v -^{(1,2)}_{1} v +^{(2,1)}_{5} e +^{(2,2)}_{(2,5)} e +^{(2,3)}_{(3,5)} \rangle$ as shown in Fig. 5.2, so we can also generate its transformation subsequence $\langle v -^{(1,2)}_{1} e +^{(2,2)}_{(2,5)} e +^{(2,3)}_{(3,5)} \rangle$ with $e +^{(2,3)}_{(3,5)}$ denoting the edge adding operation between nodes 3 and 5.

Definition 8. *(*Edit Similarity*) For two graph transformation sequences* $ts(g_i)$ *and* $ts(g_j)$*, we use* $Sim(ts(g_i), ts(g_j))$ *to denote the Edit Similarity for measuring their similarity.* $ts(g_i) = \langle op^1_{g_i} \cdots op^{(m-1)}_{g_i} \rangle, 1 \leq k \leq m-1$, *with* $op^k_{g_i} = \langle op^{(k,1)}_{g_i} \cdots op^{(k,l_k)}_{g_i} \rangle$ *with* l_k *denoting the length of* $op^k_{g_i}$.

$$Sim(ts(g_i), ts(g_j)) = \sum_{k,v} \sum_{p=1}^{l_k} \sum_{q=1}^{l_v} \hbar^{(k,p),(v,q)}_{(i,j)}$$

where $\hbar^{(k,p),(v,q)}_{(i,j)} = I(op^{(k,p)}_{g_i} = op^{(v,q)}_{g_j})$, $I(\cdot)$ *is 1 if the condition inside is true; otherwise, 0.*

Example 2: In Fig. 5.2, the graph transformation sequence for the graph-shapelets of class A is $\langle v +^{(2,1)}_{5} e +^{(2,2)}_{(2,5)} e +^{(2,3)}_{(3,5)} \rangle$, denoted by $ts(g_i)$. The graph transformation sequence of the corresponding sub-time-variant graph in the test-

ing time-variant graph is $\langle v +_{5}^{(2,1)} e +_{(1,5)}^{(2,2)} e +_{(2,5)}^{(2,3)} e+_{(2,8)}^{(2,4)} \rangle$, denoted by $ts(g_t)$, $Sim(ts(g_i), ts(g_t)) = 2$ because of the same sequential operations.

When comparing two operations, we ignore their superscripts. For example, $e+_{(2,5)}^{(2,3)} = e+_{(2,5)}^{(2,2)}$. We use sliding windows to extract time series subsequences .

Definition 9. *(*Graph-Shapelet Pattern*) Given a time-variant graph dataset* $\mathcal{G}$ *which consists of two classes,* A *and* B*, graph-shapelet pattern is a special graph transformation subsequence with maximum edit similarity to the time-variant graphs in different class.*

When converting time-variant graphs to transformation sequences, we use the following strategies to represent the time-variant graphs and make the transformation sequences between two successive graphs unique.

Strategy 1: (The Admissibility). The transformation strategy between two successive graphs $g^{(j)}, g^{(i)}$ is as follows,

Table 5.2: Symbols and notations

Symbols	**Descriptions**
T	time series
ts	transformation sequence
op	transformation operation
l_k	the length of $op_{g_i}^k$
m, m'	the length of time-variant graph g_i and sub-time-variant graph g_i'
A,B	class label
$\mathcal{V}, E$	node and edge sets
node insertion $v+_u^{(j,k)}$	insert a node of ID u into $g_i^{(j)}$ which leads to $g_i^{(j+1)}$
node deletion $v-_u^{(j,k)}$	delete an isolated node of ID u in $g_i^{(j)}$ which leads to $g_i^{(j+1)}$
edge insertion $e+_{(u_1,u_2)}^{(j,k)}$	insert an edge between two nodes u_1 and u_2 into $g_i^{(j)}$ and obtain $g_i^{(j+1)}$
edge deletion $e-_u^{(j,k)}$	delete an edge between two nodes u_1 and u_2 in $g_i^{(j)}$ to obtain $g_i^{(j+1)}$

$*$: k is the index of operation in a transformation sequence.

$$v{+}_{u_1}^{(j,k)} \prec e{+}_{(u_1,u_2)}^{(j,k')}, e{+}_{(u_1,u_2)}^{(j,k)} \prec e{-}_{(u_1,u_2)}^{(j,k')}, e{-}_{(u_1,u_2)}^{(j,k)} \prec v{-}_{u_1}^{(j,k')}$$

where $k < k'$, and the definitions of $v{+}_{u}^{(j,k)}, v{-}_{u}^{(j,k)}, e{+}_{(u_1,u_2)}^{(j,k)}, e{-}_{(u_1,u_2)}^{(j,k)}$ are illustrated in Table 5.1.

In terms of the same operation with different IDs, we use a lexicographic order to generate a unique sequence, *e.g.*, $v{+}_{1}^{(1,1)} \prec v{+}_{2}^{(1,2)}$.

Strategy 2: (The Representation). Any time-variant graph can be represented by the four operations in Table 5.1 with a given initial interstate $g_i^{(1)}$ [44].

Table 5.2 summarizes major notations used in the chapter.

5.3 Overall Framework of Graph-shapelet based TVG Learning

Fig. 5.3 shows the framework of our method which first explore a univariate time series from each time-variant graph via a simple kernel method (Section 5.4.1). We then calculate shapelets from the time series using shapelets mining (Section 5.4.2). After that, sub-time-variant graphs that match the shapelets from the original time-variant graphs are located. Each sub-time-variant graph corresponds to a unique graph transformation subsequence by applying the Admissibility strategy and the Representation strategy. In the last step, we calculate the graph edit similarity between graph transformation subsequences and find the most discriminative transformation subsequences as graph-shapelets (Section 5.4.3).

Although many methods exist to find shapelet for time series classification, it is very challenging to restore sub-time-variant graph from the time series shapelets. To address the challenge, we record the locations of time series segments that have the minimum Euclidean distance, while introducing a transformation of time-variant graphs to time series.

Algorithm 5 SubTVG: Generate sub-time-variant graph candidates

Require:
$\mathcal{G}$: A set of time-variant graphs;
l: Length of graph-shapelet patterns;
Ensure:
$\mathcal{G}'$: Graph subsequence candidates;
1: $D \leftarrow$ Apply $\mathcal{G}$ to obtain corresponding time series D;
2: $cSet \leftarrow \emptyset$; // Time series candidates with length l
//Generate time series candidate with length l
3: **for** t **in** D **do**
4: $S_t^l \leftarrow$ Extract time series subsequence with length l from t
5: $cSet \leftarrow cSet \bigcup \mathcal{S}_t^l$;
6: **end for**
7: Shapelets $\leftarrow$ Apply $cSet$ to D based on shapelet mining[147].
8: Record the Shapelets locations in each time series
9: **for each** g_i **in** $\mathcal{G}$ **do**
10: $g_i' \leftarrow$ Apply recoded Shaplelet location to the related sub-time-variant graphs in g_i;
11: $\mathcal{G}' = \mathcal{G}' \bigcup g_i'$
12: **end for**
13: **return** $\mathcal{G}'$

5.4 Graph-shapelet Feature Exploration

5.4.1 Graph Sequences to Time Series

We first design a kernel method to explore time-series information $t_i = \{t_i^{(1)}, \cdots, t_i^{(m)}\}$ from a time-variant graph g_i, $K_{t_i}^{(j)} = N_n(g_i^{(j)}) + N_e(g_i^{(j)})$, where $N_n(g_i^{(j)})$ and $N_e(g_i^{(j)})$ denotes the statistical information of the nodes and edges in an individual graph $g_i^{(j)}$ at time j with $1 \leq j \leq m$. After that, efficient shapelet pattern mining algorithms can be applied to quickly locate graph changes in time-variant graphs. The reason why we use the graph statistic method is that we can find the location of graph-shapelets and return to graphs after discovering graph-shapelets. By using this method, we can speed up the search time and save the graph information since we only use a segment of a time-variant graph.

Example 3: In Fig. 5.2, the number of nodes and edges in graphs $g_i^{(1)}, g_i^{(2)}, \cdots, g_i^{(m)}$ are $7, 5, \cdots, 6$ respectively. Thus, we can build a time series $\{7, 5, \cdots, 6\}$, as

shown in Fig. 5.2.

5.4.2 Graph-Shapelet Pattern Candidates

All sub-time-variant graphs are potential graph-shapelet candidates. The maximum similarity of a candidate to all training sequences can be used for graph prediction by ranking the similarity of a candidate *w.r.t.* a testing time-variant graph. Therefore, in order to find graph-shapelet patterns, the algorithm needs to generate potential sub-time-variant graphs as candidates and then find the one, which has maximum similarity *w.r.t.* other subsequences from the candidate sub-time-variant graphs, as graph-shapelet patterns.

Algorithm 5 shows the detailed procedures for finding sub-time-variant graph candidates from time-variant graphs. Given a time-variant graph database $\mathcal{G}$ along with user-defined graph-shapelet length l. Line 1 converts time-variant graphs $\mathcal{G}$ to time series D based on the number of nodes and edges in each graph. After that, the algorithm generates all candidates within a sliding window (lines 3-6). For each shapelet candidate, the algorithm calculates the Euclidean distance between the candidate and other time series and records the locations with minimum distance in each time series (lines 7-8). Meanwhile, the algorithm updates the distance threshold and sub-time-variant graph locations, if any candidate has a smaller distance than the current distance threshold. Finally, the algorithm returns the sub-time-variant graphs and labels in each time-variant graph based on the sub-time-variant graph locations (lines 9-12).

5.4.3 Finding Graph-Shapelet Patterns

Graph-shapelet patterns can be found from the generated sub-time-variant graph candidates. To calculate graph-shapelet patterns from sub-time-variant graphs, time-variant graphs should be converted to transformation sequences so that we can use edit similarity as the measure. The transformation sequences between two graphs in a given time-variant graph can be represented by a series of operations, *e.g.*, node and edge insertions and deletions based on *Strategy 1* and *Strategy 2*. According to the time-variant graph representation, an algorithm to compile $\mathcal{G} =$

Algorithm 6 GraphSequenceCompiler

Require:
$\mathcal{G}'$: time-variant graph candidates;
Ensure:
ts: graph transformation sequences;
1: $ts \leftarrow \emptyset$;
2: **for each** $g_i \in \mathcal{G}'$ **do**
3: **for each** $g_i^{(j)} \in g_i$ **do**
4: Represent $g_i^{(j)}$ to $\mathcal{V}_{v+}, \mathcal{V}_{v-}, E_{e+}, E_{e-}$ follow *Strategies 1, 2*;
5: $k \leftarrow 1$
6: **for each** v in $\mathcal{V}_{v+}$ **do**
7: $ts \leftarrow ts \bigcup v+_{id(v)}^{(j,k++)}$; // Get the insertion nodes set
8: **end for**
9: **for each** (v, v') in E_{e+} **do**
10: $ts \leftarrow ts \bigcup e+_{(id(v),id(v'))}^{(j,k++)}$; // Get the insertion edges set
11: **end for**
12: **for each** (v, v') in E_{e-} **do**
13: $ts \leftarrow ts \bigcup e-_{(id(v),id(v'))}^{(j,k++)}$; // Get the deletion edges set
14: **end for**
15: **for each** v in $\mathcal{V}_{v-}$ **do**
16: $ts \leftarrow ts \bigcup v-_{id(v)}^{(j,k++)}$; // Get the deletion nodes set
17: **end for**
18: **end for**
19: **end for**
20: **return** ts

$\{g_i | g_i = \langle g_i^{(1)} \cdots g_i^{(m)} \}$ to a set of transformation sequences $ts(\mathcal{G}) = \{ts(g_i) | g_i \in \mathcal{G}\}$ is summarized in Algorithm 6.

The procedure basically follows *Strategies 1, 2* to convert graph sequences (line 4). Then, the node and edge insertions are appended to the transformation sequences, as shown on lines 6-8 and 9-11 respectively. After that, node and edge deletions are carried out on lines 12-14 and 15-17 respectively, with the changes appended to the transformation sequences.

Example 4: In the graph sequence $\langle g_i^{(2)} g_i^{(3)} \rangle$ given in Fig. 5.2, the insertion of node ID 5 and edge insertions $(2, 5)$ and $(3, 5)$ from $g_i^{(2)}$ to $g_i^{(3)}$, result in the transformation sequences, *i.e.*, $v+_5^{(2,1)}, e+_{(2,5)}^{(2,2)}, e+_{(3,5)}^{(2,3)}$.

Algorithm 7 Graph-shapelet Pattern Classifier

Require:
$\mathcal{G}$: a set of n time-variant graphs;
l: Length of the graph-shapelets;
Ensure:
L_t: The label of a test time-variant graph g_t;
// **Training Phase:**
1: $graph\_shapelt \leftarrow \emptyset$;
2: $candidates \leftarrow$ SubTVG($\mathcal{G}, l$); // Algorithm 5
3: $max\_sim \leftarrow 0$;
4: $ts \leftarrow$ GraphSequenceCompiler($candidates$); // Algorithm 6
5: **for** each $ts_i \in ts$ **do**
6: $d_i \leftarrow \sum_j Sim(ts_i, ts_j), ts_j \in ts(i \neq j)$;
7: **if** $d_i > max\_sim$ **then**
8: $max\_sim \leftarrow d_i$;
9: $graph\_shapelets \leftarrow ts_i$;
10: **end if**
11: **end for**
// **Test Phase:**
12: $ts(g_t) \leftarrow$ GraphSequenceCompiler(g_t); // Algorithm 6
13: **for** each $graph\_shapelet$ in $graph\_shapelets$ **do**
14: calculate $Sim(graph\_shapelet, g_t)$;
15: **end for**
16: L_t is the label with maximum similarity;
17: **return** L_t

5.5 Graph-shapelet based TVG Classification Algorithm

5.5.1 Classification with Graph-Shapelet Patterns

Fig. 5.3 demonstrates an example of finding two graph-shapelet patterns for a binary classification problem, and the algorithm details are summarized in Algorithm 7. First, the algorithm calls SubTVG() to generate all graph-shapelet pattern candidates (line 2) and calls function GraphSequenceCompiler() to generate graph transformation sequence for each generated sub-time-variant graph candidates (line 4). Then, it calculates distance between each generated sub-time-

variant graph and all time-variant graphs in each class (line 6). The sub-time-variant graph with maximum similarity are selected as graph-shapelet pattern, and its transformation sequences are used as classification features (lines 7-10).

Whenever a test time-variant graph arrives, the algorithm converts it to transformation sequences and computes the distance between test transformation sequences and features selected for different classes, by using Edit Similarity (Definition 5). The output label of the test graph is the same as the label of the graph-shapelet pattern which has the maximum similarity with respect to the test time-variant graphs (lines 12-17).

5.5.2 Time Complexity Analysis

The proposed graph-shapelet pattern based time-variant graph classification algorithm includes (1) converting time-variant graphs to graph transformation sequences, and (2) finding frequent transformation subsequences as graph-shapelet patterns for classification. The entire graph classification time complexity is $O(mrl * n)$, while that of graph-shapelet pattern mining is $O(ml * n)$, where m is the length of the query time-variant graphs, l is the length of the graph-shapelet patterns, r is the average length of transformation sequences between two graphs, and n is the size of time-variant graph database.

5.6 Experiments

5.6.1 Data sets

5.6.1.1 Synthetic Time-Variant Graph Data

In our experiments, we first create synthetic information propagation graph data, so we can validate algorithm performance on data with known ground-truth properties. We use a well-known model, namely Small World [131], to generate synthetic message propagation graphs for testing and comparisons. Small world is defined as a graph in which the typical distance ζ between two randomly chosen nodes (the number of steps required) grows proportionally to the logarithm of the number of nodes N in the graph, that is $\zeta \propto \log N$. We use parameter α to denote

that each node is connected to α nearest neighbors in topology, and p denotes the rewiring probability. To better simulate real-world information diffusion, we generate synthetic graphs under a power-law degree distribution exponent $\alpha = 1.5$, which corresponds to the power-law degree exponent of the weibo graph. Synthetic information propagation graphs (*i.e.,* time-variant graphs) are generated using the following methods on the simulated graphs. First, we randomly select a root node r with a non-zero out-degree. The node r is then added to the initially empty list of the infected nodes I and all outgoing edges (r, s) are added to the initially empty FIFO queue of the infected-susceptible node pairs S. We choose an edge from the candidate set each time and calculate the time delay for the edge until the time delay exceeds a given time window (we use [0,1000] as the time window). We separate each information propagation with time delay 100, 200, 300, 400, 500, 600 and 700, which means the average length is 7. The data are generated by repeating the above steps to generate 200 graph sequences. If the number of nodes in a graph sequence is more than 100, we label it as outbreak class; otherwise, non-outbreak.

5.6.1.2 Real-World Time-Variant Graph Data

We evaluate the performance of the proposed method on two real-world data sets, including phone call graph obtained from MIT [32] and the information propagation graph data crawled from Sina Weibo.

The phone call time-variant graph contains dynamics of 75 students/faculty members in the MIT Media Laboratory, and 25 incoming students at the MIT Sloan business school adjacent to the Media Laboratory. The experiment was designed to study community dynamics (classified as personal behavior or interpersonal interactions), by tracking a sufficient amount of people with their personal mobile phones. In our experiments, a unique ID is assigned to each person participating the communication and an edge links two persons if they communicate via phone call in a day. By doing so, we can obtain a daily graph $g^{(j)}$. According to the daily node degree (threshold is set to 5), each person can be either hub person or non-hub person. The person having a high degree is considered to be a hub person in the community. We obtain a set of weekly time-

variant graph data, *i.e.*, $\mathcal{G}$, and the total numbers of weeks (number of sequences) are 40. We randomly sampled $|V| = (1 \sim 20)$ persons, and assign label equals 1 if the weekly phone call contains a hub person (*i.e.,* hub person usually triggers interpersonal interactions), otherwise 0. The average length of each sequence is 7.

The weibo time-variant graph data set is obtained from a research community in Weibo. First, we crawl 200 weibo propagations from a research community within one month. We sample 50 users and assign each of them a node ID (varying from 1 to 50). The propagation of the information from users follows a temporal order. At each time point, all nodes reached by the weibo form a graph. For a new weibo, its forwarding re-forwarding propagation graphs collected in a time period form a time-variant graph (*i.e.* a time-variant graph), and our learning goal is to use time-variant graphs to predict whether the time-variant graph for a new weibo will outbreak or not. In our experiments, we obtain 200 time-variant graphs from propagation of 200 weibo at different time periods, and the average length of each sequence is 15. We define a time-variant graph as an outbreak, if the total number of nodes is more than 100.

