Dynamic Implicit Social Recommendation

Qin Zhang

Faculty of Engineering and Information Technology

University of Technology Sydney

A thesis submitted for the degree of

Doctor of Philosophy

January 2018

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

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.

Signature of Student

Production Note:
Signature removed prior to publication.

Acknowledgements

I would like to express my earnest thanks to my supervisor, Professor Chengqi Zhang, and co-supervisors, Doctor Peng Zhang, Doctor Guodong Long and Doctor Jing Jiang. They have provided tremendous support and guidance for my research and my life in the past three and half years.

I would like to express my sincere appreciation to Professor Chengqi Zhang, for his wise guidance and supervisions during my Ph.D study. Professor Chengqi Zhang provided me an opportunity to study in the Center for Artificial Intelligence (which used to be the Center for Quantum Computation and Intelligent Systems), where I met and learnt a lot from many smart and sharp people. I benefited significantly from his unselfish help and valuable suggestion on my research career.

I would like to thank Doctor Peng Zhang who encouraged me staying in academia after I got my Master degree from the University of Chinese Academy of Sciences. Doctor Peng Zhang got me start to love research and taught me how to write an academic paper step by step with his endless patience. His great passion on research also impacted me so much and made me longing to learn new technologies. I could remember the time we discuss the research details, mainly because his broad knowledge background often motivated me to target some interesting fields, which were always exciting experiences.

I also would like to thank Doctor Guodong Long, who gave me a different point of view on how to observe the world, how to know more about myself and how to plan my career. The conversations with him always benefited me so much and gave me a new perspective

on the problems. His vision, creativeness and enthusiasm in solving challenging problems both in research and in projects have greatly encouraged me and inspired my works. I also want to give thanks to him as a friend of mine, for giving me invaluable instructions and suggestions during my life.

I would like to thank Doctor Jing Jiang who gave me a lot of help even before I came to UTS. From her, I learnt a lot on time and energy management. I benefited so much from various discussions and communications with her. She gave me advices, encouragement on the difficult problems in research and in life.

I also wish to express my sincere appreciation to Professor Wei Wang for all the support and help during my visiting in University of California, Los Angeles (UCLA). The guidance and support from her and her team benefited me so much. Her persistent enthusiasm on research and life impressed me so much. I learnt a lot from her, not only on research, but also on how to explore the real-would problem for great ideas, on how to keep firm and tenacious on what we love, and on how to manage well a big team.

I would like to give special thanks to Professor Ivor Tsang. I was so fortunate to spend time and cooperated with him. Discussing a problem and writing a paper with him have been always a pleasure and eye-opening experience. He gave me sufficient freedom and encouragement to think and explore my research interests.

I would also like to thank all the people that had a positive influence on my day-to-day enjoyment of the job: Jia Wu, Shuirui Pan, Haishuai Wang, Maoying Qiao, Tongliang Liu, Zhibin Hong, Shaoli Huang, Sujuan Hou, Zhiguo Long, Zhe Xu, Bozhong Liu, Bo Du, Daokun Zhang, Chaoyue Wang, Shan Xue, Xiaolin Zhang, Ruizhi Zhou, Yue Ma, Li Gao, Guoqing Xu, Stephanie Li, Grace Lin, Christina Tong, FuChien Tseng, Paulina Rivera, Teigan Valenzuela. They are the ones who have given me support during both joyful and stressful times, to whom I will always be thankful. Without their endless patience,

generous support, and constant guidance, this thesis could not have been accomplished.

Finally, and above all, I want to thank my family for their continuous support. I especially thank my parents for their unconditional encouragement and support, both emotionally and financially. No words could possibly express my deepest gratitude for their endless love, self-sacrifice and unwavering help. To them I dedicate this dissertation.

Publications

In Journals

1. **Qin Zhang**, Peng Zhang, Guodong Long, Wei Ding, Chengqi Zhang and Xindong Wu. Online learning from trapezoidal data streams. *IEEE Transactions on Knowledge and Data Engineering (**TKDE**)*, vol. 28(10), pp. 2709-2723, 2016. (Australia ERA Ranked **A**, Core ranked **A**)

2. **Qin Zhang**, Jia Wu, Peng Zhang, Guodong Long and Chengqi Zhang. Discriminative subsequence mining for time series clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence (**TPAMI**)*, **Under third-round review**, 2018. (Australia ERA Ranked **A***, Core ranked **A***)

3. Qinzhe Zhang, Jia Wu, **Qin Zhang**, Peng Zhang, Guodong Long and Chengqi Zhang. Dual influence embedded social recommendation. *World Wide Web Journal (**WWWJ**)*, pp. 1-26, 2017. (Australia ERA Ranked **A**, Core ranked **A**)

In Conferences

4. **Qin Zhang**, Jia Wu, Hong Yang, Yingjie Tian and Chengqi Zhang. Unsupervised feature learning from time series. *26rd International Joint Conference on Artificial Intelligence (**IJCAI**)*, pp. 2322-2328, 2016. (Australia ERA Ranked **A**, Core ranked **A***)

5. **Qin Zhang**, Jia Wu, Peng Zhang, Guodong Long, Ivor W. Tsang and Chengqi Zhang. Inferring latent network from cascade data for dynamic social recommendation. *International Conference on Data Mining (**ICDM**)*, pp. 669-678, 2016. (Australia ERA Ranked **A**, Core ranked **A***)

6. **Qin Zhang**, Peng Zhang, Guodong Long, Wei Ding, Chengqi Zhang and Xindong Wu. Towards mining trapezoidal data streams. *International Conference on Data Mining (**ICDM**)*, pp. 1111-1116, 2015. (Australia ERA Ranked **A**, Core ranked **A***)

7. Qinzhe Zhang, **Qin Zhang**, Guodong Long, Peng Zhang and Chengqi Zhang. Exploring heterogeneous product networks for discovering collective marketing hyping behavior. *Pacific-Asia Conference on Knowledge Discovery and Data Mining (**PAKDD**)*, pp. 40-51, 2016. (Australia ERA Ranked **A**, Core ranked **A**)

8. Li Gao, Yao Lu, **Qin Zhang**, Weixue Lu and Yue Hu. Query expansion for exploratory search with subtopic discovery in community question answering. *International Joint Conference on Neural Networks **IJCNN***, pp. 4715-4720, 2016. (Australia ERA Ranked **A**, Core ranked **A**)

Abstract

Due to the potential value of social relations [125] [34], social recommendation has attracted a lot of attention recently in the research communities.

Modelling time drifting data is a central problem in this area. Temporal changes, such as the emergence of new products or services and the changes in users' purchase behaviour pattern, bring unique challenges. The need for modelling time changes at the level of each individual significantly reduces the amount of available data for detecting such changes. Thus we should resort to more accurate techniques that suffice the dynamic nature of social recommendations.

Besides, the essence of social recommendation methods is utilizing users explicit social connections to improve recommendation results. However, explicit social connection information is not always available in real-world recommender systems. Only few Web sites have implemented the social or trust mechanisms, like Epinions and Douban. Lacking social recommendation data greatly limits the impact and utilization of social recommendation methods. Fortunately, in case that we do not have explicit social information, we can always compute a set of implicit social information to improve the recommendation performance.

To this end, we propose a new Implicit Social Recommendation(ISR) model from cascade data (a variant of time series) in Chapter 3, which makes recommendations based on the inferred latent social network. It can sufficiently mine the information contained in time by mining the cascade data and identify the dynamic changes in the users in time by using the latest updated social network to make recommendations.

Further, since the temporal information is crucial in social recommendation, we explore the temporal behaviour pattern from users' time series data in Chapter 4. The main challenge is finding the discriminative and explainable features (shapelets) that can best represent the raw time series data. We build an economical shapelet learning model that can automatically-learn shapelets without labels. Specifically, a novel Unsupervised Discriminative Subsequence Mining (UDSM) model that automatically-learn shapelets from unlabelled time series data.

Another challenge in social recommendation is the continually appeared new users and new items. In real-world application, this problem is unavoidable. How to involve this kind of trapezoidal data streams and how to handle this problem smoothly must be considered. Most of the existing research either builds a general model in a sample scenario such as ignoring the dynamic nature of the system and the problem of new users and items, or only focuses on some specific fields, such as dynamic restaurant recommendation and timely news recommendation. We propose a new Online Learning with Streaming Features (OLSF) algorithm and its two variants OLSF-I and OLSF-II for mining trapezoidal data streams in Chapter 5. OLSF and its variants combine online learning and streaming feature (i.e. continuously appeared new items) together to handle the double-streaming data.

Finally, we introduce a novel General Online Dynamic Social Recommendation(GODSR) model in Chapter 6, which combines network inference from cascade data, double-streaming users and items, and collaborative filtering together in an iterative process. By inferring the latent dynamic core social networks from cascade data, identifying the drift of a users preferences and involving new users and items in the online learning process, GODSR method reacts rapidly and makes accurate recommendations to users when new data arrive.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Data mining is a fundamental task and has drawn increasing interest in the last decade, due to the ever increasing of gigantic data and the demand for discovering useful information from large scale data. Traditional data mining algorithm performs on data represented by a flat table with instance-feature format. However, due to the rapid development of electronic devices, networking, and social media technologies, recent years have witnessed an increasing number of applications where data is no longer represented in simple instance-feature format, but exhibit as complex network structures indicating dependency relationships. Due to the potential value of social relations [125] [34], social recommendation has attracted a lot of attention recently in the research communities of information retrieval [147], machine learning [172] and data mining [99]. We have witnessed the many popular commercial social recommender systems such as Movielens [51], Epinions and Douban [81].

Modeling time drifting data is a central problem in recommender system. Often, data is changing over time, and up to date modeling should be continuously updated to reflect its present nature. The analysis of such data needs to find the right balance between discounting temporary effects that have very low impact on future behavior, while capturing longer-term trends that reflect the inherent nature of the data. This led to many works on the problem, which is also widely

known as concept drift [113] [134].

Modeling temporal changes in customer preferences brings unique challenges. One kind of concept drift in this setup is the emergence of new products or services that change the focus of customers. Related to this are seasonal changes, or specific holidays, which lead to characteristic shopping patterns. All those changes influence the whole population, and are within the realm of traditional studies on concept drift. However, many of the changes in user behavior are driven by localized factors. For example, a change in the family structure can drastically change shopping patterns. Likewise, individuals gradually change their taste in movies and music. All those changes cannot be captured by methods that seek a global concept drift. Instead, for each customer we are looking at different types of concept drifts, each occurs at a distinct time frame and is driven towards a different direction.

The need to model time changes at the level of each individual significantly reduces the amount of available data for detecting such changes. Thus we should resort to more accurate techniques than those that suffice for modeling global changes. For example, it would no longer be adequate to abandon or simply underweight far in time user transactions. The signal that can be extracted from those past actions might be invaluable for understanding the customer herself or be indirectly useful to modeling other customers. Yet, we need to distill long term patterns while discounting transient noise. This requires a more sensitive methodology for addressing drifting customer preferences. It would not be adequate to concentrate on identifying and modeling just what is relevant to the present or the near future. Instead, we require an accurate modeling of each point in the past, which will allow us to distinguish between persistent signal that should be captured and noise that should be isolated from the longer term parts of the model.

Modeling user preferences is relevant to multiple applications ranging from spam filtering to market-basket analysis. Our main focus in the thesis is on modeling user preferences for building a recommender system, but we believe that general lessons that we learn would apply to other applications as well. Automated recommendations is a very active research field [12]. Such systems analyze patterns of user interest in items or products to provide personalized

recommendations of items that will suit a user's taste. We expect user preferences to change over time. This may stem from multiple factors, some are fundamental while others are more circumstantial. For example, in a movie recommender system, users may change their preferred genre or adopt a new viewpoint on an actor or director. In addition, they may alter the appearance of their feedback. E.g., in a system where users provide star ratings to products, a user that used to indicate a neutral preference by a "3 stars" input, may now indicate dissatisfaction by the same "3 stars" feedback. Similarly, it is known that user feedback is influenced by anchoring, where current ratings should be taken as relative to other ratings given at the same short period. Finally, in many instances systems cannot separate different household members accessing the same account, even though each member has a different taste and deserves a separate model. This creates a de facto multifaceted meta-user associated with the account. A way to get some distinction between different persons is by assuming that time-adjacent accesses are being done by the same member (sometimes on behalf of other members), which can be naturally captured by a temporal model that assumes a drifting nature of a customer.

All these patterns and the likes should have made temporal modeling a predominant factor in building recommender systems. Nonetheless, with very few exceptions, the recommenders literature does not address temporal changes in user behavior. Perhaps, because user behavior is composed of many different concept drifts, all acting in a different timeframe and different directions, thus making common methodologies for dealing with concept drift and temporal data less successful at this setup. We are showing that capturing time drifting patterns in user behavior is essential to improving accuracy of recommenders. This also gives us hope that the insights from successful time modeling for recommenders will be useful in other data mining applications.

On the other hand, the essence of social recommendation methods is utilizing users explicit social connections to improve recommendation results. However, explicit social connection information is not always available in real-world recommender systems. Only few Web sites have implemented the social or trust mechanisms, like Epinions (http://www.epinions.com, a general consumer review site that was established in 1999, where users can also add other users into their

trust list) and Douban (http://www.douban.com, the largest Chinese Web 2.0 site devoted for movies, books, and music reviews that was launched in 2005). Lacking of social recommendation data greatly limits the impact and utilization of social recommendation methods. Fortunately, in case that we do not have explicit social information, we can always compute a set of implicit social information for each user. In summary, we would like to build generalized dynamic social recommendation models and algorithms to explore the dynamics, the implicit social networks and the typically influence groups in the users to further improve the recommendation performance.

1.2 Motivations and Significances

1.2.1 Implicit networks in social recommendation

The researches on social recommendation can be divided into two types: explicit and implicit social recommendation. The existing and available social network information is often used to enhance the performance of a recommender system, i.e. explicit social recommendation [85] [63] [64][84]. The most successful and common strategy is to integrate social information, either trust or friendship, into a collaborative filtering model in a certain way.

However, social information data may not necessarily be available for every recommendation scenario due to practical difficulties or privacy concerns. For example, Taobao, the most popular online shopping platform in China which would be greatly improved by a social recommender system, has not built a social network module for its users. On the other hand, most of the signals that a user provides about his preferences are implicit, such as watching a video or clicking on a link.

Furthermore, for most applications with disclosed social relationships, data are usually given as a binary decision value(e.g., whether two people are friends), while the strength of their relationship is missing. Knowing the strength of social relationships is very helpful for a recommender system, as it is reasonable to assume people have more trust in their close friends compared to their acquaintances. In addition, the quality of the given social information is sometimes

questionable. Since most social data are collected from the web or social network services, inevitably they contain noise. Although it is generally believed that trust or friendship are positively correlated with the level of common-taste of people, the work in [3] shows that two users may not have similar rating tastes even though they strongly trust each other. Thus, an absolute acceptance of the given social connections can harm recommendation performance.