5.6.2 Experimental Settings

5.6.2.1 Baseline Approaches

Because no existing approaches are available for time-variant graph classification, we compare the proposed graph-shapelet patterns based algorithm (gShapelet for short) with the following benchmark methods in terms of prediction accuracy and running time to evaluate the classification performance.

- **Frequent subsequences based method**. This type of methods convert all training time-variant graphs to graph transformation sequences and discover frequent subsequences as features for classification. This method resembles to the work in [44], and the difference is that [44] does not consider classification problems.

- **Frequent subgraph based method**. Because we cannot obtain stable subgraphs from time-variant graphs, for frequent subgraphs based method,

we mine frequent subgraphs, *e.g.*, gSpan [142], from the last graph in a time-variant graph as features.

- **gShapelet_n**. When transforming a time-variant graph into a univariate time series, gShapelet_n counts the number of nodes as statistical magnitude.

- **gShapelet_e**. In order to transform a time-variant graph into a univariate time series, gShapelet_e counts the number of edges as statistical magnitude.

5.6.2.2 Evaluation Measures

We randomly select 60% time-variant graphs for training, with remaining time-variant graphs being used for testing. When using frequent subsequences or subgraphs methods, we need to set the support threshold σ. Given a set of data $\mathcal{G} = \{g_i | g_i = \langle g_i^{(1)} \cdots g_i^{(m_i)} \rangle\}$, a support value σ of a transformation subsequence and subgraph in graph sequence g' are defined as

$$\sigma = \frac{|\{g_i | g_i \in \mathcal{G}, ts(g') \sqsubseteq ts(g_i)\}|}{|\mathcal{G}|} \; for \; subsequences,$$

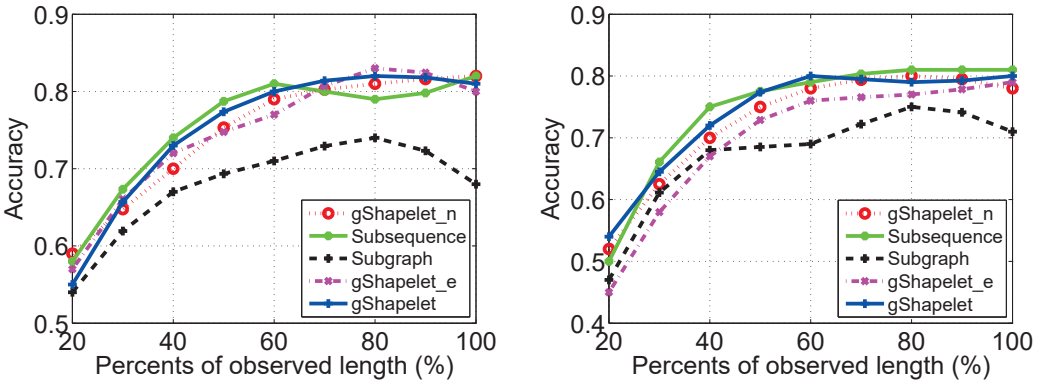
$$\sigma = \frac{|\{g_i | g_i \in \mathcal{G}, g' \sqsubseteq g_i\}|}{|\mathcal{G}|} \; for \; subgraphs.$$

5.6.3 Experimental Results

The four algorithms are compared under different observation length of the time-variant graphs, *i.e.*, 20%, 40%, 60%, 80% and 100% in each time-variant graphs. In Fig. 5.4 and Fig. 5.5, we report the algorithm performance in terms of classification accuracy and running times, respectively.

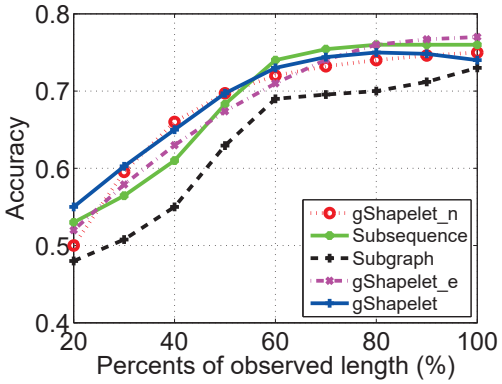
5.6.3.1 Effectiveness Results

The results in Fig. 5.4 show that the classification accuracy increases with respect to the observation times, and the results tend to be stable as the time elapsing. This is mainly because more nodes are observed which leads to more accurate shapelet or subsequence or subgraph patterns. Once the percent of participant



(a) Synthetic time-variant graph

(b) Phone call time-variant graph

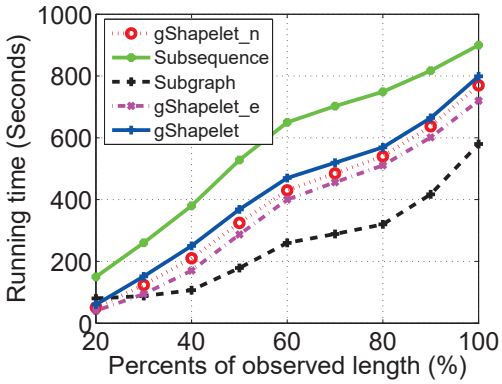


(c) Weibo time-variant graph

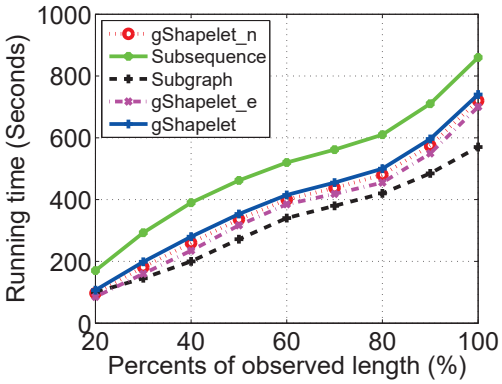
Figure 5.4: Accuracy comparisons with respect to different time stages on both synthetic and real-world time-variant graph data sets.

length reaches 60%, the algorithms likely find the best graph-shapelet patterns, so the accuracy tends to become stable. These results imply that graph-shapelet patterns can be used for early prediction.

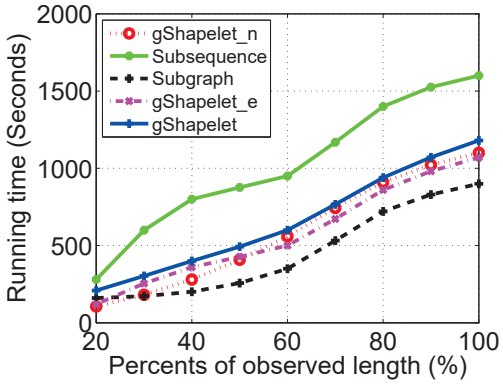
As we can observe from Figure 5.4, the graph-shapelet based approaches have comparable accuracy with subsequence-based method while perform much better than subgraph-based method. This is because frequent subsequences based method mine features from the entire time-variant graphs. This kind of approach can ensure the classification accuracy but requires more running time. While the graph-shapelet based methods first find discriminative segments from time-variant graphs, and then find features from the segments instead of the whole time-variant graphs. This way is a significant reduction in running time. Con-



(a) Synthetic time-variant graph



(b) Phone call time-variant graph



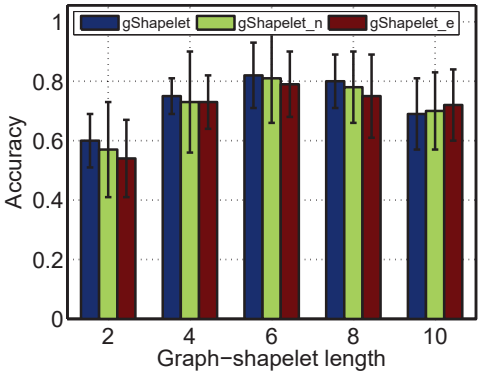
(c) Weibo time-variant graph

Figure 5.5: The average CPU time with respect to different time stages on both synthetic and real-world time-variant graph data sets.

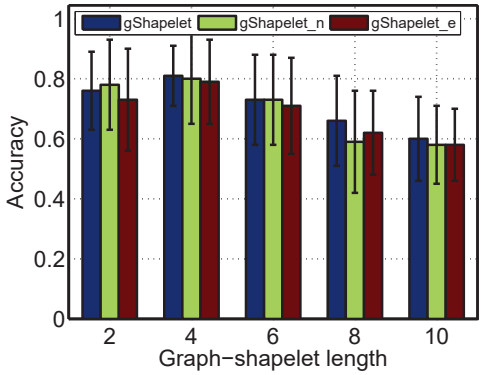
sequently, the same as on-line and off-line methods, we have to trade off the accuracy with running time performance, especially on large scale data sets. Our aim is to make the proposed methods be more efficiency under comparable accuracy. The reason why the proposed graph-shapelet methods outperform subgraph based method is that subgraph based method does not consider the temporal correlations during subgraph mining, and the distance between subgraph patterns may change irregularly over time.

5.6.3.2 Efficiency Results

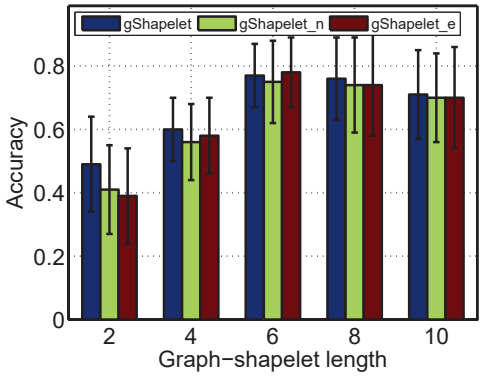
Fig. 5.5 shows the running time of the methods, where subgraph based method has the lowest running time while subsequence based method is the most time-



(a) Synthetic time-variant graph data



(b) Phone call time-variant graph data



(c) Weibo time-variant graph data

Figure 5.6: Comparisons with varying graph-shapelet length on both synthetic and real-world time-variant graph data sets.

consuming. The reason is that subsequence base method needs search the entire time-variant graphs to find discriminative substructures whereas subgraph based method only mine frequent subgraphs from single graph. There is no significantly difference among gShapelet, gShapelet_n and gShapelet_e, because the only difference of these three methods is when transforming a time-variant graph into a univariate time series.

The classification accuracy and running time results show that the graph-shapelet based methods have higher accuracy than subgraph based method, and lower running time than subsequence based method. This is because graph-shapelet pattern based methods consider transformation relationships between changing graphs, and graph-shapelet patterns can learn discriminative structures

of data belonging to different classes. In addition, graph-shapelet patterns are usually much shorter than the original time-variant graph, which avoid comparisons on the entire data set and result in more efficient running time performance.

5.6.3.3 Analysis of gShapelet Algorithm

Fig. 5.6 shows the accuracy with respect to graph-shapelet pattern length when using nodes and edges as the graph statistics (gShapelet), nodes (gShapelet_n), and edges (gShapelet_e) respectively. From Fig. 5.6, we can observe the highest classification accuracy is achieved when *len* is 6, 4, 6 for the Synthetic, MIT, and Weibo time-variant graph data, respectively.

Fig. 5.7 shows a simple yet interesting example extracted from the MIT phone call time-variant graph data. In the first sequence, the "normal" persons 1 and 2 connect with a hub person 2, which leads to an increase number of participants. On the other hand, communications generated by "normal" persons have a relatively stable number of users. This implies that the first sequence is likely to contain interesting information, and the potential graph-shapelet patterns can be explored from the corresponding graph transformation sequence.

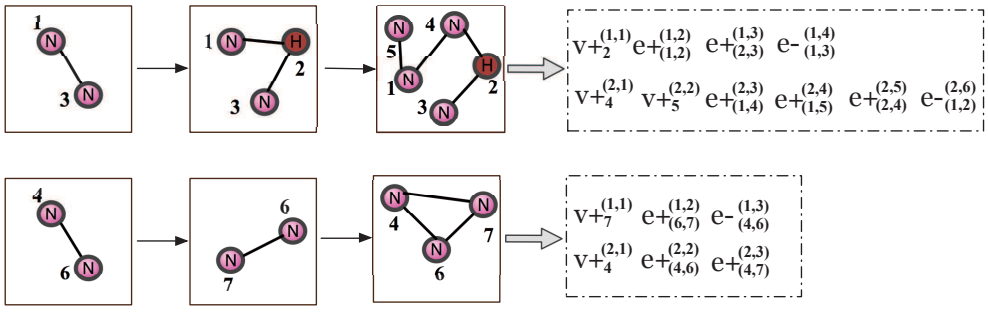


Figure 5.7: Two graph transformation sequences extracted from the MIT phone call time-variant graph data. The symbol "N" represents normal person and "H" represents hub person. The first graph sequence shows that a weekly phone call time-variant graph data contain a hub person, while the second graph sequence shows that all the participants are normal persons.

Figure 5.8: An example of message propagated in a Sina weibo time-variant graph. At different propagation stage, the diffusion (including reached nodes and propagation edges) constitutes a graph. The propagation of the message in temporal order will form a set of temporally related graphs. Each weibo propagation is regraded as a time-variant graph.

5.7 Discussions

Intuitively, since the outbreak of an information propagation graph is justified by the number of nodes influenced by the propagation, alert readers may wonder why do we transfer time variant-graphs as transformation sequence and further identify graph-shapelet patterns for classification, or why don't we directly use the change of number of nodes as features for classification? In short, our hypothesis is that outbreak in information propagation is driven by some special graph structure, instead of being simply determined by nodes with a large number of degrees. Indeed, an information propagation may not be outbreak even if it reaches some nodes with very high degrees (*i.e.* popular users). This is consistent with existing researches [67, 153, 83], which observed that information propagation is not mainly spread by nodes with a large number of degrees by rather spread by nodes with a medium number of degrees.

To further validate our hypothesis, we use a real-world case study from weibo time-variant graph data to demonstrate the discovered graph-shapelet patterns. Fig. 5.8 is an example of outbreak information propagation, which was crawled every half an hour. The weibo time-variant graph information propagation is about a news message propagated in a University community, regarding an interesting documentary of the institute. During different propagation stages, the propagation of the news forms variant graphs. Therefore, one message diffusion in different propagation stage constitutes one time-variant graph, and the diffusion of messages in the community forms time-variant graph dataset. The message diffusion includes outbreak or non-outbreak time-variant graph, and our aim is to classify each information propagation graph record into correct classes (outbreak vs. non-outbreak). A naive method to predict time-variant graph outbreak is to count the node volumes, however, the time-variant graph is not only influenced by the nodes but by structures connected the nodes, such as who is sending the message, who resends the message, when the message was sent or resent etc., which is exactly captured by time-variant graphs. The general graph classification is to use each graph at a certain period separately to discover features. In reality, a diffusion graph will influence following graphs. For time-variant graph classification, we consider variation between different graphs. Because a graph-

shapelet pattern denotes the transformation of nodes and edges between different graphs, it can inherently capture the nodes information, structure information, and temporal variation graphs, for time-variant graph classification.

Fig. 5.8 shows an example of the mined graph-shapelet patterns. This graph-shapelet pattern example represents the outbreak class, which demonstrate that a time-variant graph tends to be outbreak if there is a sequence of special node and edge insertions, such as node PKU and edge (PKU_news, PKU) which represents Peiking Universality.

5.8 Conclusion

In this chapter, we formulated a new time-variant graph classification task and proposed to use graph-shapelet patterns as features to represent time-variant graphs for learning and classification. We argued that existing graph classification methods are all based on static settings, where training graphs are assumed to be independent of each other and the structure of each graph remains unchanged. In reality, many applications involve data with dynamic changing structures. Accordingly, a time-variant graph can be used to represent a sequence of graphs and capture dynamic evolution of the underlying object. The inherent structural dependency and temporal correlations of the time-variant graphs raise significant challenges, and we advocated a graph-shapelet pattern concept to detect significant changes in the time-variant graph as features. By using graph-shapelet patterns to convert a time-variant graph as a tokenized sequence, we can effectively calculate the distance between two time-variant graphs for classification. Experiments on synthetic and real-world data demonstrated that our method is much more accurate than traditional subgraph feature based approachers.

The key innovation of the chapter, compared to existing work in the area is threefold: (1) a new time-variant graph learning task to model dynamic changes in structure data; (2) a unique graph-shapelet pattern as feature to capture structural and temporal dependency of the graphs; and (3) a fast time-variant graph classification model using structure features and edit similarity.

Chapter 6

Application of Online Diffusion Provenance Detection from TVG

6.1 Introduction

In recent years, information diffusion in large networks has attracted much attention. The spread of malicious information such as viruses, spams and rumors has made various networks vulnerable to privacy attacks, viral advertising, etc. To stop the propagation of malicious information, researchers recently proposed several models to identify the diffusion provenances in large networks. Online diffusion provenance discovery is also a significant presence on social networks in business. No matter how small, medium or large a business is, a brands health and reputation is often defined by the information diffused in social media. While fake reviews or deceptive messages, spread by malevolent people, in social media are inevitable, they may cause damage to brands or corporate reputation. A recent study (*i.e.,* Spam Trends in Today's Business World [106]) reported that the productivity cost of malicious information to European companies was an estimated US$2.8 billion, while US-based companies reported a loss of US$20 billion. According to The Washington Post [1], on Tuesday, April 24, 2013, a single hoax message sent via Twitter erased $200 billion from the US stock market in 2 minutes. In cases like these, locating and severing the provenances of the

[1] http://rt.com/business/tweet-hackers-wall-street-us-326/

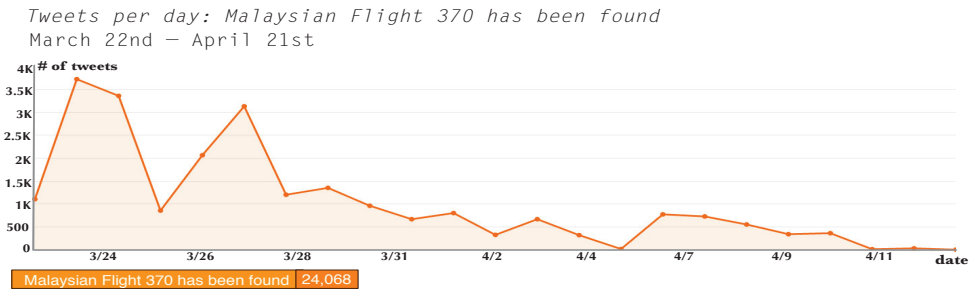


Figure 6.1: The rumor "*Malaysian Flight 370 has been found*" propagated on Twitter from March 22 to April 21, 2014. The x axis is the time and the y axis is the total number of tweets including the rumor.

diffusion in a timely manner is critical.

A false rumor (*i.e.,* libelous statement) about the financial performance of a firm may be spread by market manipulators to influence the price of the firm stock, resulting in fines from regulators or data protection enforcement agencies. While legal recourse exists for victims of libel, law enforcement agencies still need to identify the original source of the rumor and those who spread it. If the perpetrators are anonymous, tracing the IP address and identities of the individual profiles carrying or linking to the opinion piece becomes significantly more difficult and time-consuming.

Existing diffusion provenance identification models can be roughly categorized into two classes: the snapshot-based provenance identification [96, 75] and the detector-based identification [77, 93, 53]. The snapshot-based methods are under the assumption that a snapshot of the entire network can be obtained and the provenances can be estimated under stochastic propagation models such as the SI [109] and SIR [157] models. Although these methods have shown promising results in experiments, fetching a snapshot of the entire network is very expensive, if not impossible. The detector-based methods assume that only a small subset of nodes in a network can be monitored, and the provenances can be inferred from the observations (samples) from these detectors. This group of methods has recently attracted increasing attention due to its potential usage in real-world applications.

However, to our best knowledge, most existing work on the diffusion prove-

nance locating problem falls into the category of offline identification, where the data are assumed to be static and available all the time. In fact, for time-critical security monitoring applications, it is necessary to unveil the diffusion provenances as soon as an observation arrives. This way, it is important to detect diffusion provenances as early as possible to enable early warning, real-time awareness, and real-time response of malicious information spreading. For example, Fig. 6.1 shows the propagation of the rumor of *"Malaysian Flight 370 has been found"* on Twitter. The rumor was retweeted more than $3,500$ times in a single day (3/24). It can be seen that the rumor can be propagated to a large population in a very short time. Hence, it is imperative to detect a rumor provenance promptly.

Motivated by the urgent demand of real-time and continuous diffusion provenance detection in networks, we propose to use regression learning as the basic detection model. The regression learning is favorable for real-time applications due to the freely available prior distribution. Fig. 6.2 shows a network propagation with only one provenance S_0. We extract a propagation path with seven nodes $\{S_0, \cdots, S_6\}$. Assume that we have the privilege to observe nodes S_2 and S_4 (detectors). The two nodes are activated at time points $t^* + \tau$ and $t^* + 2\tau$ respectively. The goal is *to use the least square regression to minimize the error rate between the observed time delay and the estimated time delay.* Assume at time $t_i \in T = [t^*, t^* + 3\tau]$, the i^{th} node is observed. At any time t_i, we have, $\min_x \sum_{i=2,4} [(t^* + i * \tau) - a_i^T x]^2, \quad s.t. : \ x \geq 0$, where $x \in \mathbb{R}^7$ is the target variable with element x_i denoting the probability that node i is the diffusion provenance, $a_i \in \mathbb{R}^7$ is a column vector with element a_{ij} denoting the propagation path length from the detector i to the j^{th} node, e.g., $a_2 = (1, 2, 0, 1, 3, 4, 2)^T$ and $a_4 = (2, 1, 3, 4, 0, 1, 5)^T$. The objective function is convex and non-negative and achieves its minimum value 0 when $x = (1, 0, 0, 0, 0, 0, 0)^T$. For offline detection, since data from both nodes S_2 and S_4 are known, we can obtain the result as $x = (1, 0, 0, 0, 0, 0, 0)^T$, which correctly indicates that S_0 is the diffusion provenance. However, for online detection, we first estimate x by only observing data from the detector S_2 and the result is $(0, 0.8, 0, 0.2, 0, 0, 0)^T$. When data from S_4 arrive, the result of x is updated to $(1, 0, 0, 0, 0, 0, 0)^T$. We can see that an online algorithm demands dynamic and continuous computation of x, and the result of

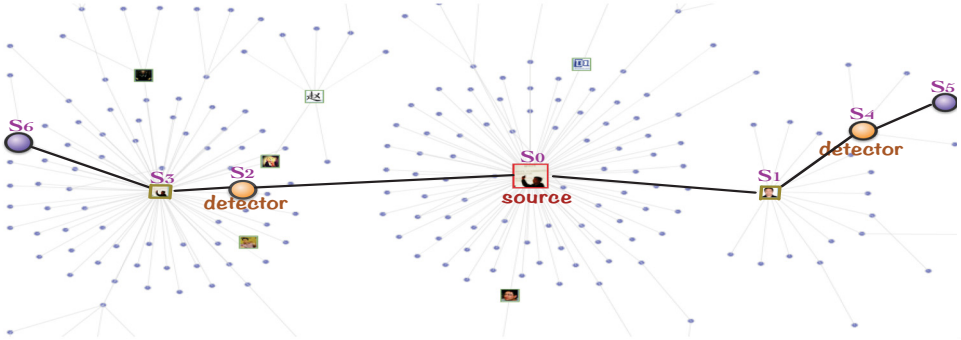


Figure 6.2: An example of twitter diffusion path. At the unknown time $t = t^*$, the information provenance S_0 initiates the diffusion of a tweet. The propagation time delay between any two nodes is τ and the time window $T = [t^*, t^* + 3\tau]$.

online learning is expected to approximate (or equal to) the offline result.

Compared to offline identification methods, online identification models have the following challenges:

- *Challenge 1: how to design an online identification model?* The online identification model needs to address five questions, the unknown number of diffusion provenance k, both activated and inactivated detectors, the unknown initial propagation time t^*, the uncertain propagation path, and the uncertain propagation time delay. The unknown number of diffusion provenance k expects a sparse solution of x. Both activated and inactivated detectors lead to partially labeled data and a non-convex objective function. The uncertain propagation path and propagation time delay demand an aggregate Gaussian distribution to describe the estimated propagation time delay.

- *Challenge 2: how to design a stochastic learning algorithm to solve the online identification model?* Because the detectors are activated sequentially while a decision needs to be made once a detector is activated, the algorithm needs to digest data in a continuous and converging way. Ideally, the results of the online algorithm approximate those of the offline algorithms.

- *Challenge 3: how to evaluate the performance of the proposed online identification method?* Given the unique characteristics of the problem, various

data are demanded to evaluate performance.

In this chapter, we propose a new online regression learning model to identify diffusion provenances in large networks [124, 128]. To solve *Challenge 1*, we use an l_1 non-convex regression learning model built on top of an aggregate Gaussian propagation time delay, where the network transmission weights are inferred a prior from offline collected cascades. To tackle *Challenge 2*, we present an Online Stochastic Sub-gradient algorithm (OSS for short) that can converge to local minima. Also, we evaluate the model using four synthetic network data to address *Challenge 3*.

The contributions of the work are twofold:

- We present a new online regression learning model to identify diffusion provenances in large networks. The proposed model can handle the issues of the unknown number of diffusion provenances k, the partially activated detectors, the unknown initial propagation time t^*, the uncertain propagation path, and the uncertain propagation time delay.
- We present an online stochastic algorithm to solve the proposed online regression learning model. The algorithm uses a stochastic sub-gradient decent algorithm to continuously detect the provenances.

The remainder of the chapter is organized as follows. Section 6.2 introduces the related preliminaries. The regression learning and online algorithm are given in Sections 6.3 and 6.4 respectively. Section 6.5 empirically evaluates the algorithms. We conclude the chapter in Section 6.6.

6.2 Preliminaries

Consider a network $G = \{V, E\}$, where the vertex set V has N nodes, and the edge set E has L edges. In the network, we have two types of data for model training:

1) *Offline data of cascades*: a set C of cascades $\{c_1, \cdots, c_n\}$, where each cascade c is a sequence of activated times $\{t_1, \cdots, t_N\}$ within a given time window

T. Each time point t_i records the i^{th} activated node. For nodes that are not activated within window T, the activated time is unknown.

2) *Online data collected from detectors.* We budget a small subset of detectors, denoted by $\mathcal{D} = \{d_i\}_{i=1}^{m}$, to monitor the network. During the monitoring time window $[t^*, t^* + T]$, where t^* is the initial propagation time and T is the size of the window, there are a subset of detectors activated, denoted by D_a, and the remaining inactivated detectors are denoted by D_u. We aim to estimate the locations of provenances, denoted by a random vector $s^* \in \mathrm{R}^N$, given the status of all the detectors $D = \{(d_1, t_1 - t^*), \cdots, (d_m, t_m - t^*)\}$, where t_m is the activated time point of detector d_m and label $t_m - t^*$ denotes the time delay. Note that time labels of inactivated nodes are unknown, and their time delay exceeds window T.

We adopt an SI propagation model to describe how the infection spreads in network G. The reasons for using this kind of propagation model are that most posts on social networks are usually not removed, i.e., an infected node stays infected. Thus, SI is an appropriate model for online postings and modeling opinion dynamics on social networks. In the SI model, each node in a network has three possible states: susceptible (s), infected (I) and non-susceptible (n). As the infected nodes are those nodes that possess the infection, and will remain infected throughout, an infected detector cannot be recovered. Therefore, a detector cannot receive the same information multiple times and will not send the same information multiple times to the same node.

Now we infer the **edge transmission probability matrix** W based on the cascades collected offline. Given a cascade $c_t \in C$, we use $f(c_t|W)$ to denote the likelihood function of observing the cascade c_t under an unknown transmission matrix W. The likelihood function consists of the joint probability of activated nodes $v_i \in V$ $(t_i \leq T)$ and inactivated nodes $v_m \in V$ $(t_m > T)$. For an activated node $v_i \in V$ activated at time $t_i \leq T$ from an adjacent node $v_j \in V$ and $t_j < t_i$, the probability of observing v_i activated by v_j at time t_i is a joint probability of v_j infecting v_i at time t_i and v_i is not activated by any other neighbors v_k which have been already activated by the time t_i, i.e., $t_k < t_i$. By summing up all the possible neighbors v_j in the network, i.e., summing up all v_j with $t_j < t_i$, we can

achieve the probability $f(c_{t_i}|W)$ of a node v_i activated at time t_i as in Eq. (6.1),

$$f(c_{t_i}|W) = \sum_{\substack{v_j \in g(v_i) \\ t_j < t_i}} \Big[P(t_j \to t_i|W) \prod_{\substack{v_k \in g(v_i) \\ v_k \neq v_j, t_k < t_i}} P(t_k \nrightarrow t_i|W) \Big]. \tag{6.1}$$

where $g(v_i)$ denotes the set of neighbors to v_i, and $P(t_j \to t_i|W)$ denotes the probability of the j^{th} activated node v_j activates its neighbor v_i at time t_i. We can use three well-known parametric functions for $P(t_j \to t_i|W)$, such as the widely used *exponential*, *power-law* and *Rayleigh* models [36]. Without loss of generality, we use the exponential model, where $P(t_j \to t_i|W)$ equals $w_{ji}e^{-w_{ji}(t_i - t_j)}$ if $t_j < t_i$ and 0 otherwise. The probability $P(t_j \nrightarrow t_i|W) = 1 - P(t_j \to t_i|W)$.

Based on Eq. (6.1), the probability of observing all the activated nodes in a cascade c_t is as follows,

$$f(c_{t \leq T}|W) = \prod_{v_i : t_i \leq T} f(c_{t_i}|W). \tag{6.2}$$

Because inactivated nodes also provide information for transmission weight estimation, the probability of observing an entire cascade c_t , $f(c_t|W)$, is a joint probability of observing all the activated nodes in the cascade c_t and unobserving the remaining inactivated nodes as in Eq. (6.3),

$$f(c_t|W) = f(c_{t \leq T}|W) \times \left(\prod_{v_m : t_m > T} \prod_{\substack{v_i \in g(v_m) \\ t_i \leq T}} P(t_i \nrightarrow t_m|W) \right). \tag{6.3}$$

The likelihood function of observing all cascades C is the product of all the likelihoods of each cascade given in Eq. (6.3),

$$L(W) = log \prod_{c_t \in C} f(c_t|W) = \sum_{c_t \in C} log f(c_t|W) \tag{6.4}$$

Then, the edge transmission matrix W can be estimated as follows,

$$W^* = argmax\ L(W),\ s.t., W \geq 0. \tag{6.5}$$

For simplicity, we denote $\phi(W; j, i) = P(t_j \to t_i|W)$, then Eq. (6.4) can be

rewritten as in Eq. (6.6),

$$L(W) = \sum_{c_t \in C} \Big(\sum_{\substack{v_i \\ t_i \le T}} log \sum\nolimits_{\substack{v_j \in g(v_i) \\ t_j < t_i}} \frac{\phi(W; j, i)}{1 - \phi(W; j, i)} + \sum\nolimits_{\substack{v_i \\ t_i \le T}} \sum\nolimits_{\substack{v_k \in g(v_i) \\ t_k < t_i}} log[1 - \phi(W; j, i)] + \sum\nolimits_{\substack{v_m: \\ t_m > T}} \sum\nolimits_{\substack{v_i \in g(v_m) \\ t_i \le T}} log[1 - \phi(W; i, m)] \Big) \tag{6.6}$$

The likelihood in Eq. (6.6) is convex, and thus the first-order gradient guarantees a global optimum. By letting the derivative of Eq. (6.6) to 0, we obtain $\phi'(W_{ji}) = 0$ for each cascade. That is, $W_{ji} = \frac{1}{t_i - t_j}$ for each observed pair of neighboring nodes. For each inactivated node $t_m > T$ in a cascade, as we don't observe the exact time of t_m, we approximately let $t_m = T$ and $W_{im} = \frac{1}{T - t_i}$. Moreover, we set the default propagation probability of each edge to be $W_{ji} = \frac{1}{T}$ in case that we don't observe any propagation data between nodes v_j and v_i in a cascade. Then, by averaging over $|C|$ cascades, we can obtain the final result as follows,

$$W^*_{ji} \propto \begin{cases} \frac{1}{\kappa} \sum_{c_t \in C} I_{c_t}(v_i, v_j) \frac{1}{t_i - t_j}, & t_i \le T, t_j < t_i; \\ \frac{1}{\kappa} \sum_{c_t \in C} I_{c_t}(v_i, v_j) \frac{1}{T - t_i}, & t_i > T, t_j \le T. \end{cases} \tag{6.7}$$

where $I_{c_t}(v_i, v_j)$ is an indicator and equals to 1 if the cascade c_t satisfies the time constraint. κ is the total number of node pairs that meet the constraint. If the given time constraints are not met, the default weigth W_{ji} is $1/T$.

6.3 Regression Model

In this part, we formulate the objective function for diffusion provenances detection based on *online data collected from the detectors*. We assume that the prior distribution of s^* is uniform over the network, i.e., any node in the network is equally likely to be the provenance. Thus, the location of the provenances can be recovered by maximizing the likelihood function of the observed status of the detectors D given provenances $s \in G$, as shown in Eq. (6.8),

$$s^* = \arg\max_{s \in G} P(D|s) = \arg\max_{s \in G} P(D_a|s) \Big[1 - P(D_u|s) \Big] \tag{6.8}$$

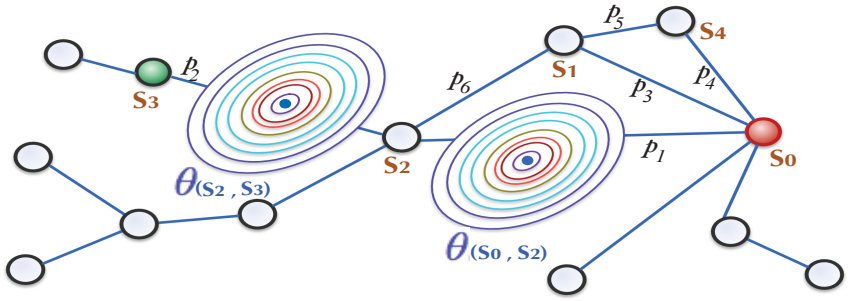


Figure 6.3: Gaussian time delay and the shortest-path propagation. The propagation path from the provenance s_0 to detector S_3 is approximated by the shortest path $\mathcal{P}(S_0, S_3) = \{S_0 \rightarrow S_2 \rightarrow S_3\}$. The propagation delay is an aggregate Gaussian distribution of paths p_1 and p_2.

where $P(D_a|s)$ denotes the probability of observing D_a given a set of spread provenances s, and $1 - P(D_u|s)$ denotes the probability that D_u is inactivated under provenances s. Thus, $P(D_a|s)\big[1 - P(D_u|s)\big]$ denotes the joint probability of both observing the activated nodes D_a and inactivated nodes D_u.

Because malicious information such as virus and rumors are often spread under *the snowball phenomenon* [85], it is reasonable to approximately use the shortest-path spread, denoted by $\mathcal{P}(i, j)$ between nodes $v_i, v_j \in V$.

Because the offline obtained edge transmission matrix W may suffer unstable change when used online, we further use a random variable $\theta_{i,j}$ to describe delay time along edge $e_i \in E$. The random variables are independent identically distributed with Gaussian distribution $\theta_{i,j} \sim N(\mu_{i,j}\sigma_{i,j}^2)$, where the mean $\mu_{i,j}$ are from matrix W.

Figure 6.3 shows an example of the Gaussian time delay and the shortest-path propagation. In the worst case, the search for the shortest path takes time $O(N^2)$.

Thus, the probability $P(D_a|s)$ in Eq. (6.8) can be rewritten as follows,

$$P(D_a|s) = \prod_{i=1}^{m_a} \sum_{\prod_s^a} P(\prod_s^a|s) P(d_a|\prod_s^a) = \prod_{i=1}^{m_a} P(\theta_{s,d_a}), \tag{6.9}$$

where m_a is the number of observers during the time window, and θ_{s,d_a} is the time delay. Based on the Gaussian distribution $\theta_{i,j} \sim N(\mu_{i,j}\sigma_{i,j}^2)$, the mean

and variance of the random variable θ_{s,d_a} are $\mu_a^T x$ and $(\sigma_a^2)^T x$ respectively. Let $\Lambda = (\sigma_a^2)^T x$, the Eq. (6.9) can be converted to Eq. (6.10),

$$P(D_a|s) = \prod_{i=1}^{m_a} (2\pi\Lambda)^{-1/2} e^{-(t_a - t^* - \mu_a^T x)^2/(2\Lambda)}. \tag{6.10}$$

Similarly, for unobservers, let $\Lambda' = (\sigma_u^2)^T x$, we have

$$P(D_u|s) = \prod_{i=1}^{m_u} (2\pi\Lambda')^{-1/2} e^{-(t_u - t^* - \mu_u^T x)^2/(2\Lambda')}. \tag{6.11}$$

Thus, Eq. (6.8) can be rewritten as

$$P(D|s) = \prod_{i=1}^{m_a} (2\pi\Lambda)^{-1/2} e^{-(t_a - t^* - \mu_a^T x)^2/(2\Lambda)} \cdot \prod_{i=1}^{m_u} [1 - (2\pi\Lambda')^{-1/2} e^{-(t_u - t^* - \mu_u^T x)^2/(2\Lambda')}]. \tag{6.12}$$

Let all time delays share the same variance σ^2, then the log-likelihood function in Eq. (6.12) is

$$lnP(D|s) = -\frac{1}{2\Lambda} \sum_{i=1}^{m_a} (t_a - t^* - \mu_a^T x)^2 + \frac{1}{2\Lambda} \sum_{i=1}^{m_u} (t_u - t^* - \mu_u^T x)^2 + C, \tag{6.13}$$

where C is a constant, $t_{a(u)} - t^*$ is the observed (unobserved) time delay.