These concerns emerge another research direction named *implicit social recommendation*, which aims at mining implicit user social relationships from historical rating data for better recommendations. Implicit social recommendation can be further divided into two types, one is to determine the social connection strength of the existing binary social network [35] [36] to enhance the quality of recommendation based on the given rating data. The other is data to generate an implicit social network from given historical ratings without any explicit social data [46] [77] [81] [85]. The pseudo links and/or their strengths can then act as a surrogate of the explicit social network [39] [161] to be incorporated into any explicit social recommendation model.

However, most of the existing implicit social recommendation research ignore the dynamic nature of the users. They only use the rating data to study the implicit social relationships, and ignore the dynamic information propagation. The dynamic nature of users' preferences means that they may drift over time in dynamic recommendation, resulting in users having different preferences for particular items at different times [126] [164]. For example, if the user gets married or moves to another city, his preferences are likely to change dramatically, and it is natural for him to turn to new friends for advice rather than old friends. A recommender system should take a user's actions into account instantaneously and adapt recommendations to the user's most recent preference. In fact, the fresher the feedback, the more informative it is on the user's current preference. Thus it is much desired that the dynamic social information can be effectively utilized to capture the dynamic drift of the users' preferences and give exact recommendations in time, and finally improve the recommendation performances.

To this end, Implicit Social Recommendation (ISR) should be proposed to capture the most recent preferences of the users and give exact recommendations.

On the other hand, same as the traditional collaborative filtering methods

which adopt batch-trained algorithms, IDSR suffers from two major drawbacks [79]. Firstly, they scale poorly. The nature of many collaborative filtering algorithms make them hard to be parallelized and recently there are a few works try to investigate this approach. Before training, they require that all data are available. During training, at each iteration, all ratings must be scanned through once to perform the algorithms. This is very expensive since real-world datasets cannot be loaded into the memory easily. The second drawback of batch-trained algorithms is that they are unsuitable for dynamic ratings. A recommender system may change in many ways, such as the appearance of the new users and new items; new purchase and new reviews of the users. In these cases, to capture the change, the batch-trained collaborative filtering methods have to rebuild the model, which is very expensive. Considering these actual situation, we try to overcome the two main drawbacks of IDSR method with an online learning approach. Thus, we first introduce an online learning method which is suitable for the doubly-streaming data, then we apply it to the IDSR method.

1.2.2 Temporal Pattern learning from User behaviours

Based on the inferred latent social network, we want to further explore the salient behaviour pattern of the users from their behaviour timeline. It can be formulated as a problem of significant feature learning from unsupervised time series. The main challenge is finding the discriminative and explainable features that best distinguish the time series from others [5]. To solve this challenge, a line of work that extract *shapelets* from time series [93] [155] has recently been proposed. Shapelets are maximally discriminative subsequences in a time series that enjoy the merits of high prediction accuracy and are easy to explain [150]. Hence, discovering shapelets has become one research focus in time series data analytics.

The seminal work on shapelet discovery [150] demands a full-scan of all possible time series segments, which are then ranked according to a predefined distance metric, such as information gain. The segments that best predict the class labels are selected as shapelets. This method suffers from high time complexity and much effort has been expended to speed-up the algorithm [90] [102] [16]. However, all these methods still face the difficulty of a large number of time series

segments as candidates to choose from [45]. For instance, the Synthetic Control data set [66] contains 600 samples of time series and each is 60 in length, which means the number of candidate segments for all lengths is 1.098×10^6! Such a massive number brings a huge challenge to shapelet selection approaches.

A recent work, [45], proposed a brand new perspective on shapelet discovery in time series. Instead of searching for shapelets in a candidate pool, they borrowed from the strengths of regression learning and opened a new door for *learning shapelets from time series*. The beauty of their method is that the shapelets are detached from the candidate segments, and the learned shapelets differ from any candidate segment [4]. More importantly, in addition to the efficiency, shapelets are also robust to noise [130].

However, the above shapelet learning approach requires supervised learning, and has a labelling cost problem when applied to a large data set. This motivates us to build a more-economical shapelet learning model that can automatically-learn shapelets without the effort of human labelling.

To this end, a new unsupervised discriminative subsequence mining method should be proposed to further explore the typical features in the time series data which is the typical influence path between the users.

1.2.3 Dynamics in doubly-streaming Data

For the real-world social recommendation problem, the data normally appear in a doubly-streaming scenario where both data volume and data dimensions increase with time, since there will be new users enter the market and new features of users will continuously appear. This scenario also suits for many other problems, such as in graph node classification, both the number of graph nodes and the node features (e.g., the ego-network structure of a social network node) often change dynamically. In text classification and clustering, both the number of documents and text vocabulary increase over time, such as the *infinite vocabulary topic model* [157] to allow the addition, invention and increased prominence of new terms to be captured.

We refer to the above doubly-streaming data as *trapezoidal data streams* where data dynamically change in both volume and feature dimension. The problem of

learning from trapezoidal data streams is much more difficult than existing data stream mining and online learning problems [159, 160]. The main challenge of learning from trapezoidal data streams is how to design highly dynamic classifiers that can learn from increasing training data with an expanding feature space. Obviously, existing online learning [68, 70], online feature selection [132] and streaming feature selection algorithms [136] cannot be directly used to handle the problem because they are not designed to deal with the simultaneous change of data volume and data dimension.

Online learning algorithms [70] were proposed to solve the problem where training instances arrive one by one but the feature space is fixed and known a prior before learning. The algorithms update classifiers using incoming instances and allow the sum of training loss gradually to be bounded [70]. To date, online learning algorithms, such as the Perceptron algorithm [107], the Passive Aggressive algorithm [19] and the Confidence-Weighted algorithm [20], are commonly used in data-driven optimizations, but cannot be directly used to handle a dynamic feature space.

Online feature selection algorithms [70, 132] were proposed to perform feature selection in data streams where data arrive sequentially with a fixed feature space. Online feature selectors are only allowed to maintain a small number of active features for learning [132]. These algorithms use sparse strategies, such as feature truncation, to select representative features. Sparse online learning via truncated gradient [70] and the OFS algorithm [132] are typical algorithms. However, these algorithms cannot solve the trapezoidal data stream mining problem because they assume the feature space is fixed.

Online streaming feature selection algorithms [136] were proposed to select features in a dynamic feature space where features arrive continuously as streams. Each new feature is processed upon its arrival and the goal is to select a "best so far" set of features to train an efficient learning model. It, in some ways, can be seen as the dual problem of online learning [136]. Typical algorithms include the online streaming feature selection (OSFS) algorithm [137] and the fast-OSFS [136] algorithm. However, these algorithms consider only a fixed training set where the number of training instances is given in advance before learning.

To this end, an online learning algorithm towards on streaming features is

needed to explore the special kind of double-streaming data, and be further introduced to the social recommendations.

1.2.4 All-in-One: Online dynamic implicit social recommendation

Ultimately, we would like to build a general and dynamic social recommendation algorithm which can make timely and accurate personalized recommendations. The quicker the recommender system responds, the greater the likelihood that it will identify the user's current preferences and needs. For instance, after a user books a flight, hotels and places of interest in the destination city should be recommended to the user in a timely fashion. Also many recommendations should be given to users in advance due to capacity and budget issues, such as hotel recommendations which should be provided several days or months in advance so as to allow the user sufficient time to choose and book a hotel in a lower price. However there are still three main challenges in this area.

The first one is the drift of a user's preferences and needs over time. A recommender system needs to capture users' dynamic preferences and needs rapidly and deliver the right recommendations at the right time[162]. For instance, if the user has recently become pregnant, her preferences are likely to change dramatically to baby-related goods such as toys, buggies and car seats etc.

The second challenge is how to obtain the latent dynamic core social networks of users. A core social network is different from the general explicit social network, as it only contains the very close friends of the users who truly influence him. However, this is difficult to obtain since it is dynamic and implicit. For instance, supposing a user has recently moved to a new city, it is natural for him to ask for recommendations from his friends in the new city rather than other people. But this change is difficult to reflect in the whole large explicit social network which contains a large amount of redundant information and noise. If we consider the whole explicit social network, it will divert much attention from the network of real influence and harm the final performance. On the other side, most disclosed social data are normally binary decision values, e.g. whether two people are friends or not, so the strength of their relationship is missing [35]. Also

many signals of user's preferences are implicit [36], such as watching a video or clicking on a link, like *Google News* [23]. The complete core social network with weights is important for a social recommendation model and would improve the recommendations significantly.

The last challenge is how to deal with new users and new items which appear in the system. In reality, this problem is unavoidable. New users and new items enter the recommender system continuously, so how to involve them and how to handle this problem smoothly must be considered.

However, to the best of our knowledge, currently there is no research that tries to address these challenges simultaneously. Most of the existing research either builds a general model in a sample scenario such as ignoring the dynamic nature of the system and the problem of new users and items, or only focuses on some specific fields, such as dynamic restaurant recommendation [17] and timely news recommendation [23].

To this end, combining the three techniques introduced above, a generalized online dynamic social recommendation model is proposed to address these challenges simultaneously and effectively.

1.3 Research Problems

As each subtask is essential in real-life applications, our research examines the unique challenge of each task carefully and exploits several research problems accordingly.

1.3.1 Implicit social recommendation with cascade data

Implicit Social Recommendation should be proposed to capture the most recent preferences of the users and give exact personalized recommendations. In this problem, we use cascade data to infer latent social network and attempts to identify the dynamic changes in the users' preferences to improve the performance of recommender system. Cascade data are defined as the observed action time stamps of users on certain items. We follow the idea of the generative probabilistic model to infer both the structure and the strength of the latent social network.

By jointly learning the social relationships and ratings from the data, our model can absorb the potential damage caused by a noisy explicit social network. By exploring the information in time data, we capture the most recent preferences of the users and give exact recommendations.

1.3.2 Unsupervised user behaviour pattern learning

User influence path learning, which is as same as unsupervised discriminative subsequence mining, should be proposed to further explore the influence path among the users, i.e. typical features in the time series.

To this end, we propose a novel **U**nsupervised **D**iscriminative **S**ubsequence **M**ining (UDSM) model that automatically-learn shapelets from unlabelled time series data. We first introduce a *pseudo-label* to transform unsupervised learning into supervised learning. Then, we use the popular *regularized least-squares technique* along with *spectral analysis* to learn both the shapelets, the pseudo-class labels, and the pseudo classification boundaries. A new shapelet regularization term is also added to avoid learning similar shapelets. We adopt a coordinate descent algorithm to simultaneously obtain the pseudo-class labels and learn the best shapelets in an iterative manner. Compared to existing unsupervised shapelet selection models [155] [93], our method aims to learn shapelets instead of simply selecting shapelets from unlabelled time series data.

1.3.3 Online learning with streaming features

Online Learning with trapezoidal data which are in double-streaming scenario is needed to deal with the now appeared users and items. The new coming data is a popular problem in social recommendation. The algorithms would combine online learning and streaming feature selection to continuously learn from trapezoidal data streams. Specifically, when new training instances carrying new features arrive, a classifier updates existing features by following the passive-aggressive update rule used in online learning and updates the new features by following the structural risk minimization principle. Then, feature sparsity is introduced by using feature projected truncation.

1.3.4 Generalized online dynamic implicit social recommendation

Generalized Online Dynamic Social Recommendation is needed to address the complex real recommendation challenges: the drift of a user's preferences over time; the inference of the latent dynamic core social networks of users; the appearance of new users and new items. Specifically, we combine network inference, online learning and collaborative filtering together in an iterative process. By inferring the latent dynamic core social networks from cascade data, identifying the drift of a user's preferences and involving new users and items in the online learning process, it reacts rapidly and makes accurate recommendations to users when new data arrive.

1.4 Thesis Contributions

The thesis addresses a number of fundamental problems of *dynamic social recommendation* from four aspects: implicit social recommendation; discriminative feature learning from time series; online learning with trapezoidal data streams and generalized online dynamic social recommendation. The main contributions of this study can be summarized in four parts, accordingly.

1.4.1 Implicit social recommendation with cascade data

Contributions:

1) We study a new problem of implicit social recommendation, which infers the latent social network from cascade data and uses rating data to make recommendations based on the inferred latent social network. It can sufficiently mine the dynamic information propagation and identify the dynamic changes of users' preferences in time.

2) We propose a new Implicit Social Recommendation (ISR) model to combine the learning of social network and rating prediction together as an unified optimization problem, which is different from most of the existing approaches of implicit social network recommendations which treat the

learning of the social networks and recommendations as two independent tasks.

3) We use two new real-world datasets (Zomato and Douban movie data) we collected, and a public MovieLens dataset to evaluate the performance of the proposed model. Experiments show that the proposed model outperforms the state-of-the-art solutions in both explicit and implicit social recommendation scenarios on these three real-world datasets.

Outcome:

- The implicit social recommendation algorithm was published in ICDM-2016 [162].

1.4.2 Feature learning from unsupervised time series

Contributions:

1) UDSM, the proposed new unsupervised discriminative subsequence mining model, extends supervised shapelet learning models to unlabelled time series data by combining the strengths of pseudo-class labels, spectral analysis, shapelet regularization and regularized least-squares techniques.

2) We empirically validate the performance of the proposed method on a synthetic data set and 36 real-world data sets. The results show promising performance compared to the state-of-the-art unsupervised time series learning models.

Outcome:

- The feature learning algorithm for time series was published in IJCAI-2016 [161].

1.4.3 Online learning with trapezoidal data streams

Contributions:

1) We study a new problem of learning from trapezoidal data streams where training data change in both data volume and feature space;

2) We propose a new learning algorithm OL_{SF} and its two variants. OL_{SF} combines the merits of online learning and streaming feature selection methods to learn from doubly-streaming data;

3) We theoretically analyze the performance bounds of the proposed algorithms;

4) We empirically validate the performance of the algorithms extensively on 14 real-world data sets.

Outcome:

- Our online learning algorithm for trapezoidal data was published in ICDM-2015 [163], and we further explored it, and published a paper in TKDE-2016 [164].

1.4.4 Online dynamic implicit social recommendation

Contributions:

1) We study a new problem of online dynamic social recommendation with latent core social networks, which addresses the concerns of the continuous drift of a user's preferences and needs, the dynamic core social network of users and the problem of newly appearing users and items to make timely and accurate recommendations.

2) We propose a new General Online Dynamic Social Recommendation method which combines network inference, online learning and the collaborative filtering techniques together in an iterative process to make timely and accurate recommendation. It addresses the three aforementioned concerns simultaneously and effectively, which is different with most current researchers that tried to design the methods in a defined specific scenario.

3) We use two real-world datasets (Zomato and Douban movie data) we crawled, and the public MovieLens dataset to evaluate the performance of the pro-

posed model. Experiments show that the proposed model outperforms the state-of-the-art solutions on these three real-world datasets.

Outcome:

- Our generalized online dynamic social recommendation algorithm will be submitted to *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)* in February 2018.