Then, maximizing the log-likelihood[1] in Eq. (6.13) is tantamount to minimizing a quadratic regression function in Eq. (6.14), where the target vector $t \in \mathrm{R}^{m_a}$ is the observed time delay, i.e., $t_a - t^*$ for detector d_a, the coefficient matrix $A = (\mu_1^T, \cdots, \mu_{m_a}^T) \in \mathrm{R}^{N * m_a}$ is the shortest path time delay *w.r.t.* nodes $d_a \in D_a$. $\mathcal{T} \in \mathrm{R}^{m_u}$ is a vector of value T, which approximates the time delay of nodes D_u, i.e., $t_u - t^* \approx (T + t^*) - t^* = T$. Matrix $B = (\mu_1^T, \cdots, \mu_{m_u}^T) \in \mathrm{R}^{N * m_u}$ denotes the shortest time delay *w.r.t.* inactivated nodes $d_u \in D_u$. The parameter $\lambda > 0$ controls the weight. The constraints guarantee the optimal solution sparse and non-negative.

$$s^* = \arg\min_x \frac{1}{2} \|t - A^T x\|_2^2 - \lambda \|\mathcal{T} - B^T x\|_2^2 \quad s.t.: \ e^T x \leq \tau, \ \ x \geq 0. \tag{6.14}$$

[1] the log-likelihood is $\ln P(D|s) \propto -1/2 \sum_{i=1}^{m_a} (t_a - t^* - \mu_a^T x)^2 + \lambda \sum_{i=1}^{m_u} (t_u - t^* - \mu_u^T x)^2$, where $t_a - t^*$ is the observed time delay, and the same goes to inactivated nodes d_u. The second part is obtained because $ln(1 - e^x) \approx -e^x \approx -x$.

The above formulation relaxes traditional discrete optimization on graphs. Such a relaxation leads to efficient algorithms. Now we intuitively **explain** the above problem from the viewpoint of *multi-criteria quadratic programming*. The first objective function, $\|t - A^T x\|_2^2$, aims to fit time delay of activated detectors D_a, the second objective function $\|\mathfrak{T} - B^T x\|_2^2$ aims to fit time delay of inactivated detectors D_u. Because D_u are not actually activated during the time window T, a negative parameter $-\lambda < 0$ is used to avoid the fitting. The trade-off *Pareto solutions* can be obtained by varying parameter λ.

6.4 Online Algorithm

In this section, we present an Online Stochastic Sub-gradient algorithm to solve the regression model in Eq. (6.14). The proposed regression model, compared to the classical regression learning, has several new challenges: 1) the dependent variable t is implicit, because the initial propagation time is unknown, 2) the l_1 non-convex objective function expects sparse and fast convergent algorithm, and data collected from detectors arrive continuously. To solve the challenges, we use the *Relative Time Difference of Arrivals* and *Online Sub-gradient based on convex approximation* as the solutions.

6.4.1 Relative Time Difference

The dependent variable t in Eq. (6.14) is implicit, because the initial propagation time t^* is unknown. To solve this challenge, we can use an "anchor node" to cancel out the initial time t^*. Assume the α^{th} detector d_α is the "anchor node", its activated time is $t_\alpha = t^* + \sum_{e_i \in \mathcal{P}(s^*, d_\alpha)} \theta_i$, where θ_i is the time delay along edges $e_i \in \mathcal{P}(s^*, d_\alpha)$. Then, the *Relative Time Difference of Arrivals* (RTDA) between d_α and the k^{th} ($k \neq \alpha, 1 \leq k \leq m_a$) detector d_k is $\bar{t}_k := t_k - t_\alpha = \sum_{e_i \in \mathcal{P}(s^*, d_k)} \theta_i - \sum_{e_i \in \mathcal{P}(s^*, d_\alpha)} \theta_i$.

Based on RTDA, Eq. (6.14) has an elementary column change, i.e.,

$$\overline{A}(:, c_k) := A(:, c_k) - A(:, c_\alpha), \quad \overline{B}(:, c_k) := B(:, c_k) - B(:, c_\alpha). \tag{6.15}$$

Thus, the dimension of $\bar{t}$, $\overline{A}$ and $\overline{B}$ are $\bar{t} \in \mathbb{R}^{m_a - 1}, \overline{A} \in \mathbb{R}^{N * (m_a - 1)}$ and $\overline{B} \in$

$\mathbb{R}^{N*(m_u-1)}$ respectively. Then, Eq. (6.14) can be relaxed to Eq. (6.16),

$$s^*(x) = arg\min_x \frac{1}{2}\|\bar{t} - \overline{A}^T x\|_2^2 - \lambda\|\overline{B}^T x\|_2^2 + \rho\|x\|_1^1. \tag{6.16}$$

6.4.2 Convex Approximation

To address the non-convex challenge, we wish to find a sequence of convex programs, through which the non-convex function can be approximated and converge to a local minimum. To obtain a convergent sequence, at step $k+1$, the concave part $-\lambda\|\overline{B}^T x\|_2^2$ is linearly approximated by using the differential at the previous iterative point x_k, i.e.,

$$\frac{\partial}{\partial x}(-\lambda\|\overline{B}^T x\|_2^2)|_{x_k}, \quad x \geq -2\lambda\overline{B}^T\overline{B}\, x_k x. \tag{6.17}$$

At step $k+1$, we only solve a convex optimization as follows:

$$x_{k+1} \leftarrow \left\{ \min_{x\geq 0} \frac{1}{2}\|\bar{t} - \overline{A}^T x\|_2^2 - 2\lambda\overline{B}^T\overline{B}\, x_k x + \rho\|x\|_1^1 \right\}. \tag{6.18}$$

Lemma 4. The non-convex program in Eq. (6.16) converges under iterations $\{x_1, \cdots, x_k, \cdots\}$ generated by Eq. (6.18).

Proof. Denote Eq. (6.16) as $J(x)$, which is a combination of the convex part $J_{vex}(x) = \frac{1}{2}\|\bar{t} - \overline{A}^T x\|_2^2$ and the concave part $J_{cav}(x) = -\lambda\|\overline{B}^T x\|_2^2$. At step $k+1$, we have $J_{vex}(x_{k+1}) + J'_{cav}(x_k)x_{k+1} \leq J_{vex}(x_k) + J'_{cav}(x_k)x_k$. Moreover, due to the definition of concavity, $J_{cav}(x_{k+1}) \leq J_{cav}(x_k) + J'_{cav}(x_k)(x_{k+1} - x_k)$. Adding both sides of the above two inequalities, we obtain the result $J(x_{k+1}) \leq J(x_k)$. Therefore, Eq. (6.16) under the sequence $\{x_1, \cdots, x_k, \cdots\}$ generated by Eq. (6.18) is convergent. □

6.4.3 Online Sub-gradient

We use the *sub-gradient* method to solve the l_1 regularization problem in Eq. (6.16). Let $\mathcal{J}(\mathbf{x}) = F(\mathbf{x}) + G(\mathbf{x}) = \frac{1}{2}\|\bar{t} - \overline{A}^T x\|_2^2 - 2\lambda\overline{B}^T\overline{B}\, x_k x$, which is an approximation of the first two parts in Eq. (6.16) by using the CCCP programming

Function 1 Online Detecting During the Time Window

Require:

$\overline{A}$: Observation matrix;
$\bar{t}$: Time delay;
ϵ: Stop criteria;
η_t: Learning rate;

Ensure:

$\mathcal{X}^* = \{x_1, \cdots, x_j\}$: The indicator vector $\mathcal{X}^*$;

$x_{k+1} \leftarrow$ a best guess;
$\mathcal{L}(\mathbf{x}) = \boldsymbol{f}(\mathbf{x}) = \frac{1}{2}\|\bar{t} - \overline{A}^T x\|_2^2$
$\hat{s}^*(x) = arg\min_{x \geq 0} \mathcal{L}(x) + \rho\|x\|_1^1$
repeat
  $x_k \leftarrow x_{k+1}$;
  for $i = 1$ *to* m_a **do**
    $\bigtriangledown \mathcal{L}(x) = (\bar{t}_i - a_i x)^T(-a_i)$;
    $x_{k+1} \leftarrow \left\{x_k - \eta_t(\bigtriangledown \hat{s}^*(x))\right\}$;
  end for
until $\|x_{k+1} - x_k\| \leqslant \epsilon$
$\mathcal{X}^* \leftarrow x_{k+1}$ in a descending order;
$j^* = \underset{j}{\text{argmax}}\, |x_j^* - x_{j-1}^*|$;
return $\mathcal{X}^* = (x_1, \cdots, x_{j^*})$;

[150] at the concave part $-\lambda\|\overline{B}^T x\|_2^2$. The sub-gradient of $\mathcal{J}$ is as follows,

$$\begin{aligned}\bigtriangledown_j \mathcal{J}(x) &= F_j'(x) + G_j'(x) \\ &= \frac{1}{2}[\sum_{i=1}^{m_a}(\bar{t}_i - {a_i}^T x)^2]_j' + [-2\lambda \sum_{i=1}^{m_u}({b_i}^T b_i x_k x)]_j' \\ &= -(\overline{A}^T \bar{t} - \overline{A}^T \overline{A}\, x)_j - 2\lambda \overline{B}^T \overline{B} x_k.\end{aligned}$$

Now Eq. (6.16) turns to $s^*(x) = arg\min_x \mathcal{J}(x) + \rho\|x\|_1^1$ and this objective function is non-differentiable. Assume $\overline{X} = (\overline{x}^1, \cdots, \overline{x}^n)^T$ is the global optimal point. Consider the j^{th} variable $\overline{x}^j$. The first-order optimality conditions are:

$$\begin{cases} \bigtriangledown_j \mathcal{J}(x) + \rho sign(\overline{x}^j) = 0, & s.t.\ |\overline{x}^j| > 0 \\ \left\{ \bigtriangledown_j \mathcal{J}(x) + \rho e : e \in [-1, 1] \right\}, & s.t.\ \overline{x}^j = 0 \end{cases} \tag{6.19}$$

Function 2 Online Detecting After the Time Window

Require:
$\overline{A}$: Observation matrix;
$\overline{B}$: Unobservation matrix;
$\overline{t}$: Time delay;
ϵ: Stop criteria;
η_t: Learning rate;

Ensure:
$\mathcal{X}^* = \{x_1, \cdots, x_j\}$: The indicator vector $\mathcal{X}^*$;

1: $x_{k+1} \leftarrow$ a best guess;
2: **repeat**
3: $x_k \leftarrow x_{k+1}$;
4: $x_{k+1} \leftarrow \left\{ x_k - \eta_t(\bigtriangledown s^*(x)) \right\}$;
5: **until** $\|x_{k+1} - x_k\| \leqslant \epsilon$
6: $\mathcal{X}^* \leftarrow x_{k+1}$ in a descending order;
7: $j^* = \underset{j}{\operatorname{argmax}} |x_j^* - x_{j-1}^*|$;
8: **return** $\mathcal{X}^* = (x_1, \cdots, x_{j^*})$;

where $sign(\overline{x}^j) = 1, \overline{x}^j > 0; -1, \overline{x}^j < 0; 0, \overline{x}^j = 0$.

These conditions can be used to define a sub-gradient for each $\overline{x}^j$:

$$\bigtriangledown_j s^*(x) = \begin{cases} \bigtriangledown_j \mathcal{J}(x) + \rho sign(\overline{x}^j), & |\overline{x}^j| > 0 \\ \bigtriangledown_j \mathcal{J}(x) + \rho, & \overline{x}^j = 0, \bigtriangledown_j \mathcal{J}(x) < \rho \\ \bigtriangledown_j \mathcal{J}(x) - \rho, & \overline{x}^j = 0, \bigtriangledown_j \mathcal{J}(x) > \rho \\ 0, & \overline{x}^j = 0, -\rho \leq \bigtriangledown_j \mathcal{J}(x) \leq \rho \end{cases}. \tag{6.20}$$

Thus, based on the three solutions, the sub-gradient method uses iterations:

$$x_{k+1} = x_k - \eta_t(\bigtriangledown s^*(x)), \tag{6.21}$$

where the parameter $\eta_t > 0$ is the learning rate. In our analysis, we only consider constant learning rate with a fixed $\eta_t > 0$.

Algorithm 8 The Online Detection Algorithm

Require:
$\mathcal{G}$: Network Graph;
$\mathcal{D}$: Detectors $\{D_a \cup D_u\}$;
u: The propagation mean parameters $\mu = (\mu_1, \cdots, \mu_{|E|})^T$;
σ: The propagation variance parameters $\sigma^2 = (\sigma_1^2, \cdots, \sigma_{|E|}^2)^T$;
ϵ: Stop criteria;
Ensure:
$\mathbf{s}^* = \{s_1, \cdots, s_m\}$: A set of diffusion provenances $\mathbf{s}^*$;

$d_{\alpha(\beta)} \leftarrow \text{anchor}(D_{a(u)})$; // randomly pick anchors
if $t < T$ **then**
  for each $d_i \in D_a$ **do**
    for each $v_j \in \mathcal{V}$ **do**
      $\mathcal{P}(d_j, v_j) \leftarrow \text{BFS}\ (\mathcal{G}, d_i, v_j)$; // breath first
      $A_{ij} = \sum_{e_k \in \mathcal{P}(d_i, v_j)} \mu_k$; // matrix A
      $B_{ij} = \sum_{e_k \in \mathcal{P}(d_i, v_j)} \mu_k$; // matrix B
      if $i > \alpha$ **then**
        for each $k \leq m_a$ **do**
          $\overline{A}(:, c_k) \leftarrow A(:, c_k) - A(:, c_\alpha)$; // matrix $\overline{A}$
          $\bar{t} := t_k - t_\alpha$; // build $\bar{t}$
        end for
        $\mathcal{X}^* \leftarrow$ Function 1$(\overline{A}, \bar{t}, \epsilon, \eta_t)$; // Call Function 1
      end if
    end for
  end for
else
  for each $k \leq m_u$ **do**
    $\overline{B}(:, c_k) \leftarrow B(:, c_k) - B(:, c_\beta)$; // matrix $\overline{B}$
  end for
  $\mathcal{X}^* \leftarrow$ Function 2$(\overline{A}, \overline{B}, \bar{t}, \epsilon, \eta_t)$; // Call Function 2
end if
return $\mathbf{s}^* \leftarrow \mathcal{G}(\mathcal{X}^*)$;

6.4.4 The Online Stochastic Sub-gradient (OSS) Algorithm

In this part, we design an Online Stochastic Sub-gradient detection algorithm to continuously infer the diffusion provenances. Algorithm 8 summarizes the solution to Eq. (6.16), where the sparse non-convex regression is solved by iteratively calculating the convex program in Eq. (6.18) and the non-smooth l_1 program in

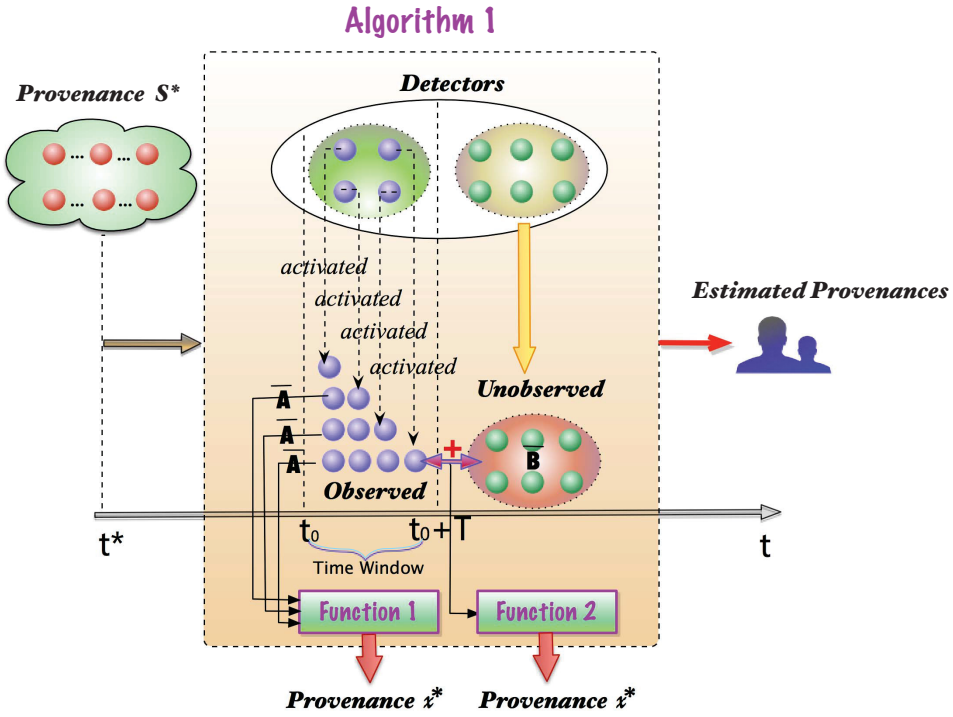


Figure 6.4: An illustration of the *OSS* algorithm. Detectors are split into two sets. The first set is observed (activated) within the time window, and the second set is unobserved (inactivated) outside the time window. Function 1 is called by OSS within the time window T, and Function 2 is called after the time window T.

Eq. (6.19). In terms of online learning, we use stochastic sub-gradient iterations to calculate $\bigtriangledown_j \mathcal{J}(x)$ during the time window T, i.e.,

$$\bigtriangledown_j \mathcal{J}(x) = (\bar{t}_i - a_i x)^T (-a_i). \tag{6.22}$$

As shown in Algorithm 8, the proposed online algorithm calls two functions for continuous learning. Function 1 corresponds to the online computation within the time window T where the detectors are activated one by one. Function 2 corresponds to the one-time computation after the time window T. The corresponding overall framework can be found in Figure 6.4.

Based on the input of the proposed algorithm and the overall framework in Figure 6.4, there are some assumptions for the OSS algorithm. 1) At least 1

detector needs to be activated during the time window. We used the time delay of both activated detectors (matrix A and D_a) and inactivated detectors (matrix B and D_u) in the proposed algorithm. During the time window, Function 1 is only called for activated detectors, and each is activated sequentially. After the time window, Function 2 is called because data from both the activated and inactivated detectors are available. Therefore, the proposed algorithm can be used to infer diffusion provenances when there is at least one activated detector (*i.e.,* the activated detector set D_a is non-null and Function 1 can be called). 2) The network structure is known prior, *i.e.,* the network graph G, the propagation mean μ and variance σ.

In the sequel, we introduce the two functions respectively.

Function 1: During the time window T, detectors are activated sequentially. To conduct continuous detection, once a detector is activated, the regression learning model is used for provenance estimation. At this stage, data from the inactivated nodes are unavailable, the parameter λ in Eq. (6.16) equals 0, i.e.,

$$s^*(x) = arg\min_{x\geq 0} \frac{1}{2}\|\bar{t} - \overline{A}^T x\|_2^2 \ + \ \rho\|x\|_1^1. \tag{6.23}$$

Each row of A and t in Eq. (6.23) denotes the shortest path and time delay from an observed detector to other detectors. By choosing an anchor node, Eq. (6.15) is used to calculate the matrix $\overline{A}$ and $\bar{t}$ in Eq. (6.23). Due to the increasing of the number of activated detectors, matrix $\overline{A}$ and $\bar{t}$ in Eq. (6.23) increase dynamically.

Function 2: After the time window T, data from both the activated and inactivated nodes are available. Thus, we can use Eq. (6.16) to locate the diffusion provenances.

We use Figure 6.4 to show the procedure of the online learning: (1) During the time window T, the detectors are activated one by one. Once a detector is activated, we use Function 1 to estimate the provenance; (2) After the time window, we obtain all the activated detectors and inactivated detectors, and Function 2 can be used for detection.

Example: Consider a network in Figure 6.5. Assume $t^* = 1, D = 3$, and the time window T is $[1, 5]$. Also, we assume detectors S_1, S_3 and S_5 are observed at

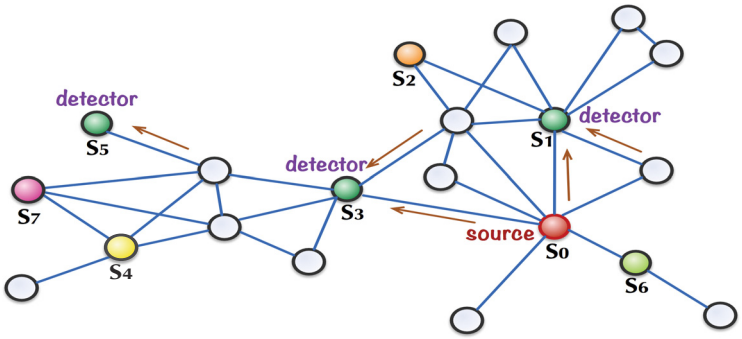


Figure 6.5: A network with one provenance S_0, and three detectors S_1, S_3 and S_5. At the unknown time $t = t^*$, propagation starts from S_0. Time delay along each edge is $\theta_i \sim N(1, 0.01)$. Monitoring time window $T = [t^*, t^* + T]$, where T is the size of the time window. Detector S_1 is activated at time point $t = t^* + \frac{1}{2}T$. Detectors S_3 and S_5 are inactivated during time window T. We only consider eight nodes $\{S_0, \cdots, S_7\}$.

time $t_1 = 1, t_3 = 3$ and $t_5 = 5$ respectively. Let S_1 be the anchor node. At $t = 1$, we have $A = [1, 0, 1, 2, 4, 4, 2, 4]^T$. Then, at $t = 3$, S_3 is activated, and we update the matrix A and t as follows,

$$A = \begin{bmatrix} 1 & 0 & 1 & 2 & 4 & 4 & 2 & 4 \\ 1 & 2 & 2 & 0 & 2 & 2 & 2 & 2 \end{bmatrix}^T, \quad t = \begin{bmatrix} 1 - t^* \\ 3 - t^* \end{bmatrix}.$$

The A and t are used to calculate $\overline{A}$ and $\bar{t}$ by Eq. (12),

$$\overline{A} = \begin{bmatrix} 0 & 2 & 1 & -2 & -2 & -2 & 0 & -2 \end{bmatrix}^T, \quad \bar{t} = [2].$$

At the last step, $t = 5$ and S_5 is observed. The variables $\overline{A}$ and $\bar{t}$ are continuously updated to

$$\overline{A} = \begin{bmatrix} 0 & 2 & 1 & -2 & -2 & -2 & 0 & -2 \\ 2 & 4 & 3 & 0 & -2 & -4 & 2 & -2 \end{bmatrix}^T, \quad \bar{t} = \begin{bmatrix} 2 \\ 4 \end{bmatrix},$$

$$\overline{B} = \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & -2 & 0 \\ 2 & 4 & 3 & 0 & -3 & -2 & 2 & -4 \end{bmatrix}^T.$$

The corresponding online solutions of x approximate the offline optimal solution when the time window T ends. We will leave the competitive boundary analysis in the future work. We have used Figure 6.5 to show how to calculate matrix $\overline{A}$, $\overline{B}$ and t in the above example. Figure 6.5 shows the shortest path from nodes $\{S_0, \cdots, S_7\}$ to anchor node S_1. For example, the elements value in matrix A are based on calculating shortest path to S_1 in Figure 6.5.

6.5 Experiments

In this section, we report experimental results on two synthetic network data sets and two real-world data sets. The experiments are designed to validate 1) the optimal parameters for the new model, 2) the superiority of the proposed model compared with benchmark methods, and 3) the performance of the proposed methods.

6.5.1 Experimental Data

We use two synthetic data sets (*Albert Barabasi [11] and Small world [132]*) and two real-world data sets (*Twitter* and *Facebook* from SNAP [1]) for parameter study, performance testing and algorithm comparison. The parameter settings are listed in Table 6.1.

Barabasi-Albert generates random scale-free networks using a preferential attachment mechanism. The network begins with an initial connected network containing β_0 nodes. New nodes are added to the network one at a time. Each new node is connected to $\beta \leq \beta_0$ existing nodes with a probability proportional to the number of links that existing nodes already have. Formally, the probability p_i that the new node is connected to node i is $p_i = k_i / \sum_j k_j$, where k_i is the degree of node i and the sum is made over all pre-existing nodes j. In this model, we set the parameter $n = 4$, which denotes the number of edges created by each new node.

Small World is defined as a network in which the typical distance ζ between two randomly chosen nodes (the number of steps required) grows proportionally

[1] http://snap.stanford.edu/data

Table 6.1: List of the four data sets.

DataSet	Nodes	Edges	Avg. Degree	Avg. Path length	Avg. Clustering Coefficient	Other parameters
Albert-Barabasi	1000	3990	7.98	3.172	0.037	$n = 4$
Small World	1000	3999	7.99	5.079	0.473	$\alpha = 4, p = 0.1$
Twitter	10269	36173	30.54	5.702	0.627	$\mu = 0.01, \sigma = 0.01$
Facebook	3320	10352	19.78	3.892	0.671	$\mu = 0.01, \sigma = 0.01$

to the logarithm of the number of nodes N in the network, that is $\zeta \propto \log N$. We use parameter α to denote that each node is connected to α nearest neighbors in the topology, and p denotes the rewiring probability. In particular, we set $alpha = 4$ and $p = 0.1$ in our experiments.

The original datasets only contain network structure information without time propagation labels. We use Gaussian distribution $N(1, 0.001)$ as the time delay distribution. We simulated the propagation by randomly setting five diffusion sources.

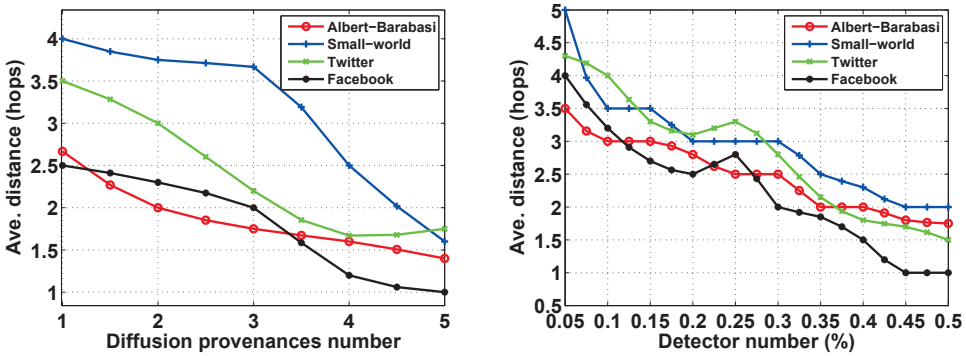
We used the Independent Cascade Model to generate cascades. When node u becomes active, it has a single chance of activating each currently inactive neighbor v. The attempt to activate succeeds independently with a probability of p_{uv}, set to $p_{uv} = 0.5$ as a constant in our experimental setting. Because the cascades are generated by the Independent Cascade Model, each node has a single chance of activating each currently inactive neighbor, so this is one-to-one communication. Thus, the datasets can be considered as half-real data.

6.5.2 Experimental Setup

We assess our methods *w.r.t* the average distance (hops) between actual locations of the diffusion provenances and the estimated locations. Smaller hops indicate that the algorithms have higher accuracy. Let S^* denote the actual provenance, and the estimated provenance set as $\hat{S}$. Since the size of S^* may not be equal to that of $\hat{S}$, we measure their distance $h(S^*, \hat{S})$ by calculating the average number of hops between each element $s_i \in \hat{S}$, $h(S^*, \hat{S}) = \frac{1}{|S^*|} \sum_{i=1}^{|\hat{S}|} \|\hat{s}_i - f(S^*, \hat{s}_i)\|^2$, where $f_i(S^*, \hat{s}_i)$ selects the node in S^* that is closest to $\hat{s}_i$, and $\|\hat{s}_i - F_i(S^*, \hat{s}_i)\|^2$ denotes the hops between nodes $\hat{s}_i$ and $f(S^*, \hat{s}_i)$.

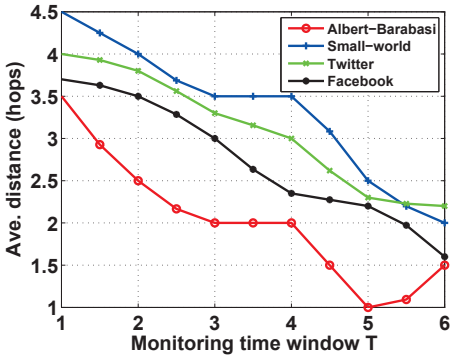
6.5.3 Experimental Results

Parameter study. We first test the parameters in Eq. (6.16) *w.r.t.* the number of diffusion provenances k, the number of detectors m, the length of monitoring time window T, and the parameters λ and ρ. The default parameters are: the number of diffusion provenances $k = 5$, the number of detectors $d = 20\% * N$,

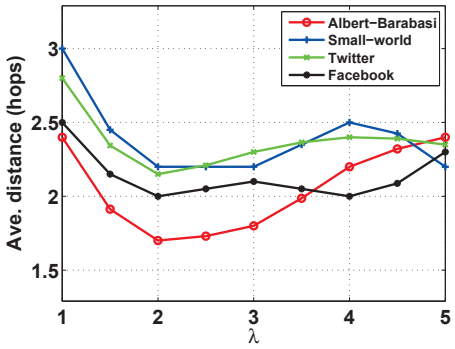


(A) # of diffusion provenances k

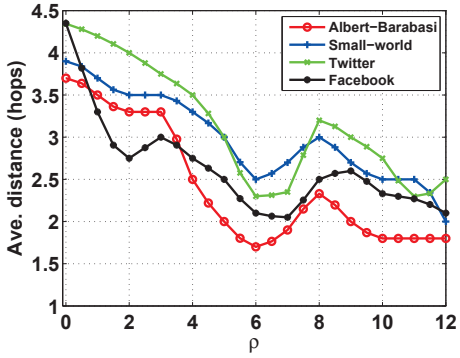
(B) # of detectors m



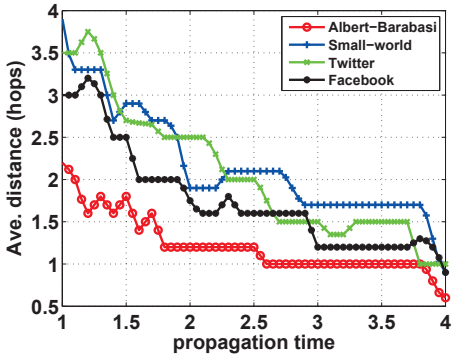
(C) window T



(D) parameter λ



(E) parameter ρ



(F) online detection

Figure 6.6: Parameter study on synthetic and real-world data sets by using the proposed regression learning model and Online Stochastic Sub-gradient algorithm. The number of hops (error rate) *w.r.t*: (A) the diffusion provenances number k; (B) the detector number m; (C) the monitoring time window T; (D) the parameter λ; (E) the parameter ρ. (F) the online detection. The average distance on synthetic/real-world data sets *w.r.t* propagation time.

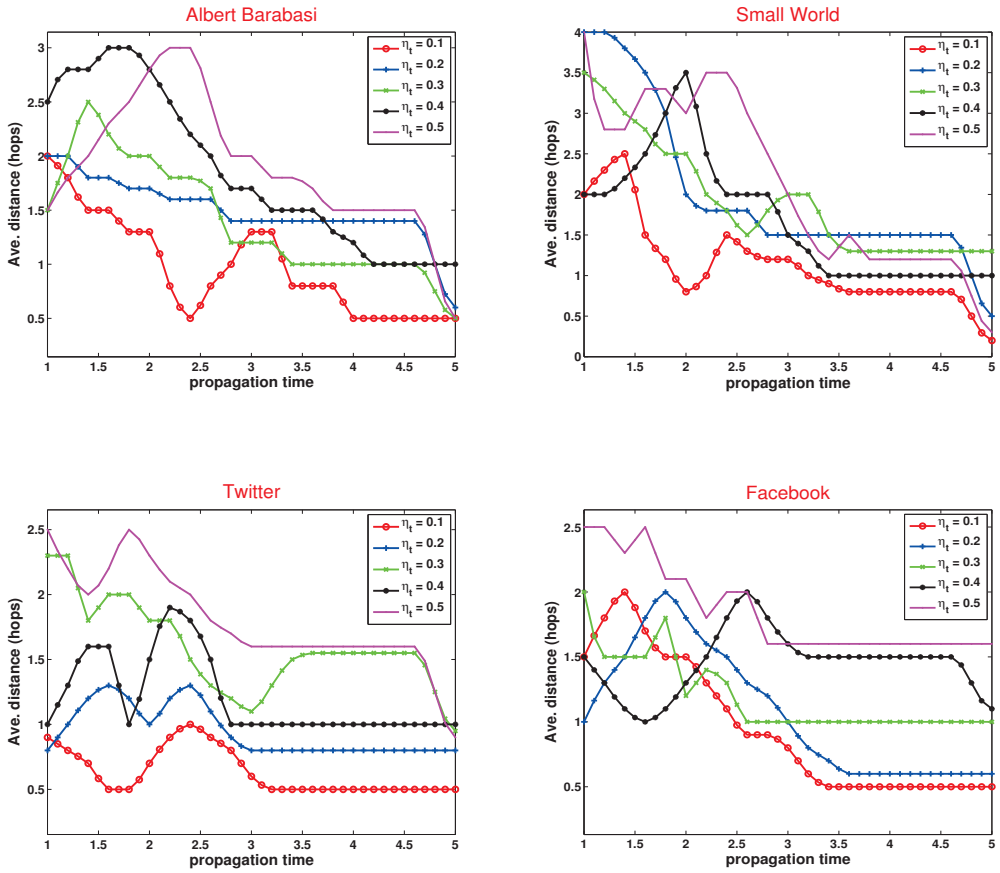


Figure 6.7: Parameter η_t in the proposed Online Stochastic Sub-gradient *OSS* algorithm.

the monitoring time window $T = 5$ minutes. The selection of provenances and detectors are random. The propagation time delay along each edge follows a Gaussian $\theta_i \sim N(1, 0.01)$. Figure 6.6 demonstrates the model performance *w.r.t.* different parameters on the synthetic and real-world datasets. Figures 6.6 (A)-(E) are parameter studies. They are conducted after the time window with Function 2 and Eq. (6.16). Figure 6.6 (F), Figure 6.7 and Figure 6.9 show the results in real-time manner, and the x-axis is the propagation time. At different propagation time, we can real-time calculate the matrix A in Eq. (6.23) and call Function 1 to conduct the online estimation of the diffusion sources. Figures 6.7 and 6.9 demonstrate the online applications of the proposed *OSS* algorithm under various learning rates η_t on difference data sets. We replicated the algorithms five times

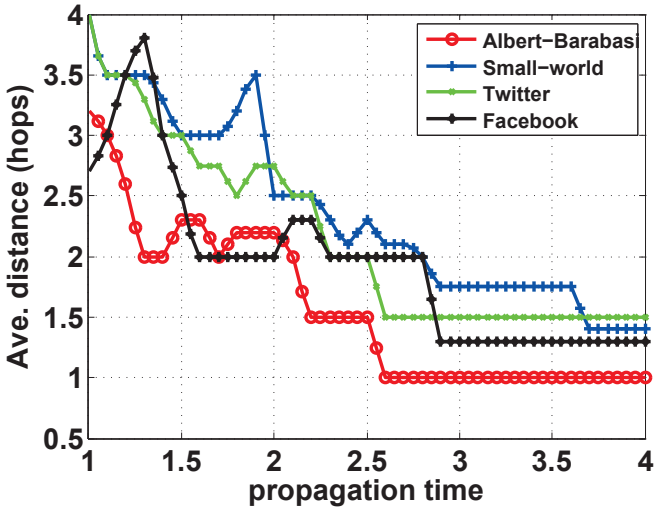


Figure 6.8: The online detection under the Linear Threshold propagation process.

and used the mean as the evaluation indicator.

The number of diffusion provenances k. From Figure 6.6(A), we can see that the error rate (hops) decreases *w.r.t.* the number of diffusion provenances. We can conclude that the more diffusion provenances, the higher probability to be found out at least one provenance.

The number of detectors m. Figure 6.6(B) shows the average distance (hops) *w.r.t* the number of detectors. The result demonstrates that the accuracy improves *w.r.t* the number of detectors.