1.5 Thesis Overview

The thesis is structured into four parts: implicit social recommendation with cascade data; user influence path learning; online learning with trapezoidal data streams and generalized online dynamic social recommendation. Table 1.1 gives the overall structure of our research with the chapters of this thesis. The detailed roadmap of the thesis is summarized as follows:

Table 1.1: Structure of the thesis with references to the chapters.

Research Task	Chapter	Related publication
Introduction & Literature Review	Chapter 1, 2	– –
Implicit social recommendation with cascade data	Chapter 3	[162]
Feature learning from time series	Chapter 4	[161]
Online learning with trapezoidal data streams	Chapter 5	[163] [164]
Online dynamic social recommendation	Chapter 6	– –
Conclusion and Future works	Chapter 7	– –

Before formally studying the sub-problems, we review all related works from different aspects in Chapter 2, including implicit social recommendation with cascade data, online learning with trapezoidal data streams, feature learning from time series and generalized online dynamic social recommendation.

Chapter 3 presents the algorithm on implicit social recommendation, which infers the latent social network from cascade data and uses rating data to make recommendations based on the inferred latent social network. It can sufficiently mine the dynamic information propagation and identify the dynamic changes of users' preferences in time. It combines the learning of social network and rating prediction together as an unified optimization problem, which is different from most of the existing approaches of implicit social network recommendations which treat the learning of the social networks and recommendations as two independent tasks.

Chapter 4 depicts a new unsupervised discriminative subsequence mining model which extends supervised shapelet learning models to unlabelled time series data by combining the strengths of pseudo-class labels, spectral analysis, shapelet regularization and regularized least-squares techniques.

Chapter 5 describes a new online learning algorithm for trapezoidal data streams and its two variants to deal with the new problem of learning from trapezoidal data streams where training data change in both data volume and feature space. It combines the merits of online learning and streaming feature selection methods to learn from doubly-streaming data.

Chapter 6 studies a new problem of online dynamic social recommendation with latent core social networks, which addresses the concerns of the continuous drift of a user's preferences and needs, the dynamic core social network of users and the problem of newly appearing users and items to make timely and accurate recommendations. We propose a new General Online Dynamic Social Recommendation method which combines network inference, online learning and the collaborative filtering techniques together in an iterative process to make timely and accurate recommendation. It addresses the three aforementioned concerns simultaneously and effectively, which is different with most current researchers that tried to design the methods in a defined specific scenario.

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

Chapter 2

Literature review

This thesis studies several fundamental problems for dynamic social recommendation. As a result, this work is closely related to a variety of works from different aspects: explicit and implicit social recommendation, online learning with trapezoidal data, feature learning from time series and online dynamic social recommendation. In this chapter, we review the related work accordingly.

2.1 Implicit social recommendation

In this section, we review several major approaches to social recommendations. Firstly we briefly introduce the principle of social recommendation, then we introduce the recent research on the two main categories: explicit social recommendation which uses external social networks and implicit social recommendation which uses the social networks inferred from the rating data.

2.1.1 Traditional recommender systems

Social recommendation [84] is defined as any recommendation with online social relations as additional input, i.e., augmenting an existing recommendation engine with additional social signals [123]. Social relations can be trust relations, friendships, memberships or following relations and so on [85] [61]. The underlying assumption is that users are correlated when they establish social relations [84] [88].

One of the most commonly-used and successfully-deployed recommendation approaches is collaborative filtering [85] [61]. It can predict user interests directly by uncovering complex and unexpected patterns from a user's past behaviors such as product ratings without any domain knowledge. The underlying assumption of collaborative filtering based recommender systems is that if users have agreed with each other in the past, they are more likely to agree with each other in the future than to agree with randomly chosen users. The general idea of matrix factorization is to model the user-item interactions with factors representing the latent characteristics of the users and items in the system, like the preference class of users and the category class of items [145]. Numerous social matrix factorization based RSs have recently been proposed to improve recommendation accuracy [147] [84] [146]. In the field of collaborative filtering, two types of methods are widely studied: memory-based approaches and model-based approaches [123].

Memory based approaches use either the whole user-item matrix or a sample to generate a prediction. The key problems a memory-based collaborative filtering method has to solve are computing similarity and aggregating ratings [123]. Model based methods assume a model to generate the ratings and apply data mining and machine learning techniques to find patterns from training data [123], which can be used to make predictions for unknown ratings. Compared to memory based collaborative filtering methods, Model based collaborative filtering methods have a more holistic goal to uncover latent factors that explain observed ratings [123] [152]. Factorization based collaborative filtering methods [110] are representative method in this field, which assume that a few latent patterns influence user rating behaviors and perform a low-rank matrix factorization on the user-item rating matrix. Collaborative filtering based recommender systems are competitive for they are domain-independent and can recommend any items. However, collaborative filtering based systems have their own limitations such as the cold-start problem and the data sparsity problem [123].

2.1.2 Explicit social recommenders

Explicit social recommender uses rating data and all the information from external social networks. Early research [63] [88] searches the trust network to determine the recommended items. Later on, researchers started to bring social information into matrix factorization models with diversified assumptions for integration. SoRec [84] proposes shared user latent factors for both rating matrix and social matrix factorization. RSTE [82] predicts a user's ratings by the linear combination of the user and their trusted friends' latent factors vectors. SocialMF [64] defines that a user's latent factors should be close to the linear combination of his or her trusted friends' latent factors. Social Regularization [85] considers a pairwise assumption that two users who trust each other should have similar latent factors, and thus appends a regularization term to the classical matrix factorization model. Moreover, social influence is also considered in social recommendation, such as conditional random field [139] and probabilistic Poisson factorization [14]. TrustSVD [48] incorporates trust networks with SVD++, a variant of matrix factorization modelling implicit influence from user latent factors through observed ratings.

2.1.3 Implicit social recommenders

Implicit social recommender attempts to extract latent social correlation between two users from historical rating behaviors. The generated information serves as the surrogate for explicit social networks in explicit social recommender systems. Since it is time-consuming to evaluate the quadratic number of pairwise users or item social relations, several studies assume that an explicit social network is available but the explicit edge strength information is missing. Fazeli et al. [36] survey and compare the performance of different trust strength metrics on an explicit social network combined with SocialMF. Fang et al. [35] use support vector regression with matrix factorization to learn both the ratings and strengths from an explicit trust network. However, in order to train this model, a binary social network is still required. Despite the quadratic time complexity of evaluating relations, Guo et al. [46] study user-based collaborative filtering that recommends items using a trust network generated from predefined trust metrics. With the

existence of extra features, Lin et al. [77] (rating time) and Guo et al. [47] (text review) present methods to obtain implicit social networks. There are also several studies that apply matrix factorization techniques on implicit social networks [83]. And Social Regularization [81] reads implicit social networks generated by the evaluation of cosine similarity and Pearson correlation respectively together with predefined thresholds to determine the social connections.

In some cases, the existing explicit and implicit social recommendation cannot achieve appropriate performance, because they ignore the dynamic changes of uses' preferences. To address this issue, an implicit dynamic social recommendation is proposed in Chapter 3 to learn the structure and the dynamic strength of social connection through cascade data simultaneously, which is more general because implicit signals such as clicks can be used to obtain the cascade data. Furthermore, by modelling the rating prediction and social strength learning as a joint optimization task, the proposed model mutually reinforces the quality of each to achieve better results.

2.2 Feature learning from unsupervised time series

2.2.1 Shapelets selection

The works, [150] [101], introduce selecting short time series segments that best predict class labels. The central idea of shapelet discovery is to score all the segments of some given training data against how predictive they are with respect to the given class labels [135] [33]. The seminal work in the study of time series shapelets, [150], builds a decision-tree classifier by recursively searching for informative shapelets over distances determined by information gain. In addition to information gain, several new measures, such as Kruskall-Wallis, Mood's median and F-Stat are also used in shapelet selection [56] [78].

Since time series data usually have a large number of candidate subsequences, using brute force to select shapelets results in infeasible runtimes [45] [161]. Therefore, a series of techniques have been proposed to speed up these methods.

Some are smart implementations that use entropy pruning on the information gain heuristic and abandon distance computations early [150]. Others prune the search space and re-use computations [90] [151]; prune candidates by searching for potentially interesting candidates in a SAX representation [102]; or use infrequent shapelets [52] [50].

2.2.2 Shapelet learning

Instead of searching shapelets exhaustively, a recent work, [45], proposes learning optimal shapelets, and reports statistically significant improvements in accuracy compared to other shapelet-based classifiers. Instead of restricting the pool of possible candidates to those found in the training data and simply searching them, they consider shapelets as parameters that can be learned through regression learning. This type of learning method does not consider a limited set of candidates, but rather it can choose from arbitrary shapelets.

2.2.3 Unsupervised seature selection

A series of unsupervised feature selection methods are proposed to select discriminative features from unlabelled data [28] [106]. A traditional criterion in unsupervised feature learning is to select the features that best preserve data similarity, and construct the manifold structure from the original feature space [54] [168] [10]. However these methods fail to incorporate important information implied within data, and which cannot be directly applied to our shapelet learning problem. Earlier unsupervised feature selection methods evaluated the importance of features individually, and selected the best features one by one [54] [168], on the assumption that correlation among different features would be neglected [10] [169].

State-of-the-art unsupervised feature selection algorithms select features by simultaneously exploiting discriminative information and feature correlation [149] [76] [115] [73]. Unsupervised discriminative feature selection [149] selects the most informative features for representing data by considering its manifold structure. Since the most discriminative information in feature selection is normally encoded in the labels, it is popular to predict good cluster indicators as pseudo-class labels for unsupervised feature selection [76] [75] [74]. Another important

factor that significantly effects the performance of feature selection is outliers and noise data [91]. Real-world data is normally not distributed in an ideal fashion: outliers and noise in the data are common place [44] [138]. Thus it is important and necessary to make unsupervised feature selection robust to outliers and noise [91] [98].

2.2.4 Shapelets for clustering

Shapelets also have been used to cluster time series [155] [156] [2]. Jesin et al. [155] proposed a method that uses unsupervised shapelets (u-Shapelets) for time series clustering. The algorithm selects u-Shapelets to separate and remove a subset of the time series from the rest of the data set. It then iteratively repeats the search process in the remaining data until no data remains. This is a greedy search algorithm that attempts to maximize the gap between the two groups of time series divided by a u-Shapelet [129].

The k-shape algorithm, proposed in [93], clusters time series. K-shape is a novel algorithm for shape-based time series clustering. It is efficient and domain independent [72]. It is a scalable iterative refinement procedure that creates homogeneous and well-separated clusters. Specifically, k-Shape uses a distance measure which is invariant to scaling and shifting [24]. It adopts a normalized cross-correlation measure as a distance measure, unlike [37], to consider the shape of each time series. Based on a normalized cross-correlation, the method computes cluster centroids in each iteration and updates the assignment of the time series to the clusters.

Our work in Chapter 4 differs from the above research streams in that we introduce *unsupervised shapelet learning* to auto-learn shapelets from an unlabelled time series by combining the strengths of shapelet learning and unsupervised feature selection.

2.3 Online learning from trapezoidal data streams

The third part of our researches, i.e. online learning from trapezoidal data streams, are also closely related to online learning, online feature selection and

online streaming feature selection.

2.3.1 Traditional online learning

Traditional online learning represents an important family of efficient and scalable data mining and machine learning algorithms for massive data analysis [60] [96]. In general, online learning algorithms can be grouped into two categories, the first-order and second-order learning algorithms [60].

The first-order online learning algorithms exploit first order information during update. The Perceptron algorithm [107] [38] and Online Gradient Descent algorithm (OGD) [173] are two well-known first-order online learning methods. Moreover, a large number of first-order online learning algorithms have been proposed recently by following the criterion of maximum margin principle [132], such as the Passive Aggressive algorithms (PA) [19], Approximate Maximal Margin Classification algorithm (ALMA) [40], and the Relaxed Online Maximum Margin algorithms (ROMMA) [40].

The second-order online learning algorithms, which can better explore the underlying structure between features[60], have been explored recently. Most second-order learning algorithms assume that the weight vector follows a Gaussian distribution. The model parameters, including both the mean vector and the covariance matrix, are updated in the online learning process [60]. The Second-Order Perceptron (SOP) [12], Normal Herding method via Gaussian Herding (NHERD) [22], Confidence-Weighted (CW) learning, Soft Confidence Weighted algorithm(SCW) [20], online learning algorithms by Improved Ellipsoid (IELLIP) [143], and Adaptive Regularization of Weight Vectors (AROW) [21], New variant of Adaptive Regularization (NAROW) [92] are representative of the second-order online learning algorithms.

2.3.2 Traditional feature selection

Traditional feature selection is a widely used technique for reducing dimensionality. Feature selection aims to select a small subset of features minimizing redundancy and maximizing relevance to the class label in classification. Feature

selection can be categorized into supervised [133] [118], unsupervised [32] [89] and semi-supervised [140] [166] algorithms.

Supervised feature selection can be categorized into the filter models, wrapper models and embedded models [122]. The filter models separate feature selection from classifier learning so that the bias of a learning algorithm does not interact with the bias of a feature selection algorithm. The Relief [117], Fisher score [30] and Information Gain based methods [95] [141] are the representative algorithms. The wrapper models use the predictive accuracy of a predetermined learning algorithm to determine the quality of selected features. The embedded methods [6] [1] [171] aim to integrate feature selection into model training. It achieves model fitting and feature selection simultaneously [100] [121]. The embedded methods are usually the fastest methods.

Unsupervised feature selection attempts to select features that preserve the original data similarity or manifold structures, and it is difficult to evaluate the relevance of features [31] [122]. Laplacian Score [53], spectral feature selection [167], and recently proposed $l_{2,1}$-norm regularized discriminative feature selection [148] are representatives of unsupervised feature selection. Semi-supervised feature selection is between the supervised methods and unsupervised methods. Under the assumption that labelled and unlabelled data are sampled from the same population generated by the target concept, semi-supervised feature selection uses both labelled and unlabelled data to estimate feature relevance [166].

2.3.3 Online feature selection

Online feature selection [132] and *sparse online learning* [29] [70] aim to learn a sparse linear classifier from a sequence of high-dimensional training instances. Online feature selection combines feature selection with online learning and resolves the feature selection in an online fashion by developing online classifiers that involve only a small and fixed number of features for classification. OFS and OFS_P [132] are the representative algorithms proposed recently.

2.3.4 Online streaming feature selection

Online streaming feature selection algorithms have been studied recently [41] [136] where features arrive one by one and training instances are available before the training process starts. The number of training instances remains fixed through the process [137]. The goal is to select a subset of features and train an appropriate model at each time step given the features observed so far.

Compared with the above learning methods, the problem studied in Chapter 5 is more challenging because of the doubly streaming data scenario. Existing online learning, online feature selection and online streaming feature selection algorithms are incapable of learning from trapezoidal data streams.

2.4 Online dynamic social recommendation

In this section, we review the recent works in the most closely related areas on online dynamic social recommendation. Firstly, we briefly introduce the offline recommendation methods, and then the recent research on online recommenders in detailed.

2.4.1 Offline recommenders

The wide interest in personalized recommendations has sparked substantial research in this area [17][9]. The most common approaches are content-based approaches [94] and collaborative filtering [23] [110]. There are two types of collaborative filtering methods, which power most modern recommenders: neighborhood-based approaches and model-based approaches [85]. Neighborhood-based methods mainly focus on finding similar users [65] [23] or items [25] [111] for recommendations. Model-based approaches use the observed user-item rating to train a compact model that explains the given data, so that ratings can be predicted via the model instead of directly manipulating the original rating database, as occurs in neighborhood-based approaches [80]. Algorithms in this category include the clustering model [43], the aspect models [58] [116], the latent factor model [59], the Bayesian hierarchical model [165] and the ranking model [80].

2.4.2 Online recommenders

Recently, it has been recognized that offline recommender approaches suffer from the cost of retraining the model, they do not necessarily match online user behavior and fail to capture preference drifts. These realizations have initiated developments towards building continuously learning– online learning recommenders [17]. An online training method based on a compressed representation of the matrix is presented in [119], which is performed with a combination of variational message passing and expectation propagation. The work [23] builds an online recommendation engine for making personalized recommendations on a large web property for *Google News*, which combine recommendations from collaborative filtering using MinHash clustering, Probabilistic Latent Semantic Indexing and visitation counts together using a linear model. The work in [17]develops a preference elicitation framework and an online learning setting to address the cold-start problem in restaurant recommendations by asking a new user a few questions to quickly learn their preferences and taking advantage of the latent structure in the recommendation space using a probabilistic latent factor model. The work in [9] introduces an ϵ-greedy online user-user neighbor-based collaborative filtering method.

However, none of these methods simultaneously consider all the three concerns we have introduced, especially the latent dynamic core social networks and the newly appearing users and items. To this end, we develop our general online dynamic social recommendation algorithm to make timely and accurate personalized recommendation.

Part I

Implicit Social Recommendation with cascade data

Chapter 3

Inferring Latent Network from Cascade Data for Social Recommendation

3.1 Introduction

Social recommendation, a study aimed at incorporating the social information of users into a recommender system, has attracted much attention in recent years. It can further be divided into two types: explicit and implicit social recommendation. The existing and available social network information is often used to enhance the performance of a recommender system, i.e. explicit social recommendation [85] [63] [64][84]. The most successful and common strategy is to integrate social information, either trust or friendship, into a collaborative filtering model in a certain way.

However, social information data may not necessarily be available for every recommendation scenario due to practical difficulties or privacy concerns. For example, Taobao, the most popular online shopping platform in China which would be greatly improved by a social recommender system, has not built a social network module for its users. On the other hand, most of the signals that a user provides about his preferences are implicit, such as watching a video or clicking on a link.

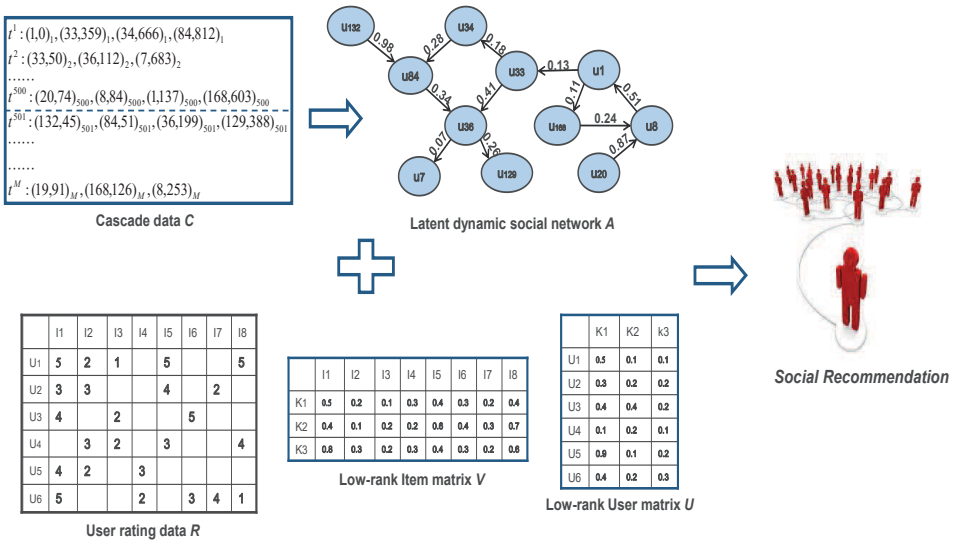


Figure 3.1: An example of Implicit Social Recommendation (ISR) framework. We infer both the structure and the strength of latent dynamic social network A from cascade data C to catch the dynamic changes of the users by using a generative probabilistic model. Based on the inferred latent dynamic social network A, ISR model learns the low-rank item matrix V and user matrix U to predict the missing values in rating data R. By jointly learning the social relationships and missing ratings from both rating and cascade data, ISR can absorb the potential damage brought up by noisy explicit social network and capture the most recent preferences of the users and give the exact recommendations.

Furthermore, for most applications with disclosed social relationships, data are usually given as a binary decision value(e.g., whether two people are friends), while the strength of their relationship is missing. Knowing the strength of social relationships is very helpful for a recommender system, as it is reasonable to assume people have more trust in their close friends compared to their acquaintances. In addition, the quality of the given social information is sometimes questionable. Since most social data are collected from the web or social network services, inevitably they contain noise. Although it is generally believed that trust or friendship are positively correlated with the level of common-taste of people, the work in [3] shows that two users may not have similar rating tastes even though they strongly trust each other. Thus, an absolute acceptance of the given social connections can harm recommendation performance.

These concerns call for another research direction named *implicit social rec-*

ommendation, which aims at mining implicit user social relationships from historical rating data for better recommendations. Implicit social recommendation can be further divided into two types, one is to determine the social connection strength of the existing binary social network [35] [36] to enhance the quality of recommendation based on the given rating data. The other type of implicit social recommendation is to generate an implicit social network from given historical ratings without any explicit social data [46] [77] [81] [85]. The pseudo links and/or their strengths can then act as a surrogate of the explicit social network [39] [161] to be incorporated into any explicit social recommendation model.

However, most of the existing implicit social recommendation research ignore the dynamic nature of the users. They only use the rating data to study the implicit social relationships, and ignore the dynamic information propagation. The dynamic nature of users' preferences means that they may drift over time in dynamic recommendation, resulting in users having different preferences for particular items at different times [126] [164]. For example, if the user gets married or moves to another city, his preferences are likely to change dramatically, and it is natural for him to turn to new friends for advice rather than old friends. A recommender system should take a user's actions into account instantaneously and adapt recommendations to the user's most recent preference. In fact, the fresher the feedback, the more informative it is on the user's current preference. Thus it is much desired that the dynamic social information can be effectively utilized to capture the dynamic drift of the users' preferences and give exact recommendations in time, and finally improve the recommendation performances.

To this end, in this chapter, we propose a new **Implicit Social Recommendation (ISR)** model which uses cascade data to infer latent social network and attempts to identify the dynamic changes in the users' preferences to improve the performance of recommender system. We define cascade data as the observed action time stamps of users on certain items. We follow the idea of the generative probabilistic model to infer both the structure and the strength of the latent social network. By jointly learning the social relationships and ratings from the data, our model can absorb the potential damage caused by a noisy explicit social network. By exploring the information in time data, we capture the most recent preferences of the users and give exact recommendations. Figure 3.1 shows an

example of the ISR framework. Experiments show that the proposed solution outperforms the state-of-the-art models in both explicit and implicit scenarios. The **contributions** of this chapter are summarized as follows:

1) We study a new problem of implicit social recommendation, which infers the latent social network from cascade data and uses rating data to make recommendations based on the inferred latent social network. It can sufficiently mine the dynamic information propagation and identify the dynamic changes of users' preferences in time.
2) We propose a new Implicit Social Recommendation (ISR) model to combine the learning of social network and rating prediction together as an unified optimization problem, which is different from most of the existing approaches of implicit social network recommendations which treat the learning of the social networks and recommendations as two independent tasks.
3) We use two new real-world datasets (Zomato and Douban movie data) we collected, and a public MovieLens dataset to evaluate the performance of the proposed model. Experiments show that the proposed model outperforms the state-of-the-art solutions in both explicit and implicit social recommendation scenarios on these three real-world datasets.

The rest of this chapter is organized as follows: Sections 3.2 and 3.3 introduce the problem and ISR model in detail. Section 3.4 discusses the proposed algorithm. Section 3.5 describes the experiments and Section 3.6 concludes the paper.

3.2 Problem definition

The rating data is denoted by matrix $R \in \mathbb{R}^{N \times M}$, where N is the number of users and M is the number of items. An observed or non-missing entry R_{ij}, records a numerical rating score that user $i, 1 \leq i \leq N$, gives to item $j, 1 \leq j \leq M$, as a training instance. An adjacent matrix $A \in \mathbb{R}^{N \times N}$ denotes the social network,

where entry A_{ij} is a positive value which represents the strength of the edge from node i to node j.

Time observations are also recorded on the N users and consist of a set C of cascades $\{t^1, \ldots, t^M\}$. Each cascade t^c is a record of observed action time stamps within the population during a time interval of length T. A cascade is an N-dimensional vector $t^c := (t_1^c, \ldots, t_N^c)$ recording when nodes' actions are observed, $t_k^c \in [0, T] \cup \{\infty\}$. The symbol ∞ labels users that are not observed acting during observation window $[0, T]$, which does not imply the nodes will never act. Each cascade is reset to start at 0 as we are not concerned about the specific time but the relative time. Lengthening the observation window T increases the number of observed infections within a cascade t^c and results in a more representative sample of the underlying dynamics [105].

Our goal is to mine the implicit social network A from cascade data C, and predict the missing value of the original rating matrix R simultaneously by solving an optimization problem.

3.3 The proposed ISR model

The formula of the **Implicit Social Recommendation** (ISR) model, which consists of dynamic social network inferring term and social recommendation regularization term, is shown in Eq. (3.1).

$$\begin{aligned} \min_{A,U,V} & -\frac{1}{2}\sum_{c\in C} log f(t^c, A) + \frac{\lambda_1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}\|U_i - U_j\|^2 \\ & + \frac{\lambda_2}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_3}{2}\|U\|_F^2 + \frac{\lambda_4}{2}\|V\|_F^2 \\ s.t. \quad & A \geq 0, U \geq 0, V \geq 0 \end{aligned} \tag{3.1}$$

where I_{ij} is the entry of indicator matrix I. I_{ij} is equal to 1 if user u_i rates item v_j and equals to 0 otherwise. Matrix $U \in \mathbb{R}^{K\times N}$ and $V \in \mathbb{R}^{K\times M}$ with $K << min(M, N)$ are two low-rank matrices to approximate the rating matrix R. Matrix $A \in \mathbb{R}^{N\times N}$ is the inferred social network and the entry A_{ij} represents the strength of the influence from user i to user j. N is the number of users,

M is the number of items as previously introduced. $t^c \in C$ is a N-dimensional cascade vector. $\lambda_1, \lambda_2, \lambda_3$ and λ_4 are positive parameters to control the weight of each term.

Indeed, the model in Eq. (3.1) mainly comprises three terms. The first term $-\frac{1}{2}\sum_{c\in C} logf(t^c, A)$ is the dynamic social network inferring. The second term $\frac{\lambda_1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}\|U_i - U_j\|^2$ is the social recommendation regularization, and the third term $\frac{\lambda_2}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(R_{ij} - U_i^T V_j)^2$ is the standard matrix factorization based recommender system, with the last two terms $\frac{\lambda_3}{2}\|U\|_F^2$ and $\frac{\lambda_4}{2}\|V\|_F^2$ been added to avoid overfitting. Our model conducts the learning of social network and rating prediction together as a unified framework. By using the dynamic social network inferring term, the proposed model can infer the latent social network from cascade data which sufficiently mine the dynamic information propagation and catch the latest dynamic changes of the users in time to improve the performance of recommendation.

3.3.1 Dynamic social network inference

In this section, we introduce the method of inferring dynamic social network from cascade data. We first introduce the pairwise transmission likelihood, then we give the formula of likelihood of a cascade, and finally show the social network inferring term.

• Pairwise transmission likelihood. With the cascades data, we calculate the pairwise transmission likelihood as follow. We assume that infections can occur at different rates over different edges of a network, and aim to infer the transmission rates between pairs of nodes in the network. Define $f(t_i|t_j, A_{j,i})$ as the conditional likelihood of transmission between a node j and node i. The transmission likelihood depends on the infection times (t_j, t_i) and a pairwise transmission rate $A_{j,i}$. A node cannot be infected by a node infected later in time. In this chapter, we estimate the well-known exponential parametric likelihood model (3.2)

$$f(t_i|t_j; A_{j,i}) = \begin{cases} A_{j,i} \cdot e^{-A_{j,i}(t_i - t_j)}, & \text{if } t_j < t_i \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

where $A_{j,i} \geq 0$ is transmission rate. If $A_{j,i} \to 0$, the likelihood of infection tends to zero and the expected transmission time becomes arbitrarily long.