The monitoring time window T. The monitoring time window indicates the detection time span. Figure 6.6(C) shows that the error hops drop with the time window.

The parameter λ. The parameter λ controls the preference between the activated nodes (convex part) and the inactivated nodes (concave part). As shown in Figure 6.6(D), the parameter λ should weigh the convex part and the concave part. If λ is selected too large, it overfits the inactivated nodes; otherwise, it overfits the activated nodes.

The parameter ρ. $\rho > 0$ is the regularization parameter, and restricts the size of x. If ρ is too large, the algorithm surfers from time cost, especially on large scale networks. From Figure 6.6(E), we observe the minimal hops when ρ is 6.

Continuous detection. We test the proposed online algorithm on the synthetic

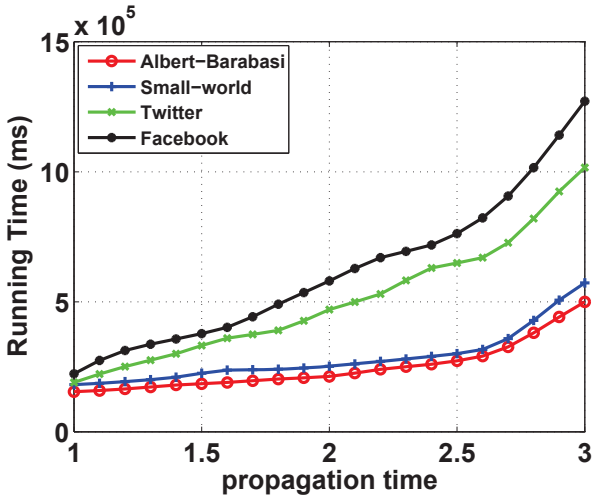


Figure 6.9: Running time *w.r.t* the propagation time on the synthetic and real-world data sets.

and real-world date sets. Also, we empirically study the learning rate parameter η_t in the sub-gradient algorithm. Figure 6.6 (F) shows the performance of the algorithm *OSS*. The hops reduce along with the propagation time because more activated detectors lead to better performance. At the end of the time window, the hops shrink as the inactivated detectors are obtained. Figure 6.7 demonstrates the *OSS* algorithm with learning rates η_t. The algorithm reaches the least average distance when η_t is 0.1.

Compare with benchmark methods. We compare the proposed non-convex sparse regression model (NSR for short) with three benchmark methods: 1) the convex-based sparse regression method (VEXR for short) [6], which uses activated nodes for regression and locating; 2) the concave-based sparse regression method (CAVR for short) [6], which uses inactivated nodes for regression; 3) *Betweenness centrality* [12], which measures a node's centrality in a network. We use betweenness centrality to measure how often a node appears on the shortest path. The betweenness centrality of a node v is given by the function: $g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the total number of the shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v; and 4) *Bonacich centrality* [15], also known as Eigenvector Centrality, measures the influence of a node in a network. The influence of nodes is based on assigning

relative scores to all nodes in the network. The centrality score of node v is defined as: $x_v = \frac{1}{\lambda}\sum_{t\in M(v)x_t} = \frac{1}{\lambda}\sum_{t\in G}\alpha_{v,t}x_t$, where G is network, $A = (\alpha_{v,t})$ be the adjacency matrix, i.e., $\alpha_{v,t} = 1$ if node v is linked to node t, and $\alpha_{v,t} = 0$ otherwise, $M(v)$ is a set of the neighbors of v, and λ is a constant (we set it as 1 in our experiment). We use UCINET 6 for Windows to compute the Bonacich centrality.

We compare the four methods on the two synthetic networks and two real-world networks. The parameters are set as default. We test two settings, the number of diffusion provenances $k = 5$ and $k = 10$ on the four synthetic datasets. For each group, we randomly sample different numbers of detectors m. From the results in Table 6.2, the results can be summarized as follows.

1). the proposed NSR model, by leveraging both activated and inactivated nodes, performs better than both CVXR and CAVR models on most of the data sets, achieving the lowest expected hops, especially when m is large. For example, on the ForestFire data set, NSR achieves the best result of 1.33 hops when $k = 10$ and $m = 0.2 * N$.

2). The NSR model performs better than the heuristic *Betweennes* model when the sample rate m/N is high. This is because the betweenness algorithm aims to find the central nodes in a network, and it cannot make proper use of information from detectors. However, when the sample rate is low, betweenness shows better results than the other three.

3). When the number of detectors increases, the error rates of NSR, CVXR and CAVR reduce significantly, among which NSR reduces the most. The performance of the betweeness model is stably poor.

4). The accuracy of centrality increases with an increased the number of sources, because centrality finds the most influential nodes in a social network. However, sometimes the source may not be among the most influential nodes. For example, malicious information may be misforwarded by one user and then spread by the other influential users. That is, centrality approaches tend to discover key infrastructure nodes or super-spreaders, but do not necessarily pinpoint the diffusions source. The accuracy increase is because

Table 6.2: Comparisons on the four data sets.

		$K = 5$				$K = 10$			
	Detectors (m)	0.01	0.05	0.1	0.2	0.01	0.05	0.1	0.2
Albert-Barabasi	NSR	**3.47±0.64**	**2.33±0.66**	**2.21±0.50**	**1.47±0.24**	3.40±0.21	**2.19±0.48**	2.08±0.45	**1.52±0.26**
	CVXR	3.48±0.25	2.37±0.31	2.38±0.69	2.17±0.34	3.72±0.51	2.38±0.47	**2.05±0.58**	1.78±0.47
	CAVR	3.82±0.66	2.77±0.68	2.99±0.26	2.10±0.43	3.86±0.44	2.98±0.63	2.93±0.57	2.20±0.39
	Betweenness	3.48±0.34	3.19±0.26	3.07±0.58	3.25±0.20	3.02±0.41	3.82±0.23	3.92±0.44	3.78±0.30
	Bonacich	3.48±0.33	3.30±0.33	3.30±0.25	3.00±0.30	**3.00±0.33**	3.55±0.30	3.00±0.44	3.44±0.23
Small-World	NSR	4.31±0.56	**3.32±0.68**	**2.30±0.28**	**2.02±0.65**	**3.27±0.48**	**3.05±0.53**	**2.28±0.30**	1.86±0.48
	CVXR	4.44±0.48	3.51±0.20	3.35±0.60	2.08±0.30	3.32±0.58	3.14±0.33	2.79±0.55	**1.49±0.20**
	CAVR	4.53±0.41	3.62±0.27	3.47±0.20	2.10±0.68	3.82±0.61	3.17±0.21	2.95±0.47	2.03±0.34
	Betweenness	**3.27±0.38**	3.46±0.48	3.11±0.22	3.09±0.40	3.32±0.60	3.17±0.66	3.26±0.29	3.52±0.25
	Bonacich	3.30±0.50	3.45±0.44	3.00±0.33	3.00±0.33	3.30±0.55	3.10±0.50	3.30±0.44	3.50±0.22
Twitter	NSR	4.33±0.51	**3.67±0.24**	**2.50±0.35**	**2.33±0.47**	**3.67±0.44**	**3.33±0.52**	2.01±0.52	**1.50±0.31**
	CVXR	3.77±0.69	4.25±0.65	3.0±0.65	2.37±0.35	**3.67±0.33**	3.67±0.55	2.43±0.40	1.62±0.67
	CAVR	4.50±0.43	4.39±0.60	3.09±0.29	2.57±0.65	3.77±0.46	3.75±0.49	2.99±0.46	2.67±0.49
	Betweenness	4.40±0.24	3.75±0.27	3.20±0.23	3.73±0.36	3.96±0.20	3.87±0.31	3.56±0.37	3.41±0.63
	Bonacich	**3.66±0.45**	3.66±0.55	3.00±0.33	3.50±0.30	3.66±0.25	3.62±0.40	3.55±0.44	3.33±0.25
Facebook	NSR	3.5±0.31	**3.20±0.21**	3.21±0.25	**2.10±0.33**	**2.80±0.32**	**2.67±0.16**	**1.85±0.51**	**1.33±0.60**
	CVXR	3.67±0.33	3.32±0.51	3.09±0.56	2.66±0.43	3.02±0.48	2.87±0.48	1.92±0.32	1.70±0.66
	CAVR	3.85±0.27	3.67±0.35	3.24± 0.66	2.52±0.59	3.50±0.31	3.06±0.54	2.12± 0.45	2.03±0.40
	Betweenness	**3.37±0.59**	3.21±0.23	3.02±0.44	3.83±0.21	3.63±0.23	3.26±0.32	3.44±0.46	3.19±0.23
	Bonacich	3.50±0.33	3.33±0.25	**3.00±0.44**	3.00±0.30	3.55±0.50	3.55±0.30	3.30±0.44	2.88±0.30

more provenances have a higher probability that the influential nodes are provenances.

5). The NSR model outperforms the other benchmarks in most cases, because centrality finds the most influential nodes in a social network. However, sometimes the source may not be among the most influential nodes. For example, malicious information may be misforwarded by one user from and then spread by other influential users. That is, centrality approaches tend to discover key infrastructure nodes or super-spreaders but do not necessarily pinpoint the diffusions source. By leveraging both activated and inactivated nodes, NSR can achieve better results than VEXR (only uses activated nodes) and CAVR (only uses inactivated nodes).

6). The NSR model outperforms the other benchmarks in most cases, because centrality finds the most influential nodes in a social network. However, sometimes the source may not be among the most influential nodes. For example, malicious information may be misforwarded by one user from and then spread by other influential users. That is, centrality approaches tend to discover key infrastructure nodes or super-spreaders but do not necessarily pinpoint the diffusions source. By leveraging both activated and inactivated nodes, NSR can achieve better results than VEXR (only uses activated nodes) and CAVR (only uses inactivated nodes).

Online algorithm in the different propagation processes. We evaluate our algorithm in the different propagation processes, *i.e.*, Linear Threshold propagation process. A node v has threshold $\theta_v \sim U[0,1]$, and is influenced by each neighbor t according to a weight b_{vt} such that: $\sum_{t\ neighbor\ of\ v} b_{v,t} \leq 1$. A node v becomes active when at least θ_v fraction of its neightbors are active $\sum_{t\ active\ neighbor\ of\ v} b_{v,t} \geq \theta_v$. Compared with Figure 6.6(F), Figure 6.8 shows the proposed online algorithm can achieve similar results in different propagation process.

Online algorithm time cost. Figure 6.9 plots the average running time of the algorithm *OSS* on the synthetic and real-world datasets. The running time raises moderately at the beginning due to the increase of activated nodes

(Function 1), and then increases sharply in the end because all the activated and inactivated detectors are used for calculation (Function 2).

6.6 Conclusions

This chapter discusses a solution for discovering the diffusion provenances in social networks in online setting. We proposed a real-time source detection algorithm by placing detectors randomly across a network. Our approach converts the problem to a regression problem, and uses an online stochastic sub-gradient algorithm to solve regression as the information is gathered from detectors in real time. This work focuses on online detection rather than offline, to meet the practical needs for early warning, real-time awareness and real-time responses to malicious information spreading within an online social network. The offline provenances detection methods assume the data is static and available all the time, resulting in the identification is made after reaching the whole diffusion snapshot. However, the proposed method is based on a new online regression learning model, which can make the identification once a detector is observed at any time. The algorithm uses a stochastic sub-gradient decent algorithm with Gaussian time delay and the shortest-path propagation to continuously detect the provenances. This work can therefore be used in time-critical security monitoring applications, such as locating false rumors in business areas.

Although our algorithm for detecting the diffusion provenances detection achieves high accuracy and meets the need for a real-time response, some limitations exist that need improvement in the future work: 1) the simulation study used real network data but the propagation process was synthetic, dubbed half-real data, so experiments on real network propagation data need to be undertaken; and 2) the proposed algorithm, based on sub-gradients, is a straightforward solution for online learning problems, however more state-of-the-art online learning algorithms could be applied to this online diffusion provenance problem, such as passive-aggressive (PA), second-order perceptron (SOP) and confidence-weight learning (CW).

This work inspires some interesting directions for future work: 1) the problem could be further extended by using popular stochastic epidemic models such as

the SR and SRI models; 2) previous mobile social network mining techniques could be used to harness geographical information to identify a culprits physical location.

Chapter 7

Conclusions and Future Work

This chapter summarize the whole thesis and provides some further research directions.

7.1 Summary of This Thesis

Graph classification is an important tool for social network and biological data analysis, where the objective is to learn and classify objective represented in graph structure. For example, chemical compounds can be represented in graph formats, predicting chemical compound activities in bioassay tests is a known graph classification problem. The main challenge in classifying graph data, compared to classifying data with feature-vector representation, is that graphs do not have features readily available so existing classification models are inapplicable to graphs. Accordingly, many research exists to mine frequent subgraphs [49, 45] as features and convert a graph into a feature vector by examining the occurrence of selected subgraphs. In traditional graph classification, graphs are assumed to be independent where each graph represents an object. In a dynamic world, it is very often that the underlying object continuously evolves by time. The change of the graph structure, with respect the temporal order, presents a new time-variant graph representation, where an object corresponds to a set of time-variant graphs. In this thesis, we formulated a new time-variant graph classification task and proposed the use of new features to represent time-variant graphs for learning and classification.

This thesis investigated a novel time-variant graph learning and classification problem with a variety of applications, *e.g.,* cascade outbreak prediction, social robots identification and online diffusion provenances detection from time-variant graphs. However, time-variant graph learning and classification is different with traditional graph classification and exposes some new challenges, as the time-variant graphs change over time, the size of subgraph features may become infinite (i.e., outbreak cascade) and non-obtainable. To handle this problem, we develop applicable features and models for effective and efficient time-variant graph learning in the thesis, including 1) Temporal shapelet feature for networked time series (Chapter 3), 2) Incremental subgraph based TVGLC (Chapter 4), and 3) Graph-shapelet based TVGLC (Chapter 5). In addition, based on the time-variant data set (i.e., cascades), we study a real application in social networks to detect information diffusion provenances timely (Chapter 6).

This thesis handles a number of fundamental problems of the time variant graph learning, and proposes a number of techniques. In summary, the dissertation solves the four research problems in Section 1.3. The research problems focus on how to find discriminative features, develop effective algorithms and apply to real applications. Accordingly, the main contributions of this dissertation are the proposed features, algorithms and applications for time-variant graph learning and classification. From the effective features perspectives, we propose a novel graph-shaplelet feature to improve the classification performance. The graph-shapelet is compact and discriminative time-variant graph features. It is more appropriately for time-variant graph than traditional subgraph features since the time-variant graph changes over time and the search space may infinite. From the view of models, we first propose a novel network regularized least square feature selection algorithm (NetRLS) to incorporate network information for shapelet selection. The NetRLS drops the independent and identically distributed assumption and enables to use rich network structure information to improve the performance. We further propose a primal-dual incremental subgraph feature selection algorithm to tackle the high-dimensional features on time-variant graphs, and a new model (ISJF) to join short-pattern subgraphs. The proposed ISF and ISJF algorithms split feature set and load into memory in a mini-batch manner, which is a significant reduction in memory and running time. From the

view of applications, we address an interesting problem to locate the provenances from large social networks. We design a real-time provenances detection algorithm based on an online regression learning model and stochastic sub-gradient algorithm. The proposed algorithm meets the practical needs for early warning, real-time awareness and real-time responses to malicious information spreading within an online social network.

Specifically,in Chapter 3, we have explored a new problem of networked time series classification, where the data contain both the typical time series data and network structure data. A network regularized least square feature selection method (NetRLS) is proposed to incorporate the network structure information for shapelet selection, accordingly. Our work drops the independent and identically distributed (i.i.d.). assumption and enables to use rich network structure information to improve the performance. The experiments and comparisons on real-world Twitter and DBLP, and validations on medical data analysis, demonstrate that NetRLS outperforms state-of-the-art time series shapelet learning algorithms and is suitable for a wide range of learning tasks. In this thesis, we only used the Euclidean distance as the measure. In the future, we will report more comparisons with other shapelets selection/learning models based on advanced time series representation and distance measures [29].

In Chapter 4, we study time-variant graph classification with incremental subgraph features. Based on the observation that subgraph features follow the *downward closure property* and long-pattern subgraph features are often buried underneath short-pattern subgraph features, we propose a *primal-dual incremental subgraph feature selection* algorithm (*ISF*) for mining incremental subgraph features, and a subgraph join feature selection algorithm (*ISJF*) to exact long-pattern subgraphs. Experiments on real-world cascade outbreak prediction in social networks demonstrate the effectiveness of the proposed models.

In Chapter 5, we proposed the use of graph-shapelet patterns as features to represent time-variant graphs for learning and classification. We argued that existing graph classification methods are all based on static settings, where training graphs are assumed to be independent of one another and the structure of each graph remains unchanged. In reality, many applications involve data with dynamic changing structures. Accordingly, a time-variant graph can be used to

represent a sequence of graphs and capture the dynamic evolution of the underlying object. The inherent structural dependency and temporal correlations of the time-variant graphs raise significant challenges, and we advocated a graph-shapelet pattern concept to detect significant changes in the time-variant graph as features. By using graph-shapelet patterns to convert a time-variant graph as a tokenized sequence, we can effectively calculate the distance between two time-variant graphs for classification. Experiments on synthetic and real-world data demonstrated that our method is much more accurate than traditional subgraph feature based approaches.

Finally, we study a case study of time-variant application, which is to real-time detect the provenances from information diffusion graphs in Chapter 6. This chapter discusses a solution for discovering the diffusion provenances in social networks in online setting. We proposed a real-time source detection algorithm by placing detectors randomly across a network. Our approach converts the problem to a regression problem, and uses an online stochastic sub-gradient algorithm to solve regression as the information is gathered from detectors in real time. This work focuses on online detection rather than offline, to meet the practical needs for early warning, real-time awareness and real-time responses to malicious information spreading within an online social network. This work can therefore be used in time-critical security monitoring applications, such as locating false rumors in business areas.

7.2 Future Work

As more and more emerged applications involve a sequence of graphs with a strict temporal order, time-variant graph classification has drawn and will continue to draw more and more attention in the research communities. Here, we outline some time-variant graph classification problems that remains unexplored in the research community, as follows:

- **Early Prediction for Time-variant Graph.** In traditional time-variant graph classification, samples are observed in full length before its class is predicted. In practical scenario, however, providing a timely output is one

of the important criteria in applications of time-variant classification. For example, an earlier decision may reduce damage when predicting the class label (i.e. outbreak or non-outbreak) of a series of cascades (time-variant graphs), such as prevent malicious information propagation as early as possible before breaking out. Therefore, an early prediction based on the structure evolution is needed for the time-variant graph. how to re-examine the existing classification methods and develop new earliness-aware models for the time-variant graph in order to achieve early prediction is a challenge in this field.