The cumulative density function $F(t_i|t_j; A_{j,i})$ is computed from the transmission likelihoods [112]. Given that node j is infected at time t_j, the survival function of edge $j \to i$ is the probability that node i is not infected by node j by time t_i:

$$S(t_i|t_j; A_{j,i}) = 1 - F(t_i|t_j; A_{j,i}) = e^{-A_{j,i}(t_i - t_j)} \tag{3.3}$$

The hazard function [112][105], or instantaneous infection rate, of edge $j \to i$ is the ratio

$$H(t_i|t_j; A_{j,i}) = \frac{f(t_i|t_j; A_{j,i})}{S(t_i|t_j; A_{j,i})} = A_{j,i} \tag{3.4}$$

• Likelihood of a cascade. We compute the probability that a node survives uninfected until time T, given that some of its parents are already infected. Consider a cascade $t = (t_1, \ldots, t_N)$ and a node i, which is not infected during the observation window (i.e., $t_i > T$). Since each infected node k may infect i independently, the probability that nodes $1 \ldots N$ do not infect node i by time T is the product of the survival functions of the infected nodes $1 \ldots N | t_k \leq T$ targeting i,

$$\prod_{t_k \leq T} S(T|t_k; A_{k,i}) \tag{3.5}$$

Since we assume infections are conditionally independent given the parents of the infected nodes, the likelihood factorization over nodes is

$$f(t^{\leq T}; A) = \prod_{t_i \leq T} f(t_i|t_1, \ldots, t_N \backslash t_i; A) \tag{3.6}$$

where $t_{\leq T} = (t_1 \ldots, t_N | t_i \leq T)$. Given an infected node i, we compute the likelihood of a potential parent j to be the first parent by applying Eq. (3.5),

$$f(t_i|t_j; A_{j,i}) \times \prod_{j \neq k, t_k < t_i} S(t_i|t_k; A_{k,i}) \tag{3.7}$$

With Eq. (3.6), the likelihood of the infections in a cascade is shown in

Eq. (3.8) by removing the condition $k \neq j$,

$$f(t^{\leq T}; A) = \prod_{t_i \leq T} \prod_{k:t_k<t_i} S(t_i|t_k; A_{k,i}) \times \sum_{j:t_j<t_i} \frac{f(t_i|t_j; A_{j,i})}{S(t_i|t_j; A_{j,i})} \tag{3.8}$$

Eq. (3.8) only considers infected nodes. We add the multiplicative survival term from Eq. (3.5) to include the information of the nodes which are not infected during the observation window and also replace the ratios in Eq. (3.8) with hazard functions,

$$\begin{aligned} f(t; A) = &\prod_{t_i \leq T} \prod_{t_m > T} S(T|t_i; A_{i,m}) \\ &\times \prod_{k:t_k<t_i} S(t_i|t_k; A_{k,i}) \sum_{j:t_j<t_i} H(t_i, t_j; A_{j,i}) \end{aligned} \tag{3.9}$$

Assuming independent cascades, the likelihood of a set of cascades $C = \{t^1, \ldots, t^M\}$ is the product of the likelihoods of the individual cascades in Eq. (3.10)

$$\prod_{t^c \in C} f(t^c; A) \tag{3.10}$$

• Social Network inferring. We aim to find the strength $A_{j,i}$ of every pair of nodes such that the likelihood of an observed set of cascades $C = \{t^1, \ldots, t^M\}$ is maximized. Thus, we can solve the following optimization problem (3.11) to obtain the latent social network from the given cascade data,

$$\begin{aligned} \min_{A} \quad & -\sum_{c \in C} log f(t^c; A) \\ s.t. \quad & A_{j,i} \geq 0, i, j = 1, \ldots, N, i \neq j \end{aligned} \tag{3.11}$$

where $A := \{A_{i,j} | i, j = 1, \ldots, n, i \neq j\}$ are the variables. The edges of the network are those pairs of nodes with transmission rates $A_{ij} > 0$.

3.3.2 Social recommendation regularization term

In the real world, it is usual to turn to friends for movie , music or book recommendations since we have confidence in the tastes of our friends. Due to the assumption that the closer the relationship between the users, the more similar

preferences they have, we set the social regularization term in Eq. (3.12)

$$\sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij}\|U_i - U_j\|^2 \tag{3.12}$$

where N is the number of users, $A_{ij}, 1 \leq i \leq N, 1 \leq j \leq N$ is the inferred implicit social network matrix.

3.3.3 Low-rank matrix factorization.

An efficient and effective approach to recommender systems is to factorize the user-item rating matrix, and utilize the factorized user-specific and item-specific matrices to make further missing data prediction [85] [104] [109] [110] [154]. The premise behind a low-dimensional factor model is that there is only a small number of factors influencing the preferences, and that a user's preference vector is determined by how each factor applies to that user[104].

In this chapter, we consider an $N \times M$ rating matrix R describing N users' numerical ratings on M items. A low-rank matrix factorization approach seeks to approximate the rating matrix R by a multiplication of K-rank factors, where $K << min(N, M)$, and the singular value decomposition method is traditionally utilized [85] to approximate a rating matrix R by minimizing $\frac{1}{2}\|R - U^T V\|_F^2$ where $U \in \mathbb{R}^{K\times N}, V \in \mathbb{R}^{K\times M}$, and $\|\cdot\|_F^2$ denotes the Frobenius norm. However, due to a large number of missing values contained in R, we only need to factorize the observed ratings in it. In order to avoid overfitting, two regularization terms are added. Hence, objective function is changed to

$$\min_{U,V} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\alpha}{2}\|U\|_F^2 + \frac{\beta}{2}\|V\|_F^2 \tag{3.13}$$

where $\alpha, \beta > 0$, and I_{ij} is the indicator function that is equal to 1 if user U_i rated item V_j and equal to 0 otherwise.

The optimization problem in Eq. (3.13) minimizes the sum-of-squared-errors objective function with quadratic regularization terms.

According to the above discussions, the implicit social network inferring term,

Algorithm 1 Implicit Social Recommendation

Require:
$C = \{t^1, \ldots, t^M\}$: Cascade data;
$R \in \mathbb{R}^{N \times M}$: Rating data;
$\lambda_1, \lambda_2, \lambda_3$ and λ_4: Tradeoff parameters;
K: Latent feature space dimension parameter.

Ensure:
$U \in \mathbb{R}^{K \times N}$: User matrix;
$V \in \mathbb{R}^{K \times M}$: Item matrix;
$A \in \mathbb{R}^{N \times N}$: Social network matrix.

Initialize: $U_0 = E, V_0 = E, A_0 = E$ where E is the identity matrix;
while Not convergent **do**
  $A_{t+1} \leftarrow$ Apply Eq. (3.16) with U_t to update A_t;
  $U_{t+1} \leftarrow$ Apply Eq. (3.20) with A_{t+1}, V_t to update U_t;
  $V_{t+1} \leftarrow$ Apply Eq. (3.24) with A_{t+1}, U_{t+1} to update V_t;
  $t \leftarrow t + 1$;
return $A^* = A_t$; $U^* = U_t$, $V^* = V_t$.

the social recommendation regularization term and the traditional matrix factorization-based recommender system term, are unified in our ISR model for implicit social recommendation.

3.4 Algorithm

In this section, we use a coordinate descent algorithm to solve the ISR model. We first introduce the algorithm in Section 3.4.1, and analyze its convergence and time complexity in Section 3.4.2 and Section 3.4.3 respectively.

3.4.1 Implicit social recommendation algorithm

In the coordinate descent algorithm, we iteratively update one variable by fixing the remaining two variables. The steps will be repeated until convergence. Algorithm 1 summarizes the steps.

- **update A with fixed U and V**

With fixed U, V, model (3.1) degrades to problem (3.14)

$$\min_{A\geq 0} \ -\frac{1}{2}\sum_{c\in C} logf(t^c, A) + \frac{\lambda_1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{i,j}\|U_i - U_j\|^2 \tag{3.14}$$

where $f(t^c, A)$ is shown in Eq. (3.9) with t replaced by t^c. Problem (3.14) is convex and consistent [105]. For the fixed i and j, problem (3.14) degrades to problem (3.15)

$$\begin{aligned}\min_{A\geq 0} \ & \frac{1}{2}\sum_{c:t_i^c\leq T, t_j^c > T} (T - t_i^c)A_{ij} + \frac{\lambda_1}{2}\|U_i - U_j\|^2 A_{ij} \\ & + \frac{1}{2}\sum_{c:t_i^c < t_j^c, t_j^c \leq T} ((t_j^c - t_i^c)A_{ij} - logA_{ij})\end{aligned} \tag{3.15}$$

To set the derivative of A_{ij} equal to 0, we can obtain the update strategy of A_{ij}^{t+1} as Eq. (3.16)

$$A_{ij}^* = \frac{2N_{ij}^{\tilde{C}}}{\sum_{c\in\hat{C}_{ij}}(T - t_i^c) + \sum_{c\in\tilde{C}_{ij}}(t_j^c - t_i^c) + \lambda_1\|U_i - U_j\|^2} \tag{3.16}$$

where $\hat{C}_{ij} = \{c \in C : t_i^c \leq T, t_j^c > T\}$, $\tilde{C}_{ij} = \{c \in C : t_i^c < t_j^c, t_j^c \leq T\}$ and $N_{ij}^{\tilde{C}}$ is the number of the cascades in $\tilde{C}_{ij}$ for the fixed i, j.

- **update U with fixed A and V**

With fixed A, V, model (3.1) degrades to problem (3.17)

$$\begin{aligned}\min_{U\geq 0} \ & \frac{\lambda_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i V_j^\top)^2 + \frac{\lambda_3}{2}\|U\|_F^2 \\ & + \frac{\lambda_1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{i,j}\|U_i - U_j\|^2\end{aligned} \tag{3.17}$$

Problem (3.17) is convex and consistent with variable U. It is equal to prob-

lem (3.18)

$$\begin{aligned}\min_{U\geq 0}\ & \frac{\lambda_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij}-\sum_{p=1}^{K}U_{ip}V_{jp})^2+\frac{\lambda_3}{2}\sum_{i=1}^{N}\sum_{p=1}^{K}U_{ip}^2 \\ &+\frac{\lambda_1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{K}A_{ij}(U_{ip}-U_{jp})^2\end{aligned} \tag{3.18}$$

For the fixed i and p, the derivative of U_{ip} is in Eq. (3.19)

$$\begin{aligned}\frac{\nabla F}{\nabla U_{ip}} =& -\lambda_2\sum_{j=1}^{M} I_{ij}V_{jp}(R_{ij}-\sum_{q=1}^{K}U_{iq}V_{jq})+\lambda_3 U_{ip} \\ &+\lambda_1\sum_{j=1}^{N}A_{ij}(U_{ip}-U_{jp})\end{aligned} \tag{3.19}$$

Make the derivative equal to zero, we can obtain

$$U_{ip}^{*}=\frac{\lambda_2\sum_{j=1}^{M} I_{ij}V_{jp}(R_{ij}-\sum_{q=1,q\neq p}^{K}U_{iq}V_{jq})+\lambda_1\sum_{j=1}^{N}A_{ij}U_{jk}}{\lambda_2\sum_{j=1}^{M} I_{ij}V_{jp}^2+\lambda_3+\lambda_1\sum_{j=1}^{N}A_{ij}} \tag{3.20}$$

- **update V with fixed A and U**

With fixed A, U, model (3.1) degrades to problem (3.21)

$$\min_{V\geq 0}\ \frac{\lambda_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij}-U_iV_j^{\top})^2+\frac{\lambda_4}{2}\|V\|_F^2 \tag{3.21}$$

Problem (3.21) is equal to problem (3.22)

$$\min_{V\geq 0}\frac{\lambda_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij}-\sum_{p=1}^{K}U_{ip}V_{jp})^2+\frac{\lambda_4}{2}\sum_{j=1}^{M}\sum_{p=1}^{K}V_{jp}^2 \tag{3.22}$$

For fixed j and p, the derivative of V_{jp} is

$$\frac{\nabla F}{\nabla V_{jp}}=-\lambda_2\sum_{i=1}^{N} I_{ij}U_{ip}(R_{ij}-\sum_{q=1}^{K}U_{iq}V_{jq})+\lambda_4 V_{jp} \tag{3.23}$$

Make the derivative equal to zero, we can obtain

$$V_{jp}^{*} = \frac{\lambda_2 \sum_{i=1}^{N} I_{ij} U_{ip} (R_{ij} - \sum_{q=1, q \neq p}^{K} U_{iq} V_{jq})}{\lambda_4 + \lambda_2 \sum_{i=1}^{N} I_{ij} U_{ip}^2} \tag{3.24}$$

3.4.2 Convergence analysis

Algorithm 1 is an typical coordinate descent algorithm, which will convergent to the local optimums. When updating A, U and V, in each step, the degraded optimization problem, which only contain one variable, is a convex optimization. We can obtain the temporary optimum in each step, which would make the objective function decrease persistently until convergent to one of its local optimums.

3.4.3 Complexity analysis

In Algorithm 1, ISR takes constant time for the initialization (line 1). When update social network matrix A, suppose the maximum number of cascades, in which $t_i < T$ contained, is I_c, it takes $O(I_c N^2)$ (line 3). When update matrix U, since rating matrix R is very sparse, it takes $O(\epsilon M + N)$ for each entry, where ϵ is the sparse rate. Thus, this part takes $O(\epsilon MNK + N^2 K)$ (line 4). When update matrix V, it takes $O(\epsilon N)$ for each entry, so the actual time expense of line 5 is $O(\epsilon NMK)$. To sum up, the total worst-case time complexity of Algorithm 1 is $O(\epsilon MNKC + N^2 KC + N^2 I_c C)$ where C is the maximum number of iterations until convergence.

3.5 Experiments

In the following, we first introduce the details of the datasets and metrics we use in Section 3.5.1 and Section 3.5.2 respectively. We then introduce the experimental setup and benchmark methods. Finally, we describe the comparison in detail with three aspects. Firstly, we evaluate the performance of the proposed algorithm with respect to MAEs and RMSEs compared with the five state-of-the-art benchmark methods in Section 3.5.4. Secondly, we explore the performance of the ISR method in terms of data freshness (time distances between the target ratings

and historical ratings) in Section 3.5.5, and finally we explore the performance of ISR method in terms of observation window T in Section 3.5.6. All experiments are conducted on a Windows 8 machine with 3.00GHz CPU and 8GB memory.

3.5.1 Datasets

We evaluate the performance of the proposed ISR model on three real-world datasets, *i.e.,* Zomato [1], MovieLens [2] and Douban movie [3] dataset. The statistics of the datasets are show in Table 3.1. The sparsity shown in the table is defined as $1 - \frac{\text{nonzero entries}}{\text{total entries}}$.

Zomato is a restaurant search and discovery service website founded in 2008. It operates in many countries, including the United States, Australia, India etc. It features restaurant information such as scanned menus and photos sourced by local street teams, as well as user reviews and ratings.

We collect the ratings (1-5) data and the time points of the posted reviews of 5336 users for the most popular 1012 restaurants in Sydney from September 2008 to April 2016. We also collect information on the existing social networks provided by the website, and the followers' and followees' data, which will be used in some of the benchmark methods for comparison. This dataset is very sparse with a 0.0031 sparsity value.

MovieLens datasetis a web-based research recommender system that debuted in the autumn of 1997. Each week, hundreds of users visit MovieLens to rate and receive recommendations for movies. MovieLens is a popular dataset used in social recommendation researches. It was collected through the MovieLens website[4] during the seven-month period from September 19th, 1997 to April 22nd, 1998. The dataset consists of 100,000 ratings (1-5) from 943 users on 1682 movies and each user has rated at least 20 movies. This data has been cleansed that users who had less than 20 ratings or did not provide complete demographic information were removed from this data set. Detailed descriptions of the data file can be found in [51]. And the sparsity of this dataset is 0.0630.

[1] https://www.zomato.com
[2] http://www.grouplens.org/node/73
[3] http://movie.douban.com
[4] https://movielens.org/

Table 3.1: Statistics of Datasets

Dataset	Douban	MovieLens	Zomato
# of users	249408	943	5336
# of items	100	1682	1012
# of ratings	877572	100000	38367
sparsity	0.0330	0.0630	0.0031
Average cascade length	8755.70	59.45	38.06
Longest cascade length	29234	583	302
Shortest cascade length	480	1	3
Time interval	2015.1.1 -2016.4.30	1997.9.17 -1998.4.22	2008.9.21 -2016.4.30

Table 3.2: Performance Comparisons with 80% of data as training set

Dataset	Metrics	BaseMF	SocialMF	$SR1_{pcc}$	$SR2_{pcc}$	SR_{i+-}^{u+-}	ISR
Douban	MAE	0.9260	0.8019	0.7946	0.7831	0.7049	**0.6609**
	Improve	28.64%	17.59%	16.83%	15.61%	6.25%	
	RMSE	1.0525	1.0308	1.0257	1.0083	0.9710	**0.9484**
	Improve	9.90%	8.00%	7.54%	5.95%	2.33%	
Movielens	MAE	1.0187	0.8984	0.8814	0.8772	0.7849	**0.7375**
	Improve	27.61%	17.91%	16.33%	15.93%	6.04%	
	RMSE	1.0964	1.0505	1.0587	1.0331	0.9909	**0.9511**
	Improve	13.25%	9.46%	10.16%	7.93%	4.01%	
Zomato	MAE	0.9773	0.8505	0.8537	0.8317	0.7489	**0.7038**
	Improve	27.99%	17.25%	17.56%	15.38%	6.02%	
	RMSE	1.1027	1.0558	1.0333	1.0249	0.9870	**0.9319**
	Improve	15.49%	11.73%	9.81%	9.07%	5.58%	

Douban is one of the most websites in China. It comprises several parts: Douban Movie, Douban Read and Douban Music, etc. Douban Movie provides the latest movie information, and users can record the movies they wish to watch and rate them after they have watched them. They can also share their reviews with their friends.

Table 3.3: Performance Comparisons with 60% of data as training set

Dataset	Metrics	BaseMF	SocialMF	$SR1_{pcc}$	$SR2_{pcc}$	SR_{i+-}^{u+-}	ISR
Douban	MAE	0.9469	0.8618	0.8622	0.8497	0.7637	**0.7046**
	Improve	25.59%	18.24%	18.28%	17.08%	7.74%	
	RMSE	1.1059	1.0893	1.0414	1.0336	1.0641	**0.9500**
	Improve	14.09%	12.78%	8.77%	8.08%	10.72%	
Movielens	MAE	1.0464	0.9386	0.9452	0.9335	0.8981	**0.8284**
	Improve	20.84%	11.75%	12.36%	11.26%	7.77%	
	RMSE	1.1962	1.1660	1.1536	1.1316	1.1297	**1.0067**
	Improve	15.84%	13.66%	12.73%	11.03%	10.88%	
Zomato	MAE	0.9958	0.9117	0.9292	0.8948	0.8150	**0.7402**
	Improve	25.67%	18.81%	20.34%	17.28%	9.18%	
	RMSE	1.1136	1.1019	1.1042	1.0934	1.0562	**0.9934**
	Improve	10.79%	9.84%	10.03%	9.14%	5.95%	

We collect the ratings and the time stamps of the rating data on the 100 most popular movies in Douban. It includes 877572 ratings (1-5) and the time stamps of 249408 users from January 2015 to April 2016. In this dataset, the data of time stamps are very dense, where the average length of cascades is 8755.7 with the longest one is 29234. The sparsity of this dataset is 0.0330.

3.5.2 Metrics

Recommender systems research has used several types of measures to evaluate the quality of recommender systems. Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are the most popular metrics used to evaluate the deviation of recommendations from their true user-specified values. Specifically, RMSE and MAE are defined in Eqs. (3.25) and (3.26) respectively.

$$RMSE = \sqrt{\frac{1}{N_{test}} \sum_{i,j} (R_{ij} - \hat{R}_{ij})^2} \tag{3.25}$$

$$MAE = \frac{1}{N_{test}} \sum_{i,j} |R_{ij} - \hat{R}_{ij}| \tag{3.26}$$

where R_{ij} denotes the rating user i gives to item j, $\hat{R}_{ij}$ denotes the rating user i gives to item j as predicted by the method, and N_{test} denotes the number of ratings in test set. From the definitions, we can see RMSE gives higher weights to larger errors as the errors are squared before taking their average. It is always larger or equal to MAE. The lower the values of RMSE and MAE, the more accurately the recommendation engine predicts.

3.5.3 Experimental setup

Training and testing data setup In many studies [111] [69], the training and testing data are randomly chosen for the experiments. But this is unsuitable for the evaluation of dynamic recommendation in which we can use only historical data but not future data for current predictions in real applications. Following [71] [126], we split the training and testing data based on time and evaluate the performance. Specifically, we sort the entire data set in normal time order, and use the earlier part as the training set to adjust all parameters in the recommendation algorithm. We run algorithms on the test set then and generate the estimated rating for each user-item pair and compare the estimated ratings with real rating to calculate RMSEs and MAEs.

Parameter settings focus on the meanings and settings of the parameters. We implement the proposed algorithm and compare it with benchmark methods under these parameters. Parameter K controls the dimension of the latent feature space. If K is too small, it is difficult for the model to make a distinction among users or items. If K is too large, users and items will be too unique for the system to calculate their similarities and the complexity will considerably increase. Here we choose the best value of K through grid search from 5 to 50 in steps of 5. The tradeoff parameters $\lambda_1, \lambda_2, \lambda_3$ and λ_4 in Eq. (3.1) control the weights of each part in the model and we choose the best value for them through grid search in $[10^{-8}, 10^{8}]$ with the step of 10^2.

Benchmark methods To evaluate the performance of our method, we com-

pare the proposed ISR with the following five approaches.

- **BaseMF** is the baseline matrix factorization approach proposed in [110], which does not take the social network into account;

- **SocialMF** [64] improves the recommendation accuracy of BaseMF by taking into account the social trust between users. It always uses all social links available in the dataset;

- **SR1**$_{pcc}$ [85] imports average-based regularization in social recommendation regularization term to form the social recommendation model;

- **SR2**$_{pcc}$ [85] is another social recommendation method where individual-based regularization is imported as its social recommendation regularization term;

- **SR**$_{i+-}^{u+-}$ [81] is an implicit social network recommendation approach which uses both implicit similar and dissimilar user information as well as similar and dissimilar item information in its recommendation model.

3.5.4 Comparisons

In this section, we compare the recommendation results of the proposed approach ISR with all benchmark methods to evaluate the effectiveness.

For the three datasets, we use different training data settings (80% and 60%) to test the algorithms. Training data 80%, for example, means we use the first 80% of the cascade and rating data as the training data to predict the last 20% of the ratings. The experimental results are shown in Table 3.2 and 3.3. For the MovieLens and Douban datasets, social network data are not provided, so we choose the five most similar users for each user as her/his friends and build the explicit social network which is used in the benchmark methods.

From the results, we can observe that our method consistently outperforms the other approaches in all the settings of the three datasets. Firstly, we can see our algorithm generates significantly better results than the BaseMF method

which does not take the social network into account. There is an average 26.05% improvement in MAEs (max 28.63%, min 20.84%), and an average 13.23% improvement in RMSEs (max15.84%, min 9.90%) on all the three datasets. This observation illustrates that employing implicit user social information helps increase the recommendation quality.

Furthermore, our method performs better than the state-of-the-art explicit social recommendation algorithms SocialMF, SR1_{pcc} and SR2_{pcc} on all the three datasets. Specifically, the proposed ISR achieves 16.93% improvement on average (max 18.81%, min 11.75%) in MAEs and 10.91% improvement on average (max 13.66%, min 8.00%) in RMSEs compared with SocialMF. Compared with the other two explicit social recommendation approaches SR1_{pcc} and SR2_{pcc}, our method achieves on average 16.95% (max 20.34%, min 12.36%) and 15.42% (max 17.28%, min 11.26%) improvement in MAEs respectively and an average 9.84% (max 12.73%, min 7.54%) and 8.54% (max 11.03%, min 5.95) improvement in RMSEs respectively. These observations demonstrate that in many situations, implicit social networks can much more accurately describe a user's preferences compared with explicit social networks and avoid the noise contained in the explicit social networks.

Compared with the implicit social recommendation approach SR_{i+-}^{u+-}, our algorithm also achieves better results an average 7.17% (max 9.18%, min 6.02%) improvement in MAEs and an average 6.58% (max 10.88%, min 2.33%) improvement in RMSEs. This observation shows that our implicit social recommendation model also outperforms the state-of-the-art static implicit social recommendation systems.

3.5.5 Performance in terms of data freshness

In this section, we explore the performance of the proposed model in terms of data freshness. All the three datasets contain abundant rating records and cascade data which are collected over a reasonable period of time and are different in composition and dynamic in nature. We define the time distances between the target ratings and historical ratings as *data freshness*. We manually assign twelve different data freshness values by classify the cascades into multiple phases, i.e.,

within 1 week, within 2 weeks,..., within 12 weeks.

Figure 3.2 shows the MAEs and RMSEs of our ISR using the data in different setting of data freshness on the three datasets. It is clear that the more fresh the data, the better the performance. Such observation means using closer data sources to train recommender system, the difference between the original ratings and the predicted ones becomes smaller.

3.5.6 Performance in terms of observation window

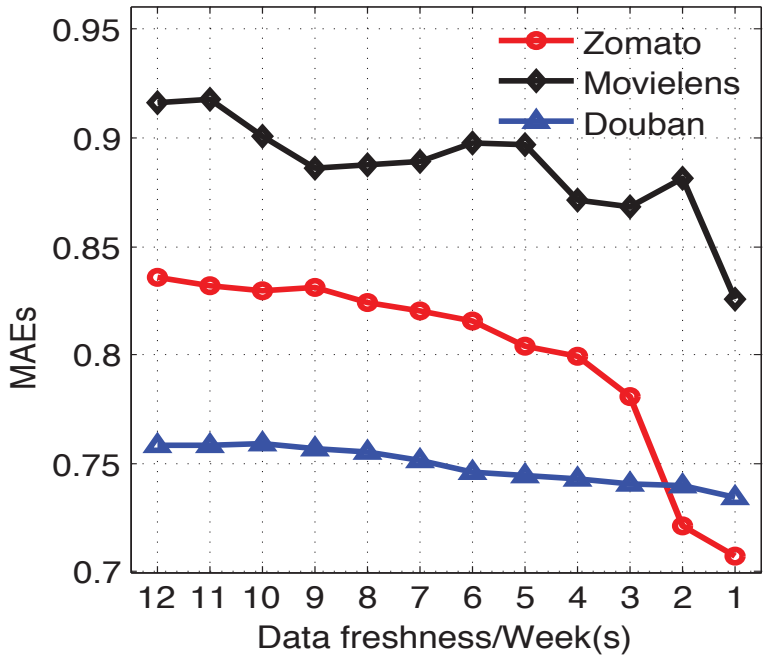
In this section, we explore the performance of the ISR model in terms of observation window T. We use different sized training sets. For MovieLens and Douban datasets, we rebuild four new training datasets based on the original one. In the new training datasets, the rating data were proposed within 1 week, 1 month, 3 months and 6 months respectively while for the Zomato dataset we rebuild four new training datasets in 1 year, 2 years, 3 years and 6 years due to its high sparsity.

Figure 3.3 shows the results of ISR with different observation windows. We can see in most cases, MAEs and RMSEs are large when the observation windows are very short. With longer observation windows, MAEs and RMSEs decrease at first and then increase again, which means the predicted results become better at first and then become worse later. This is because the longer cascades may contain more noise which harms the predicted results. However, if the cascades are too short, they do not contain enough information to infer the whole latent social network, which also harms the algorithm performances.

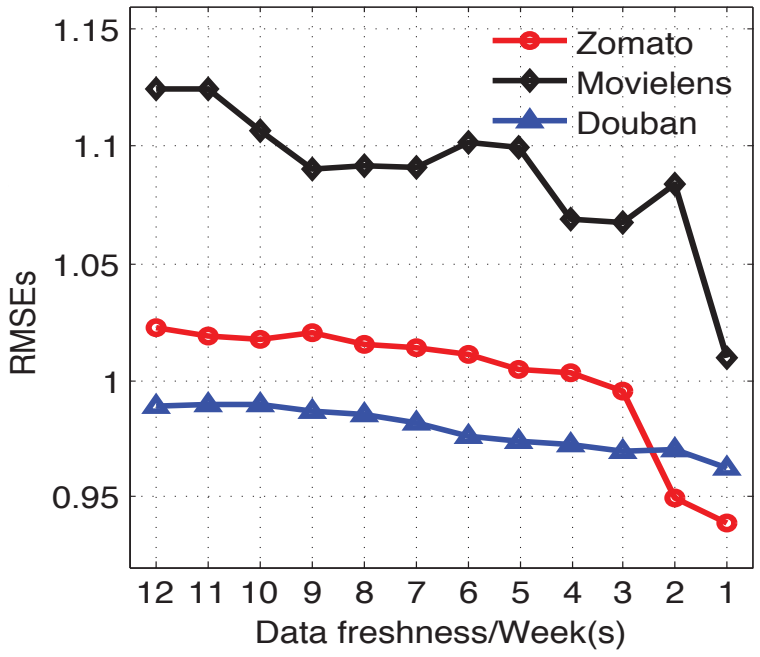
3.6 Conclusion

In this chapter, we formulate a new implicit social recommendation problem where the inferred latent dynamic social network is used for enhancing the social recommendation performance. We proposed a new model ISR which uses cascade data to infer implicit social networks. ISR addresses the commonly existing preference drafting issues in real social recommendation studies, and identifies dynamic changes in the users closely by taking advantage of the information

contained in time. This is different from most of the existing implicit social recommendation approaches which only use rating data to infer implicit social networks. The proposed ISR model also undertakes the learning of social structures and rating predictions together in a unified optimization problem, rather than treating the learning of social networks and recommendations as two independent tasks. The experiments demonstrate that the proposed ISR model outperforms the state-of-the-art models in both explicit and implicit social recommendation scenarios.