- **Imbalanced Time-variant Graph Classification.** Traditional time-variant graph classification methods can not deal with imbalanced case because they evaluate the performance on the whole data and pay less attention to rare cases in some classes. However, in real applications, the data are tend to imbalanced, such as cascade outbreak in social networks, where outburst cascades are rare compared with non-outbreak cascades. Therefore, how to deal with imbalance data combined with early prediction models over time-variant graphs is one direction in the future.

Appendix A

A. The derivation from Eq. (4.1) to Eq. (4.3)

When $\mathbf{d}$ is fixed, the inner minimization in Eq. (4.1) degenerates to a standard SVM model *w.r.t.* $\mathbf{w}$ and $\boldsymbol{\xi}$.

$$\begin{aligned} \min_{\mathbf{w},\xi,b} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1, \ldots, m. \end{aligned} \tag{1}$$

By introducing the Lagrangian multiplier $\alpha_i \geq 0$ to each constraint $y_i(w^T(x_i \odot \sqrt{d}) + b) \geq 1 - \xi_i$, we obtain

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \xi, \alpha) &= \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ &- \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}^T(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) + b) - 1 + \xi_i),\ \alpha_i \geq 0 \end{aligned} \tag{2}$$

Then by setting the derivatives of the Lagrange function to be 0 with respect to parameters $\mathbf{w}, \xi$ and b, we obtain

Appendix A

$$\begin{cases} \dfrac{\partial \mathcal{L}(\mathbf{w}, b, \xi, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) = 0 \\ \dfrac{\partial \mathcal{L}(\mathbf{w}, b, \xi, \alpha)}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \\ \dfrac{\partial \mathcal{L}(\mathbf{w}, b, \xi, \alpha)}{\partial \xi_i} = C - \alpha_i = 0 \end{cases} . \tag{3}$$

That is,

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}), \ \sum_{i=1}^{n} \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq C.$$

Plugging the above results back into Eq.(2), we obtain the dual form of the original problem as follows,

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}} \quad -\frac{1}{2} \| \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \|^2 + \mathbf{e}^T \boldsymbol{\alpha} \tag{4}$$

As the objective function $-\frac{1}{2} \| \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \|^2 + \mathbf{e}^T \boldsymbol{\alpha}$ in Eq. 4 is linear in $\mathbf{d}$ and convex in $\boldsymbol{\alpha}$, and both $\mathcal{A}$ and $\mathbf{D}$ are compact domains, Eq. (4.1) can be equivalently reformulated as follows,

$$\min_{\boldsymbol{\alpha} \in \mathcal{A}} \max_{\mathbf{d} \in \mathbf{D}} \quad \frac{1}{2} \| \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \|^2 - \mathbf{e}^T \boldsymbol{\alpha} \tag{5}$$

As both $\mathcal{A}$ and $\mathbf{D}$ are convex compact sets, the following equivalence holds by interchanging the order of $\min_{\mathbf{d} \in \mathbf{D}}$ and $\max_{\boldsymbol{\alpha} \in \mathcal{A}}$ in Eq. (4.2) based on the minimax saddle-point theorem [113],

$$\begin{aligned} &\min_{\mathbf{d} \in \mathbf{D}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} \quad -\frac{1}{2} \| \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \|^2 + \mathbf{e}^T \boldsymbol{\alpha} \\ = &\max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\mathbf{d} \in \mathbf{D}} \quad -\frac{1}{2} \| \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \|^2 + \mathbf{e}^T \boldsymbol{\alpha} \\ \Longleftrightarrow &\min_{\boldsymbol{\alpha} \in \mathcal{A}} \max_{\mathbf{d} \in \mathbf{D}} \quad \frac{1}{2} \| \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \|^2 - \mathbf{e}^T \boldsymbol{\alpha} \end{aligned}$$

B. Optimization 1 can be solved by quadratic programming with small set of

features

Let $f(\boldsymbol{\alpha}, \mathbf{d}) = \frac{1}{2}\| \sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}})\|^2 - \mathbf{e}^T \boldsymbol{\alpha}$. By introducing an additional variable $\theta \in \mathbb{R}$, the problem can be reformulated as follows:

$$\min_{\boldsymbol{\alpha} \in \mathcal{A}, \boldsymbol{\theta} \in \mathbb{R}} \boldsymbol{\theta} \quad \text{s.t. } \boldsymbol{\theta} \geq f(\boldsymbol{\alpha}, \mathbf{d}), \forall\, \mathbf{d} \in \mathcal{D}. \tag{6}$$

which is a convex quadratic programming problem. Each nonzero $\mathbf{d} \in \mathcal{D}$ defines a quadratic constraint with respect to $\boldsymbol{\alpha}$. There are as many as $(\sum_{i=0}^{B} \binom{m}{i})$ quadratic constrains in Eq. (4.4).

Because the scaling vector d is fixed, and we use $\|\mathbf{d}\| \leq B$ to encourage sparsity so that at most B subgraph features are selected. Since B is a small value, it can be solved with small set of features.

C. Derivation from Eq. 4.8 to Eq. 4.9

By introducing the Lagrangian multiplier $p_i, q_j, r_j \geq 0$ to each constraint in Eq. 4.8, we have

$$\begin{aligned} \mathcal{L}(\mathbf{d}) = & \sum_{j=1}^{t} [c_j(\alpha)]^2 d_j + \sum_{i=1}^{m} p_i [(1-\epsilon)\frac{t}{m} b_i - \sum_{j=1}^{t} d_j] \\ & + \sum_{j=1}^{t} q_j d_j + \sum_{j=1}^{t} r_j (1 - d_j) \end{aligned} \tag{7}$$

Then by setting the derivatives of the Lagrange function to be 0 with respect to parameters $\mathbf{d}$, we obtain

$$\nabla_{\mathbf{d}} \mathcal{L}(\mathbf{d}) = [c_j(\alpha)]^2 - \sum_{i=1}^{m} p_i + q_j - r_j = 0 \tag{8}$$

That is, $[c_j(\alpha)]^2 = \sum_{i=1}^{m} p_i - q_j + r_j$

As $r_j \geq 0$, we have $\sum_{i=1}^{m} p_i + r_j \geq [c_j(\alpha)]^2$. By plugging the above result to Eq. 8, we have the dual problem as follows,

$$\begin{aligned} \min_{p,r} \quad & \sum_{i=1}^{m} b_i p_i (1-\epsilon)\frac{t}{m} + \sum_{j=1}^{t} r_j \\ \text{s.t.} \quad & \sum_{i=1}^{m} p_i + r_j \geq [c_j(\alpha)]^2,\ j = 1, ..., t \\ & p_i, r_j \geq 0, 1 \leq i \leq m. \end{aligned} \tag{9}$$

For simplicity, we use only one Lagrange multiplier and map the space of r_j to the same space of p_i, the dual problem can be rewritten as follows,

$$\begin{aligned} \min_{p} \quad & \sum_{i=1}^{m} b_i p_i (1-\epsilon)\frac{t}{m} + \sum_{i=m+1}^{m+t} p_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} p_i + p_{i+j} \geq [c_j(\alpha)]^2,\ j = 1, ..., t \\ & p_i \geq 0, 1 \leq i \leq m. \end{aligned} \tag{10}$$

References

[1] C. C. Aggarwal. "On classification of graph streams". In: *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM. 2011, pp. 652–663 (cit. on p. 16).

[2] C. C. Aggarwal. "On Classification of Graph Streams". In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 2011, pp. 652–663 (cit. on p. 79).

[3] C. C. Aggarwal and N. Li. "On node classification in dynamic content-based networks". In: *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM. 2011, pp. 355–366 (cit. on p. 16).

[4] C. C. Aggarwal et al. "A framework for on-demand classification of evolving data streams". In: *IEEE Transactions on Knowledge and Data Engineering* 18.5 (2006), pp. 577–589 (cit. on p. 20).

[5] S. Agrawal, Z. Wang, and Y. Ye. "A dynamic near-optimal algorithm for online linear programming". In: *arXiv preprint arXiv:0911.2974* (2009) (cit. on pp. 55–57).

[6] N. E. Aguilera, L. Forzani, and P. Morin. "On convex regression estimators". In: *arXiv preprint arXiv:1006.2859* (2010) (cit. on p. 129).

[7] M. Babaioff et al. “Online auctions and generalized secretary problems”. In: *ACM SIGecom Exchanges* 7.2 (2008), p. 7 (cit. on p. 56).

[8] L. Bai and E. R. Hancock. “Fast depth-based subgraph kernels for unattributed graphs”. In: *Pattern Recognition* 50 (2016), pp. 233 –245 (cit. on p. 21).

[9] Y. Bai et al. “Evolutionary lazy learning for Naive Bayes classification”. In: *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE. 2016, pp. 3124–3129 (cit. on p. 11).

[10] A.-L. Barabási and R. Albert. “Emergence of scaling in random networks”. In: *science* 286.5439 (1999), pp. 509–512 (cit. on p. 63).

[11] A.-L. Barabasi and R. Albert. “Emergence of Scaling in Random Networks”. In: *Science* 286.5439 (1999), pp. 509–512 (cit. on p. 123).

[12] M. Barthelemy. “Betweenness centrality in large complex networks”. In: *The European Physical Journal B-Condensed Matter and Complex Systems* 38.2 (2004), pp. 163–168 (cit. on pp. 23, 129).

[13] M. Belkin, P. Niyogi, and V. Sindhwani. “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples”. In: *The Journal of Machine Learning Research* 7 (2006), pp. 2399–2434 (cit. on p. 17).

[14] B. Bollobás. “Random graphs”. In: *Modern Graph Theory*. Springer, 1998, pp. 215–252 (cit. on p. 63).

[15] P. Bonacich. “Power and centrality: A family of measures”. In: *American journal of sociology* (1987), pp. 1170–1182 (cit. on pp. 23, 129).

[16] S. Boyd. "Convex optimization of graph Laplacian eigenvalues". In: *Proceedings of the International Congress of Mathematicians*. Vol. 3. 1-3. 2006, pp. 1311–1319 (cit. on p. 28).

[17] N. Buchbinder and J. Naor. "Online primal-dual algorithms for covering and packing". In: *Mathematics of Operations Research* 34.2 (2009), pp. 270–286 (cit. on pp. 19, 56).

[18] J. Burgess et al. "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks." In: *Infocom*. 2006 (cit. on p. 15).

[19] L. Chi, B. Li, and X. Zhu. "Fast graph stream classification using discriminative clique hashing". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2013, pp. 225–236 (cit. on p. 16).

[20] K. Costello. "Random Walks on Directed Graphs". In: (2005) (cit. on p. 28).

[21] K. Crammer et al. "Online Passive-Aggressive Algorithms". In: *J. Mach. Learn. Res.* 7 (2006), pp. 551–585 (cit. on p. 23).

[22] P. Cui et al. "Cascading outbreak prediction in networks: a data-driven approach". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2013, pp. 901–909 (cit. on pp. 4, 19, 65).

[23] C. Deng et al. "Weakly supervised multi-graph learning for robust image reranking". In: *IEEE Transactions on Multimedia* 16.3 (2014), pp. 785–795 (cit. on p. 1).

[24] M. Deshpande et al. “Frequent substructure-based approaches for classifying chemical compounds”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.8 (2005), pp. 1036–1050 (cit. on pp. 1, 2).

[25] M. Deshpande et al. “Frequent substructure-based approaches for classifying chemical compounds”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.8 (2005), pp. 1036–1050 (cit. on p. 46).

[26] C. Ding and L. Zhang. “Double adjacency graphs-based discriminant neighborhood embedding”. In: *Pattern Recognition* 48.5 (2015), pp. 1734 –1742 (cit. on p. 21).

[27] H. Ding et al. “Querying and mining of time series data: experimental comparison of representations and distance measures”. In: *Proceedings of The VLDB Endowment (PVLDB)* 1.2 (2008), pp. 1542–1552 (cit. on p. 17).

[28] H. Ding et al. “Querying and mining of time series data: experimental comparison of representations and distance measures”. In: *Proceedings of the VLDB Endowment* 1.2 (2008), pp. 1542–1552 (cit. on pp. 29, 36).

[29] H. Ding et al. “Querying and mining of time series data: experimental comparison of representations and distance measures”. In: *Proceedings of the VLDB Endowment* 1.2 (2008), pp. 1542–1552 (cit. on p. 137).

[30] W. Dong, W. Zhang, and C. W. Tan. “Rooting out the rumor culprit from suspects”. In: *ISIT*. 2013, pp. 2671–2675 (cit. on p. 23).

[31] N. Du et al. “Scalable Influence Estimation in Continuous-Time Diffusion Networks.” In: *NIPS*. 2013, pp. 3147–3155 (cit. on p. 23).

[32] N. Eagle, A. S. Pentland, and D. Lazer. "Inferring friendship network structure by using mobile phone data". In: *Proceedings of the National Academy of Sciences* 106.36 (2009), pp. 15274–15278 (cit. on p. 94).

[33] M. Elfeky, W. Aref, and A. Elmagarmid. "Periodicity detection in time series databases". In: *IEEE Transactions on Knowledge and Data Engineering* 17.7 (2005), pp. 875–887 (cit. on p. 17).

[34] H. Fei and J. Huan. "Structured sparse boosting for graph classification". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 9.1 (2014), p. 4 (cit. on p. 18).

[35] H. Fei and J. Huan. "Structured Sparse Boosting for Graph Classification". In: *ACM Transactions on Knowledge Discovery from Data* 9.1 (2014), 4:1–4:22. ISSN: 1556-4681 (cit. on p. 21).

[36] M. Gomez-rodriguez and D. B. B. Schlkopf. "Uncovering the temporal dynamics of diffusion networks". In: *ICML*. 2011, pp. 561–568 (cit. on pp. 23, 111).

[37] J. Grabocka et al. "Learning Time-series Shapelets". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2014, pp. 392–401 (cit. on pp. 17, 80).

[38] J. Grabocka et al. "Learning time-series shapelets". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 392–401 (cit. on pp. 25, 36, 41).

[39] J. Grabocka et al. "Learning Time-series Shapelets". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery*

and Data Mining (KDD). New York, New York, USA, 2014, pp. 392–401 (cit. on p. 25).

[40] Q. Gu and J. Han. "Towards feature selection in network". In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM. 2011, pp. 1175–1184 (cit. on pp. 17, 28, 31).

[41] I. Guyon and A. Elisseeff. "An introduction to variable and feature selection". In: *The Journal of Machine Learning Research* 3 (2003), pp. 1157–1182 (cit. on p. 17).

[42] B. Hartmann and N. Link. "Gesture recognition with inertial sensors and optimized DTW prototypes". In: *Proceedings of the IEEE International Conference on Systems Man and Cybernetics (SMC)*. 2010, pp. 2102–2109 (cit. on p. 17).

[43] R. Hong et al. "Flickr Circles: Aesthetic Tendency Discovery by Multi-view Regularized Topic Modeling". In: *IEEE Transactions on Multimedia* PP.99 (2016), pp. 1–1 (cit. on p. 3).

[44] A. Inokuchi and T. Washio. "A Fast Method to Mine Frequent Subsequences from Graph Sequence Data". In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2008, pp. 303–312 (cit. on pp. 81, 88, 95).

[45] A. Inokuchi. "Mining generalized substructures from a set of labeled graphs". In: *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*. IEEE. 2004, pp. 415–418 (cit. on pp. 18, 46, 50, 135).

[46] A. Inokuchi. “Mining generalized substructures from a set of labeled graphs”. In: *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*. IEEE. 2004, pp. 415–418 (cit. on pp. 18, 46).

[47] A. Inokuchi. “Mining generalized substructures from a set of labeled graphs”. In: *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*. IEEE. 2004, pp. 415–418 (cit. on pp. 19, 46).

[48] A. Inokuchi. “Mining generalized substructures from a set of labeled graphs”. In: *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*. IEEE. 2004, pp. 415–418 (cit. on p. 20).

[49] A. Inokuchi. “Mining generalized substructures from a set of labeled graphs”. In: *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*. IEEE. 2004, pp. 415–418 (cit. on pp. 46, 135).

[50] A. Inokuchi. “Mining generalized substructures from a set of labeled graphs”. In: *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*. IEEE. 2004, pp. 415–418 (cit. on p. 46).

[51] M. Izadi and P. Saeedi. “Robust Weighted Graph Transformation Matching for Rigid and Nonrigid Image Registration”. In: *IEEE Transactions on Image Processing* 21.10 (2012), pp. 4369–4382 (cit. on p. 21).

[52] P. Jacquet, B. Mans, and G. Rodolakis. “Information propagation speed in mobile and delay tolerant networks”. In: *IEEE Transactions on Information Theory* 56.10 (2010), pp. 5001–5015 (cit. on p. 15).

[53] N. Karamchandani and M. Franceschetti. “Rumor source detection under probabilistic sampling”. In: *ISIT*. 2013, pp. 2184–2188 (cit. on pp. 23, 106).

[54] H. Kashima, K. Tsuda, and A. Inokuchi. "Marginalized kernels between labeled graphs". In: *ICML*. Vol. 3. 2003, pp. 321–328 (cit. on p. 18).

[55] E. Keogh and S. Kasetty. "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration". In: *Data Mining and Knowledge Discovery* 7.4 (2003), pp. 349–371 (cit. on p. 17).

[56] E. Keogh and C. A. Ratanamahatana. "Exact indexing of dynamic time warping". In: *Knowledge and information systems* 7.3 (2005), pp. 358–386 (cit. on p. 25).

[57] M. Khosroshahy, M. K. M. Ali, and D. Qiu. "The SIC botnet lifecycle model: A step beyond traditional epidemiological models". In: *Computer Networks* 57.2 (2013), pp. 404–421 (cit. on p. 23).

[58] J. Kleinberg. "Bursty and Hierarchical Structure in Streams". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2002, pp. 91–101 (cit. on p. 17).

[59] X. Kong, W. Fan, and P. S. Yu. "Dual active feature and sample selection for graph classification". In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. 2011, pp. 654–662 (cit. on p. 3).

[60] X. Kong and P. S. Yu. "Semi-supervised feature selection for graph classification". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2010, pp. 793–802 (cit. on p. 18).

[61] X. Kong et al. “Discriminative Feature Selection for Uncertain Graph Classification”. In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 2013, pp. 82–93 (cit. on pp. 3, 21).

[62] T. Kudo, E. Maeda, and Y. Matsumoto. “An application of boosting to graph classification”. In: *Advances in neural information processing systems*. 2004, pp. 729–736 (cit. on p. 18).