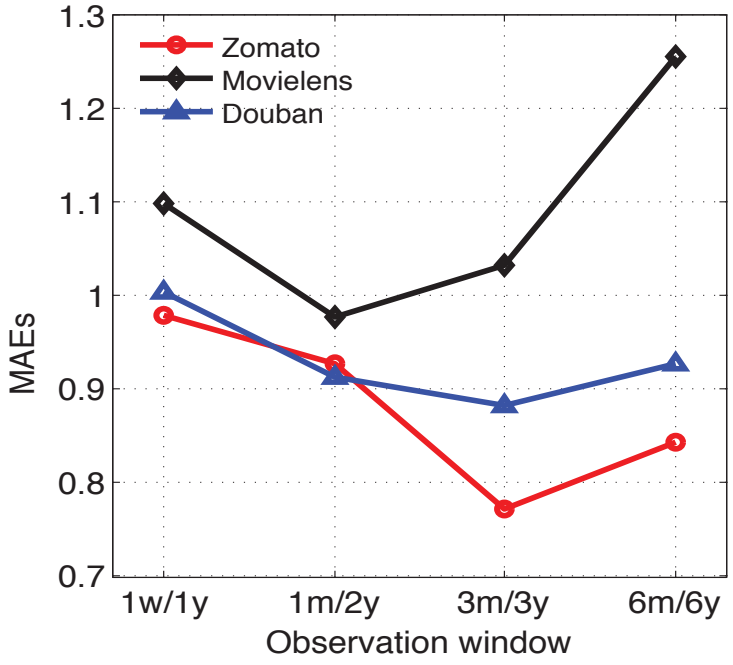


(a) MAE

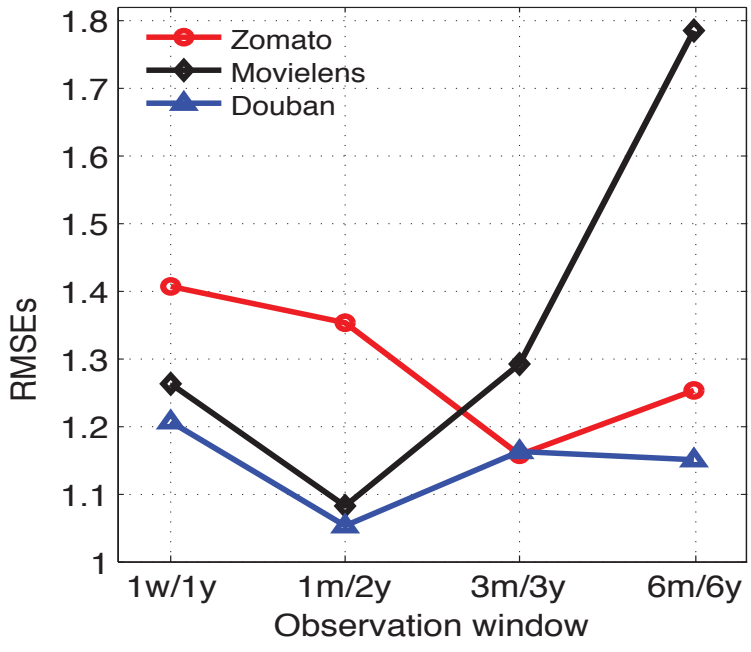


(b) RMSE

Figure 3.2: MAEs and RMSEs obtained by ISR algorithm in terms of different data freshness (time distances between the target ratings and historical ratings) values on the three datasets.



(a) MAE



(b) RMSE

Figure 3.3: MAEs and RMSEs obtained by ISR algorithm in terms of observation window on the three datasets.

Part II

Exploration on users' temporal behaviours

Chapter 4

Discriminative Subsequence Mining for Unsupervised Time Series

4.1 Introduction

We abstract the temporal information of users' behaviors as a set of time series, and learn the most discriminative features from them to explore their behavior patterns.

Time series research has been widely used in real world applications, such as finance [108] [127], medicine [57] [67], trajectory analysis [11] [55] and human action segmentation [42] [170]. The main challenge is finding the discriminative and explainable features that best predict time series class labels [5]. To solve this challenge, a line of work that extract *shapelets* from time series [93] [155] has recently been proposed. Shapelets are maximally discriminative subsequences in a time series that enjoy the merits of high prediction accuracy and are easy to explain [150]. Hence, discovering shapelets has become one research focus in time series data analytics.

The seminal work on shapelet discovery [150] demands a full-scan of all possible time series segments, which are then ranked according to a predefined distance metric, such as information gain. The segments that best predict the class labels

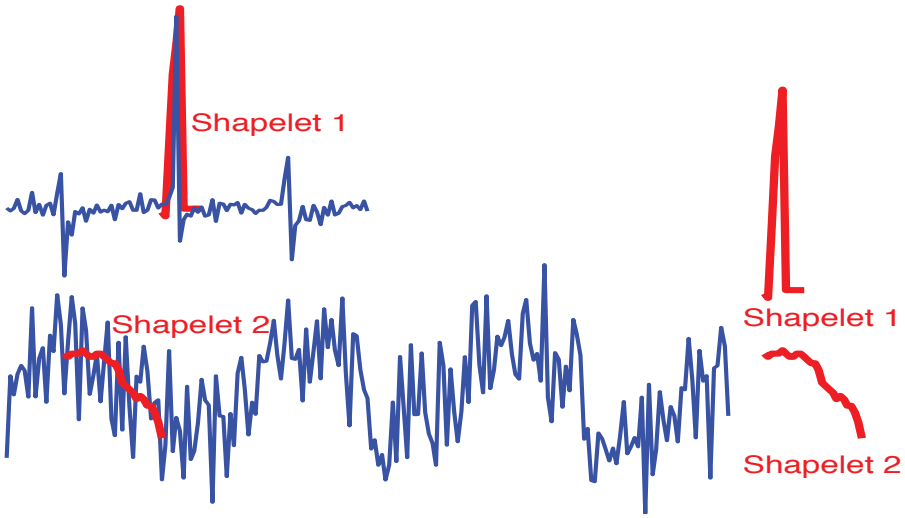


Figure 4.1: The top left blue curve is a rectangular signal with Gaussian noise while the bottom left blue curve is a sinusoidal signal with Gaussian noise. The short curves on the right, marked in bold red, are learned shapelets. We observe that the learned shapelets differ from all the candidate segments and are robust to noise.

are selected as shapelets. This method suffers from high time complexity and much effort has been expended to speed-up the algorithm [90] [102] [16]. However, all these methods still face the difficulty of a large number of time series segments as candidates to choose from [45]. For instance, the Synthetic Control data set [66] contains 600 samples of time series and each is 60 in length, which means the number of candidate segments for all lengths is 1.098×10^6! Such a massive number brings a huge challenge to shapelet selection approaches.

A recent work, [45], proposed a brand new perspective on shapelet discovery in time series. Instead of searching for shapelets in a candidate pool, they borrowed from the strengths of regression learning and opened a new door for *learning shapelets from time series*. The beauty of their method is that the shapelets are detached from the candidate segments, and the learned shapelets differ from any candidate segment [4]. More importantly, in addition to the efficiency, shapelets are also robust to noise [130] as shown in Fig. 4.1.

However, the above shapelet learning approach requires supervised learning, and has a labelling cost problem when applied to a large data set. This motivates us to build a more-economical shapelet learning model that can automatically-

learn shapelets without the effort of human labelling.

In this chapter, we propose a novel **U**nsupervised **D**iscriminative **S**ubsequence **M**ining (UDSM) model that automatically-learn shapelets from unlabelled time series data. We first introduce a *pseudo-label* to transform unsupervised learning into supervised learning. Then, we use the popular *regularized least-squares technique* along with *spectral analysis* to learn both the shapelets, the pseudo-class labels, and the pseudo classification boundaries. A new shapelet regularization term is also added to avoid learning similar shapelets. We adopt a coordinate descent algorithm to simultaneously obtain the pseudo-class labels and learn the best shapelets in an iterative manner.

Compared to existing unsupervised shapelet selection models [155] [93], our method aims to learn shapelets instead of simply selecting shapelets from unlabelled time series data. Our main contributions are summarized as follows:

- UDSM, the proposed new unsupervised discriminative subsequence mining model, extends supervised shapelet learning models to unlabelled time series data by combining the strengths of pseudo-class labels, spectral analysis, shapelet regularization and regularized least-squares techniques.

- We empirically validate the performance of the proposed method on a synthetic data set and 36 real-world data sets. The results show promising performance compared to the state-of-the-art unsupervised time series learning models.

The remainder of the chapter is as follows: Section 4.2 provides the preliminaries. Section 4.3 introduces the proposed UDSM model. Section 4.4 introduces the learning algorithm and provides its analysis. Section 4.5 details the experiments. And the conclusions are given in Section 4.6.

4.2 Preliminaries

In this chapter, scalars are represented by letters $(a, b, ...; \alpha, \beta, ...)$, vectors are represented by lower-case bold letters $(\mathbf{a}, \mathbf{b}, ...)$, and matrices are represented by bold-face upper-case letters $(\mathbf{A}, \mathbf{B}, ...)$. $\mathbf{a}_{(k)}$ is used to denote the k-th element of

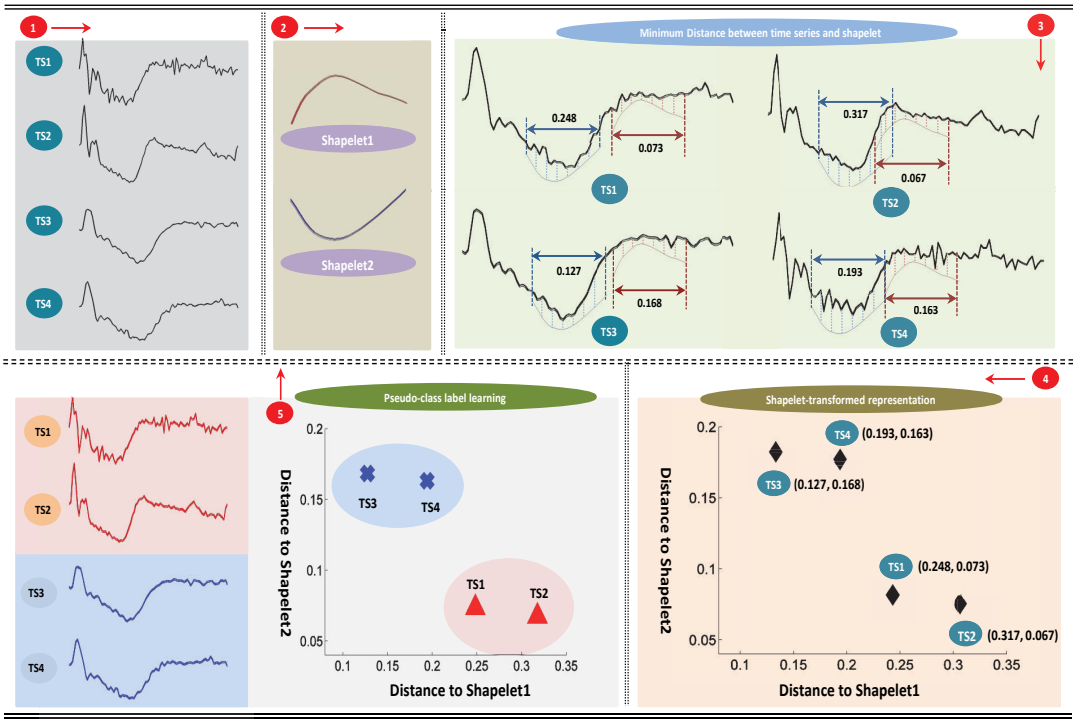


Figure 4.2: The framework of the proposed Unsupervised Discriminative Subsequence Mining (UDSM) model. From the original time series ①, we first learn shapelets using a shapelet similarity minimization principle ②. Then we calculate the minimum distance between the time series and the learned shapelets ③. We transform the time series into a shapelet-based space ④. In low-dimensional shapelet-space, the UDSM learns the pseudo-class labels and a pseudo classifier using spectral analysis and regularized least-squares minimization⑤. Then, we update the shapelet with the new learned pseudo-class labels and the pseudo classifier ②. This process is repeated until convergence. Finally, the UDSM ensures the optimal shapelets and the pseudo-class labels.

vector $\mathbf{a}$, and $\mathbf{A}_{(ij)}$ is used to denote the element located at the ith-row and j-th column in matrix $\mathbf{A}$. $\mathbf{A}_{(i,:)}$ and $\mathbf{A}_{(:,j)}$ denote the vectors of the i-th row and j-th column of the matrix. In a time series example, $\mathbf{t}_{a,b}$ denotes the segment starting from time point a to time point b.

Consider a set of time series examples $\mathbf{T} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n\}$. Each example $\mathbf{t}_i$ $(1 \leq i \leq n)$ contains an ordered set of real values denoted as $(\mathbf{t}_{i(1)}, \mathbf{t}_{i(2)}, \ldots, \mathbf{t}_{i(q_i)})$, where q_i is the length of $\mathbf{t}_i$. We wish to learn a set of the top-k most discriminative shapelets $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k\}$. Similar to the shapelet learning model in [45], we set the length of the shapelets to expand r to different length scales starting at

a minimum of l_{min}, i.e. $\{l_{min}, 2 \times l_{min}, \ldots, r \times l_{min}\}$. Each length scale $i \times l_{min}$ contains k_i shapelets and $k = \sum_{i=1}^{r} k_i$. Obviously, $\mathbf{S} \in \bigcup_{i=1}^{r} \mathbb{R}^{k_i \times (i \times l_{min})}$ where $r \times l_{min} \ll q_i$ keep the shapelets compact.

4.3 Unsupervised shapelet learning

In this section, we formulate the unsupervised discriminative subsequence mining model shown in Eq. (4.7) (Subsection 4.3.6). It combines with the spectral regularization term, the shapelet similarity regularization term and the regularized least square minimization term. A conceptual view of the proposed UDSM framework is illustrated in Fig. 4.2.

To introduce the UDSM, we first introduce the shapelet-transformed representation [45] of the time series, which transfers the time series from the original space to a shapelet-based space (Subsection 4.3.1). We then introduce the pseudo-class label, spectral analysis, least-squares minimization and shapelet similarity regularization, that are essential components for the learning model (Subsections 4.3.2 to 4.3.5).

4.3.1 Shapelet-transformed representation

Shapelet transformation was introduced by [78] to downsize a lengthy time series to a short feature vector in the shapelet feature space. The orderliness of the time series and shapelet-transformation can preserve shape information for clustering and classification.

Given a set of time series examples $\mathbf{T} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n\}$ and a set of shapelets $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k\}$, we use $\mathbf{X} \in \mathbb{R}^{k \times n}$ to denote the shapelet-transformed matrix, where each element $\mathbf{X}_{(\mathbf{s}_i, \mathbf{t}_j)}$ denotes the distance between shapelet $\mathbf{s}_i$ and time series $\mathbf{t}_j$. For simplicity, we use $\mathbf{X}_{(ij)}$ to represent $\mathbf{X}_{(\mathbf{s}_i, \mathbf{t}_j)}$ which can be calculated as

$$\mathbf{X}_{(ij)} = \min_{g=1,\ldots,\overline{q}} \frac{1}{l_i} \sum_{h=1}^{l_i} (\mathbf{t}_{j(g+h-1)} - \mathbf{s}_{i(h)}), \tag{4.1}$$

where $\overline{q} = q_j - l_i + 1$ denotes the number of segments of length l_i from series $\mathbf{t}_j$, and q_j, l_i are the lengths of time series $\mathbf{t}_j$ and shapelet $\mathbf{s}_i$ respectively.

Given a set of shapelets $\mathbf{S}$, $\mathbf{X}_{(ij)}$ is a function with respect to all candidate shapelets $\mathbf{S}$, i.e. $\mathbf{X}(\mathbf{S})_{(ij)}$. For simplicity, we omit the variable $\mathbf{S}$ and use $\mathbf{X}_{(ij)}$ instead.