[63] J. Langford, L. Li, and T. Zhang. “Sparse Online Learning via Truncated Gradient”. In: *J. Mach. Learn. Res.* 10 (2009), pp. 777–801 (cit. on p. 23).

[64] P. H. Las-Casas et al. “SpaDeS: Detecting spammers at the source network”. In: *Computer Networks* 57.2 (2013), pp. 526–539 (cit. on p. 22).

[65] J. Leskovec, L. Backstrom, and J. Kleinberg. “Meme-tracking and the dynamics of the news cycle”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2009, pp. 497–506 (cit. on p. 62).

[66] J. Leskovec, J. Kleinberg, and C. Faloutsos. “Graphs over time: densification laws, shrinking diameters and possible explanations”. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, pp. 177–187 (cit. on pp. 63, 64).

[67] J. Leskovec et al. “Cascading Behavior in Large Blog Graphs: Patterns and a Model”. In: *Proceedings of the Society of Applied and Industrial Mathematics: Data Mining (SDM)*. 2007, pp. 551–556 (cit. on p. 102).

[68] J. Leskovec et al. “Cost-effective outbreak detection in networks”. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2007, pp. 420–429 (cit. on p. 19).

[69] J. Leskovec et al. “Kronecker graphs: An approach to modeling networks”. In: *Journal of Machine Learning Research* 11.Feb (2010), pp. 985–1042 (cit. on p. 15).

[70] B. Li et al. “Nested subtree hash kernels for large-scale graph classification over streams”. In: *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE. 2012, pp. 399–408 (cit. on p. 16).

[71] B. Li, N. Chen, and U. Schlichtmann. “Statistical Timing Analysis for Latch-Controlled Circuits With Reduced Iterations and Graph Transformations”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31.11 (2012), pp. 1670–1683 (cit. on p. 21).

[72] J. Lin et al. “Finding Motifs in Time Series”. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2002, pp. 53–68 (cit. on p. 17).

[73] J. Lines et al. “A shapelet transform for time series classification”. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, pp. 289–297 (cit. on p. 25).

[74] H. Liu et al. “Feature Selection: An Ever Evolving Frontier in Data Mining.” In: *FSDM* 10 (2010), pp. 4–13 (cit. on p. 20).

[75] A. Y. Lokhov et al. “Inferring the origin of an epidemic with a dynamic message-passing algorithm”. In: *Phys. Rev. E* 90 (1 2014), p. 012801 (cit. on p. 106).

[76] M. López and G. Still. “Semi-infinite programming”. In: *European Journal of Operational Research* 180.2 (2007), pp. 491–518 (cit. on p. 19).

[77] W. Luo, W. P. Tay, and M. Leng. “How to Identify an Infection Source with Limited Observations”. In: *IEEE J. Sel. Topics Signal Process.* 8.4 (2014), pp. 586 –597 (cit. on pp. 23, 106).

[78] A. Maronidis, A. Tefas, and I. Pitas. “Subclass Graph Embedding and a Marginal Fisher Analysis paradigm”. In: *Pattern Recognition* 48.12 (2015), pp. 4024 –4035 (cit. on p. 21).

[79] M. M. Masud et al. “Classification and adaptive novel class detection of feature-evolving data streams”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.7 (2013), pp. 1484–1497 (cit. on p. 20).

[80] A. Mcgovern et al. “Identifying Predictive Multi-dimensional Time Series Motifs: An Application to Severe Weather Prediction”. In: *Data Mining and Knowledge Discovery* 22.1-2 (2011), pp. 232–258 (cit. on p. 17).

[81] A. Mueen, E. Keogh, and N. Young. “Logical-shapelets: An Expressive Primitive for Time Series Classification”. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2011, pp. 1154–1162 (cit. on pp. 17, 80).

[82] S. Nijssen and J. N. Kok. “A quickstart in frequent structure mining can make a difference”. In: *Proceedings of the tenth ACM SIGKDD interna-*

tional conference on Knowledge discovery and data mining. ACM. 2004, pp. 647–652 (cit. on p. 20).

[83] G. Niu et al. "Multi-source-driven asynchronous diffusion model for video-sharing in online social networks". In: *IEEE Transactions on Multimedia* 16.7 (2014), pp. 2025–2037 (cit. on p. 102).

[84] K. Okamoto, W. Chen, and X.-Y. Li. "Ranking of closeness centrality for large-scale social networks". In: *Frontiers in Algorithmics*. 2008, pp. 186–195 (cit. on p. 23).

[85] E. Onur et al. "Measurement-based replanning of cell capacities in GSM networks". In: *Computer Networks* 39.6 (2002), pp. 749–767 (cit. on p. 113).

[86] F. J. Ortega et al. "Propagation of trust and distrust for the detection of trolls in a social network". In: *Computer Networks* 56.12 (2012), pp. 2884–2895 (cit. on p. 22).

[87] S. Pan, J. Wu, and X. Zhu. "CogBoost: Boosting for Fast Cost-Sensitive Graph Classification". In: *IEEE Transactions on Knowledge and Data Engineering* 27.11 (2015), pp. 2933–2946 (cit. on p. 21).

[88] S. Pan et al. "Finding the best not the most: regularized loss minimization subgraph selection for graph classification". In: *Pattern Recognition* 48.11 (2015), pp. 3783–3796 (cit. on p. 65).

[89] S. Pan et al. "Graph ensemble boosting for imbalanced noisy graph stream classification". In: *IEEE transactions on cybernetics* 45.5 (2015), pp. 954–968 (cit. on p. 20).

[90] S. Pan et al. “Graph Ensemble Boosting for Imbalanced Noisy Graph Stream Classification”. In: *IEEE Transactions on Cybernetics* 45.5 (2015), pp. 954–968 (cit. on pp. 22, 79).

[91] S. Perkins and J. Theiler. “Online feature selection using grafting”. In: *ICML*. 2003, pp. 592–599 (cit. on pp. 19, 45).

[92] S. Perkins and J. Theiler. “Online Feature Selection Using Grafting”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2003, pp. 592–599 (cit. on p. 22).

[93] P. C. Pinto, P. Thiran, and M. Vetterli. “Locating the Source of Diffusion in Large-Scale Networks”. In: *Phys. Rev. Lett.* 109 (2012), p. 068702 (cit. on p. 106).

[94] A. Pnueli. “Near-optimal supervised feature selection among frequent subgraphs”. In: (2009) (cit. on p. 18).

[95] A. Pnueli. “Near-optimal supervised feature selection among frequent subgraphs”. In: (2009) (cit. on p. 21).

[96] B. A. Prakash, J. Vreeken, and C. Faloutsos. “Spotting Culprits in Epidemics: How Many and Which Ones?” In: *ICDM*. 2012, pp. 11–20 (cit. on p. 106).

[97] T. Rakthanmanon and E. Keogh. “Fast shapelets: A scalable algorithm for discovering time series shapelets”. In: *Proceedings of the thirteenth SIAM conference on data mining (SDM)*. SIAM. 2013, pp. 668–676 (cit. on pp. 25, 36).

[98] K. Riesen and H. Bunke. "Graph Classification by Means of Lipschitz Embedding". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 39.6 (2009), pp. 1472–1483 (cit. on p. 21).

[99] K. Riesen and H. Bunke. "Graph classification by means of Lipschitz embedding". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.6 (2009), pp. 1472–1483 (cit. on p. 18).

[100] K. Riesen and H. Bunke. *Graph classification and clustering based on vector space embedding*. World Scientific Publishing Co., Inc., 2010 (cit. on p. 18).

[101] L. Rossi, A. Torsello, and E. R. Hancock. "Unfolding Kernel embeddings of graphs: Enhancing class separation through manifold learning". In: *Pattern Recognition* 48.11 (2015), pp. 3357 –3370 (cit. on p. 21).

[102] H. Sabirin and M. Kim. "Moving object detection and tracking using a spatio-temporal graph in H. 264/AVC bitstreams for video surveillance". In: *IEEE Transactions on Multimedia* 14.3 (2012), pp. 657–668 (cit. on p. 3).

[103] H. Saigo et al. "gBoost: a mathematical programming approach to graph classification and regression". In: *Machine Learning* 75.1 (2008), pp. 69–89 (cit. on p. 21).

[104] T. Sakaki, M. Okazaki, and Y. Matsuo. "Tweet analysis for real-time event detection and earthquake reporting system development". In: *Knowledge and Data Engineering, IEEE Transactions on* 25.4 (2013), pp. 919–931 (cit. on p. 5).

[105] J. Sang and C. Xu. "Robust face-name graph matching for movie character identification". In: *IEEE Transactions on Multimedia* 14.3 (2012), pp. 586–596 (cit. on p. 21).

[106] *SECURITY FOCUS REPORT: Spam in Today's Business World*. Tech. rep. TREND MICRO, 2011 (cit. on p. 105).

[107] H. Seo, J. Kim, and M.-S. Kim. "GStream: A Graph Streaming Processing Method for Large-scale Graphs on GPUs". In: *SIGPLAN Not.* 50.8 (2015), pp. 253–254. ISSN: 0362-1340 (cit. on p. 79).

[108] F. Serratosa and X. Cortes. "Interactive graph-matching using active query strategies". In: *Pattern Recognition* 48.4 (2015), pp. 1364 –1373 (cit. on p. 21).

[109] D. Shah and T. Zaman. "Rumors in a Network: Who's the Culprit?" In: *IEEE Trans. Inf. Theor.* 57.8 (2011), pp. 5163–5181 (cit. on pp. 22, 23, 106).

[110] D. Shah and T. Zaman. "Detecting sources of computer viruses in networks: theory and experiment". In: *ACM SIGMETRICS Performance Evaluation Review*. Vol. 38. 1. 2010, pp. 203–214 (cit. on p. 23).

[111] S. Shalev-Shwartz and Y. Singer. "Online learning meets optimization in the dual". In: *International Conference on Computational Learning Theory*. Springer. 2006, pp. 423–437 (cit. on p. 19).

[112] M. Sion et al. "On general minimax theorems". In: *Pacific J. Math* 8.1 (1958), pp. 171–176 (cit. on p. 52).

[113] M. Sion et al. “On general minimax theorems”. In: *Pacific J. Math* 8.1 (1958), pp. 171–176 (cit. on p. 142).

[114] M. Takác et al. “Mini-Batch Primal and Dual Methods for SVMs.” In: *ICML (3)*. 2013, pp. 1022–1030 (cit. on p. 19).

[115] M. Tan, I. W. Tsang, and L. Wang. “Towards ultrahigh dimensional feature selection for big data.” In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1371–1429 (cit. on p. 51).

[116] M. Tan, L. Wang, and I. W. Tsang. “Learning sparse svm for feature selection on very high dimensional datasets”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 1047–1054 (cit. on p. 20).

[117] J. Tang, S. Alelyani, and H. Liu. “Feature selection for classification: A review”. In: *Data Classification: Algorithms and Applications* (2014), p. 37 (cit. on p. 20).

[118] K.-C. Toh and S. Yun. “An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems”. In: *Pacific Journal of optimization* 6.615-640 (2010), p. 15 (cit. on p. 31).

[119] L. H. Ungar et al. “Streaming Feature Selection using IIC.” In: *AISTATS*. 2005 (cit. on p. 19).

[120] L. H. Ungar et al. “Streaming feature selection using iic”. In: *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*. 2005, p. 357 (cit. on p. 22).

[121] M. Varma and B. R. Babu. "More generality in efficient multiple kernel learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM. 2009, pp. 1065–1072 (cit. on p. 51).

[122] C.-D. Wang, J.-H. Lai, and P. Yu. "NEIWalk: Community Discovery in Dynamic Content-Based Networks". In: *IEEE Trans. Knowl. Data Eng.* 26.7 (2014), pp. 1734–1748 (cit. on p. 23).

[123] H. Wang et al. "Defragging subgraph features for graph classification". In: *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM. 2015, pp. 1687–1690 (cit. on pp. 11, 48).

[124] H. Wang et al. "Online diffusion source detection in social networks". In: *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE. 2015, pp. 1–8 (cit. on pp. 11, 109).

[125] H. Wang et al. "Mining subcascade features for cascade outbreak prediction in big networks". In: *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE. 2016, pp. 3942–3949 (cit. on p. 11).

[126] H. Wang et al. "Temporal Feature Selection on Networked Time Series". In: *arXiv preprint arXiv:1612.06856* (2016) (cit. on p. 11).

[127] H. Wang et al. "Time-Variant Graph Classification". In: *arXiv preprint arXiv:1609.04350* (2016) (cit. on pp. 11, 82).

[128] H. Wang et al. "Towards large-scale social networks with online diffusion provenance detection". In: *Computer Networks* (2016) (cit. on pp. 11, 109).

[129] H. Wang et al. “Incremental Subgraph Feature Selection for Graph Classification”. In: *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2017), pp. 128–142 (cit. on p. 11).

[130] S. Wang et al. “Advanced weight graph transformation matching algorithm”. In: *IET Computer Vision* 9.6 (2015), pp. 960–966 (cit. on p. 21).

[131] D. J. Watts and S. H. Strogatz. “Collective dynamics of’small-world’networks.” In: *Nature* 393.6684 (1998), pp. 409–10 (cit. on p. 93).

[132] D. J. Watts and S. H. Strogatz. “Collective dynamics of’small-world’networks.” In: *Nature* 393.6684 (1998), pp. 409–10 (cit. on p. 123).

[133] D. J. Watts and S. H. Strogatz. “Collective dynamics of small-worldnetworks”. In: *nature* 393.6684 (1998), pp. 440–442 (cit. on pp. 63, 64).

[134] L. Wei et al. “Assumption-free Anomaly Detection in Time Series”. In: *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*. 2005, pp. 237–240 (cit. on p. 17).

[135] J. Wu et al. “Bag Constrained Structure Pattern Mining for Multi-Graph Classification”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.10 (2014), pp. 2382–2396 (cit. on p. 1).

[136] J. Wu et al. “Boosting for Multi-Graph Classification”. In: *IEEE Transactions on Cybernetics* 45.3 (2015), pp. 416–429 (cit. on p. 1).

[137] J. Wu et al. “Locally Weighted Learning: How and When Does it Work in Bayesian Networks?” In: *International Journal of Computational Intelligence Systems* 8.sup1 (2015), pp. 63–74 (cit. on p. 11).

[138] J. Wu et al. “Learning with both unlabeled data and query logs for image search”. In: *Computers & Electrical Engineering* 40.3 (2014), pp. 964–973 (cit. on p. 11).

[139] X. Wu et al. “Online feature selection with streaming features”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.5 (2013), pp. 1178–1192 (cit. on pp. 19, 45–47).

[140] X. Xi et al. “Fast Time Series Classification Using Numerosity Reduction”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2006, pp. 1033–1040 (cit. on p. 17).

[141] Z. Xing et al. “Extracting Interpretable Features for Early Classification on Time Series.” In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 2011, pp. 247–258 (cit. on p. 17).

[142] X. Yan and J. Han. “gSpan: graph-based substructure pattern mining”. In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2002, pp. 721–724 (cit. on pp. 20, 21, 65, 96).

[143] X. Yan, X Zhou, and J. Han. “Mining closed relational graphs with connectivity constraints”. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, pp. 324–333 (cit. on p. 47).

[144] X. Yan et al. “Feature-based similarity search in graph structures”. In: *ACM Transactions on Database systems (TODS)* 31.4 (2006), pp. 1418–1453 (cit. on p. 46).

[145] D. Yankov et al. “Detecting Time Series Motifs Under Uniform Scaling”. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2007, pp. 844–853 (cit. on p. 17).

[146] Y. Yao and L. Holder. “Scalable svm-based classification in dynamic graphs”. In: *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE. 2014, pp. 650–659 (cit. on p. 16).

[147] L. Ye and E. Keogh. “Time Series Shapelets: A New Primitive for Data Mining”. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2009, pp. 947–956 (cit. on pp. 17, 80, 89).

[148] L. Ye and E. Keogh. “Time series shapelets: a novel technique that allows accurate, interpretable and fast classification”. In: *Data mining and knowledge discovery* 22.1-2 (2011), pp. 149–182 (cit. on pp. 25, 27).

[149] Y. Zhai, Y.-S. Ong, and I. W. Tsang. “The Emerging” Big Dimensionality””. In: *IEEE Computational Intelligence Magazine* 9.3 (2014), pp. 14–26 (cit. on p. 20).

[150] P. Zhang et al. “A framework for application-driven classification of data streams”. In: *Neurocomputing* 92 (2012), pp. 170 –182 (cit. on p. 117).

[151] S. Zhang et al. “Discovering small-world in association link networks for association learning”. In: *World Wide Web* 17.2 (2014), pp. 229–254 (cit. on p. 72).

[152] X. Zhang et al. “Data stream clustering with affinity propagation”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.7 (2014), pp. 1644–1656 (cit. on p. 20).

[153] Z. Zhang et al. “Hierarchical facial landmark localization via cascaded random binary patterns”. In: *Pattern Recognition* 48.4 (2015), pp. 1277–1288 (cit. on p. 102).

[154] J. Zhao et al. “Whom to follow: Efficient followee selection for cascading outbreak detection on online social networks”. In: *Computer Networks* 75 (2014), pp. 544–559 (cit. on p. 23).

[155] J. Zhou et al. “Streamwise feature selection”. In: *Journal of Machine Learning Research* 7.Sep (2006), pp. 1861–1885 (cit. on pp. 19, 45).

[156] J. Zhou et al. “Streamwise Feature Selection”. In: *Journal of Machine learning Research* 7 (2006), pp. 1861–1885 (cit. on p. 22).

[157] K. Zhu and L. Ying. “Information Source Detection in the SIR Model: A Sample-Path-Based Approach”. In: *IEEE/ACM Trans. Netw.* PP.99 (2015), pp. 1–1 (cit. on pp. 22, 106).

[158] Y. Zhu et al. “Graph Classification: A Diversified Discriminative Feature Selection Approach”. In: *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*. 2012, pp. 205–214 (cit. on pp. 3, 21).

[159] Y. Zhu et al. “High efficiency and quality: large graphs matching”. In: *The VLDB Journal* 22.3 (2013), pp. 345–368 (cit. on p. 50).

[160] H. Zhuge and J. Zhang. "Topological Centrality and Its e-Science Applications". In: *J. Am. Soc. Inf. Sci. Technol.* 61.9 (2010), pp. 1824–1841 (cit. on p. 23).

[161] I. Zliobaite et al. "MOA Concept Drift Active Learning Strategies for Streaming Data." In: *WAPA*. 2011, pp. 48–55 (cit. on p. 20).