The distance function in Eq. (4.1) is not continuous and thus is non-differential. Based on [45], we approximate the distance function using the *soft minimum function*

$$\mathbf{X}_{(ij)} \approx \frac{\sum_{q=1}^{\bar{q}} d_{ijq} \cdot e^{\alpha d_{ijq}}}{\sum_{q=1}^{\bar{q}} e^{\alpha d_{ijq}}}, \tag{4.2}$$

where $d_{ijq} = \frac{1}{l_i} \sum_{h=1}^{l_i} (\mathbf{t}_{j(q+h-1)} - \mathbf{s}_{i(h)})$, and α is a parameter to control precision. The soft minimum approaches the true minimum when $\alpha \to -\infty$. In our experiments, we set $\alpha = -100$ following the research of [78] in the experiments.

4.3.2 Pseudo-class labels

Unsupervised learning faces the challenge of unlabelled training examples. Thus, we introduce *pseudo-class labels*. Consider clustering a time series data set into c categories, i.e. the pseudo-class label matrix $\mathbf{Y} \in \mathbb{R}^{c \times n}$ contains c labels , where $\mathbf{Y}_{(ij)}$ indicates the probability of the j-th time series example belonging to the i-th category. Time series examples sharing the same pseudo-class label fall into the same category. If $\mathbf{Y}_{(\bar{i}j)} > \mathbf{Y}_{(i,j)}, \forall i$, then the time series example $\mathbf{t}_j$ belongs to the $\bar{i}$-th cluster.

4.3.3 Spectral analysis

Spectral analysis was introduced by [26] and has been widely used in unsupervised learning [131]. The guiding principle is that examples near to each other are likely to share the same pseudo-class label [124] [131]. Assume that $\mathbf{G} \in \mathbb{R}^{n \times n}$ is the similarity matrix of time series based on the shapelet-transformed matrix $\mathbf{X}$, then the similarity matrix can be calculated in Eq. (4.3), where σ is the RBF kernel parameter.

$$\mathbf{G}_{(ij)} = e^{-\frac{\|\mathbf{x}_{(:,i)} - \mathbf{x}_{(:,j)}\|^2}{\sigma^2}}. \tag{4.3}$$

Based on $\mathbf{G}$, we expect similar data instances will share the same pseudo-class

labels. Therefore, we can formulate a spectral regularization term as follows:

$$\begin{aligned}
&\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{b}\mathbf{G}_{(ij)}\|\mathbf{Y}_{(:,i)}-\mathbf{Y}_{(:,j)}\|_2^2,\\
&=\frac{1}{2}\sum_{k=1}^{c}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{G}_{(ij)}(\mathbf{Y}_{(ki)}-\mathbf{Y}_{(kj)})^2,\\
&=\sum_{k=1}^{c}\mathbf{Y}_{(k,:)}(\mathbf{D}_G-\mathbf{G})\mathbf{Y}_{(k,:)},\\
&=tr(\mathbf{Y}\mathbf{L}_G\mathbf{Y}).
\end{aligned} \tag{4.4}$$

where $\mathbf{L}_G = \mathbf{D}_G - \mathbf{G}$ is the Laplacian matrix and $\mathbf{D}_G$ is a diagonal matrix with the diagonal elements defined as $\mathbf{D}_G(i,i) = \sum_{j=1}^{n}\mathbf{G}_{(ij)}$.

4.3.4 Least-squares minimization

Based on the pseudo-class labels, we wish to minimize the least-squares error. Let $\mathbf{W} \in \mathbb{R}^{k\times c}$ be the classification boundary of the pseudo-class labels, and the least-squares error minimizes the following objective function:

$$\min_{\mathbf{W}} \ \|\mathbf{W}^T\mathbf{X}-\mathbf{Y}\|_F^2 \tag{4.5}$$

4.3.5 Shapelet similarity minimization

We wish to learn shapelets with diverse shapes. Specifically, we penalize the model if it outputs similar shapelets. Formally, we denote the shapelet similarity matrix as $\mathbf{H} \in \mathbb{R}^{k\times k}$, where each element $\mathbf{H}_{(\mathbf{s}_i,\mathbf{s}_j)}$ represents the similarity between two shapelets $\mathbf{s}_i$ and $\mathbf{s}_j$. For simplicity, we use $\mathbf{H}_{(ij)}$ to represent $\mathbf{H}_{(\mathbf{s}_i,\mathbf{s}_j)}$ which can be calculated as

$$\mathbf{H}_{(ij)} = e^{-\frac{\|d_{ij}\|^2}{\sigma^2}}, \tag{4.6}$$

where d_{ij} is the distance between shapelet $\mathbf{s}_i$ and shapelet $\mathbf{s}_j$. d_{ij} can be calculated by following Eq. (4.2).

4.3.6 Unsupervised discriminative subsequence mining

The formal unsupervised discriminative subsequence mining (UDSM) model is given in Eq. (4.7). It is a joint optimization problem with respect to three variables: the pseudo classification boundary $\mathbf{W}$, pseudo-class labels $\mathbf{Y}$ and the candidate shapelets $\mathbf{S}$.

$$\begin{aligned}\min_{\mathbf{W},\mathbf{S},\mathbf{Y}} \ & \frac{1}{2}tr(\mathbf{Y}\mathbf{L}_G(\mathbf{S})\mathbf{Y}^\top) + \frac{\lambda_1}{2}\|\mathbf{H}(\mathbf{S})\|_F^2 \\ & + \frac{\lambda_2}{2}\|\mathbf{W}^T\mathbf{X}(\mathbf{S}) - \mathbf{Y}\|_F^2 + \frac{\lambda_3}{2}\|\mathbf{W}\|_F^2\end{aligned} \tag{4.7}$$

In the objective function, the first term is the spectral regularization that preserves local structure information. The second term is the shapelet similarity regularization term that prefers diverse shapelets. The third and fourth terms are the regularized least square minimization to learn optimal pseudo-class labels and pseudo classifier.

Note that matrix $\mathbf{L_G}$ in Eq. (4.4), matrix $\mathbf{H}$ in Eq. (4.6), and matrix $\mathbf{X}$ in Eq. (4.2) depend on the shapelets $\mathbf{S}$. We explicitly write these matrices as variables with respect to shapelets $\mathbf{S}$ in Eq. (4.7), i.e. $\mathbf{L_G(S)}$, $\mathbf{H(S)}$ and $\mathbf{X(S)}$ respectively. We propose a coordinate descent algorithm subsequently to solve the proposed model.

4.4 The algorithm

Here, we first introduce a coordinate descent algorithm to solve the UDSM model, and then analyze its convergence properties, initialization methods and the complexity analysis.

4.4.1 Learning algorithm

In the coordinate descent algorithm, we iteratively update one variable by fixing the remaining two variables. The steps are repeated until convergence. Algorithm 2 summarizes the steps.

Algorithm 2 UDSM

Require:

$\mathbf{T}$: unsupervised time series

k, r, l_{min}: parameters on shapelet number and length

i_{max}: maximum internal iterations

η: learning rate

$\lambda_1, \lambda_2, \lambda_3$: weight parameters

α, σ: similarity parameter and kernel parameter

Ensure:

$\mathbf{S}^*$: best shapelets

$\mathbf{Y}^* \in \mathbb{R}^{c \times n}$: optimal pseudo-class labels

Initialize: $\mathbf{S}_0, \mathbf{W}_0, \mathbf{Y}_0$

while Not convergent **do**

  Calculate: $\mathbf{X}_t(\mathbf{T}, \mathbf{s}_t|\alpha)$ based on Eq. (4.2)

    $\mathbf{L}_{Gt}(\mathbf{T}, \mathbf{s}_t|\alpha, \sigma)$ based on Eq. (4.4)

    $\mathbf{H}_t(\mathbf{s}_t|\alpha)$ based on Eq. (4.6)

  update $\mathbf{Y}_{t+1}$, $\mathbf{W}_{t+1}$**:**

    $\mathbf{Y}_{t+1} \leftarrow \lambda_2 \mathbf{W}_t^T \mathbf{X}_t (\mathbf{L}_{Gt} + \lambda_2 \mathbf{I})^{-1}$

    $\mathbf{W}_{t+1} \leftarrow (\lambda_2 \mathbf{X}_t \mathbf{X}_t^T + \lambda_3 \mathbf{I})^{-1} (\lambda_2 \mathbf{X}_t \mathbf{Y}_{t+1}^T)$.

  update $\mathbf{S}_{t+1}$**:**

    for $i = 1, \ldots, i_{max}$ **do**

      $\mathbf{S}_{i+1} \leftarrow \mathbf{S}_i - \eta \nabla \mathbf{S}_i$

      $\nabla \mathbf{S}_i = \frac{\partial \mathbf{F}(\mathbf{S}_i|\mathbf{X}_{t+1}, \mathbf{Y}_{t+1})}{\partial \mathbf{S}}$ is from Eq. (4.17)

    end for

    $\mathbf{S}_{t+1} = \mathbf{S}_{i_{max}+1}$

  $t \leftarrow t + 1$

return $\mathbf{S}^* = \mathbf{S}_{t+1}$; $\mathbf{Y}^* = \mathbf{Y}_{t+1}$; $\mathbf{W}^* = \mathbf{W}_{t+1}$.

4.4.1.1 Update Y by fixing W and S

By fixing $\mathbf{W}$ and $\mathbf{S}$, the function in Eq. (4.7) degenerates to Eq. (4.8),

$$\min_{\mathbf{Y}} \ \mathbf{F}(\mathbf{Y}) = \frac{1}{2} tr(\mathbf{Y} \mathbf{L}_G \mathbf{Y}^T) + \frac{\lambda_2}{2} \|\mathbf{W}^T \mathbf{X} - \mathbf{Y}\|_F^2 \tag{4.8}$$

The derivative of Eq. (4.8) with respect to $\mathbf{Y}$ is,

$$\begin{aligned} \frac{\partial \mathbf{F}_\mathbf{Y}}{\partial \mathbf{Y}} &= \mathbf{Y} \mathbf{L}_G - \lambda_2 (\mathbf{W}^T \mathbf{X} - \mathbf{Y}) \\ &= \mathbf{Y} (\mathbf{L}_G - \lambda_2 \mathbf{I}) - \lambda_2 \mathbf{W}^T \mathbf{X} \end{aligned} \tag{4.9}$$

Let Eq. (4.9) equal 0, the solution for $\mathbf{Y}$ can be obtained from

$$\mathbf{Y} = \lambda_2 \mathbf{W}^T \mathbf{X}(\mathbf{L}_G + \lambda_2 \mathbf{I})^{-1}. \tag{4.10}$$

where $\mathbf{I}$ is an identity matrix. Thus, the update of $\mathbf{Y}$ is

$$\mathbf{Y}_{t+1} = \lambda_2 \mathbf{W}_t^\top \mathbf{X}_t(\mathbf{L}_{Gt} + \lambda_2 \mathbf{I})^{-1}. \tag{4.11}$$

4.4.1.2 Update W by fixing S and Y

By fixing $\mathbf{S}$ and $\mathbf{Y}$, Eq. (4.7) degenerates to

$$\min_{\mathbf{W}} \ \mathbf{F}(\mathbf{W}) = \frac{\lambda_2}{2}\|\mathbf{W}^T\mathbf{X} - \mathbf{Y}\|_F^2 + \frac{\lambda_3}{2}\|\mathbf{W}\|_F^2 \tag{4.12}$$

The derivative of Eq. (4.12) with respect to $\mathbf{W}$ is,

$$\begin{aligned} \frac{\partial \mathbf{F}_\mathbf{W}}{\partial \mathbf{W}} &= \lambda_2 \mathbf{X}(\mathbf{W}^T\mathbf{X} - \mathbf{Y})^T + \lambda_3 \mathbf{W} \\ &= (\lambda_2 \mathbf{X}\mathbf{X}^T + \lambda_3 \mathbf{I})\mathbf{W} - \lambda_2 \mathbf{X}\mathbf{Y}^T \end{aligned} \tag{4.13}$$

Let Eq. (4.13) equal 0, we obtain the solution for $\mathbf{W}$ as

$$\mathbf{W} = (\lambda_2 \mathbf{X}\mathbf{X}^T + \lambda_3 \mathbf{I})^{-1}(\lambda_2 \mathbf{X}\mathbf{Y}^T). \tag{4.14}$$

Thus, the update of $\mathbf{W}$ is

$$\mathbf{W}_{t+1} = (\lambda_2 \mathbf{X}_t \mathbf{X}_t^\top + \lambda_3 \mathbf{I})^{-1}(\lambda_2 \mathbf{X}_t \mathbf{Y}_{t+1}^T). \tag{4.15}$$

4.4.1.3 Update S by fixing W and Y

By fixing $\mathbf{W}$ and $\mathbf{Y}$, Eq. (4.7) degenerates to

$$\begin{aligned} \min_{\mathbf{S}} \ \mathbf{F}(\mathbf{S}) =& \frac{1}{2} tr(\mathbf{Y}\mathbf{L}_G(\mathbf{S})\mathbf{Y}^T) \\ &+ \frac{\lambda_1}{2}\|\mathbf{H}(\mathbf{S})\|_F^2 + \frac{\lambda_2}{2}\|\mathbf{W}^T\mathbf{X}(\mathbf{S}) - \mathbf{Y}\|_F^2 \end{aligned} \tag{4.16}$$

Eq. (4.16) is non-convex and we cannot explicitly solve $\mathbf{S}$ as we could in finding

$\mathbf{W}$ and $\mathbf{Y}$. Instead, we resort to an iterative algorithm by setting a learning rate η, i.e. $\mathbf{S}_{i+1} = \mathbf{S}_i - \eta \nabla \mathbf{S}_i$, where $\nabla \mathbf{S}_i = \frac{\partial \mathbf{F}(\mathbf{S}_i)}{\partial \mathbf{S}}$. The iterative steps guarantee the convergence of the objective function. The derivative of Eq. (4.16) with respect to $\mathbf{S}_{(mp)}$ is

$$\begin{aligned}\frac{\partial \mathbf{F}(\mathbf{S})}{\partial \mathbf{S}_{(mp)}} =& \frac{1}{2}\mathbf{Y}^T\mathbf{Y}\frac{\partial \mathbf{L}_G(\mathbf{S})}{\partial \mathbf{S}_{(mp)}} \\ &+\lambda_1 \mathbf{H}(\mathbf{S})\frac{\partial \mathbf{H}(\mathbf{S})}{\partial \mathbf{S}_{(mp)}} \\ &+\lambda_2 \mathbf{W}(\mathbf{W}^T X - Y)\frac{\partial \mathbf{X}(\mathbf{S})}{\partial \mathbf{S}_{(mp)}}\end{aligned} \tag{4.17}$$

where $m = 1, \ldots, k$, and $p = 1, \ldots, l_m$.

Because $\mathbf{L}_G = \mathbf{D}_G - \mathbf{G}$ and $\mathbf{D}_G(i,i) = \sum_{j=1}^{n} \mathbf{G}_{(ij)}$, the first term in Eq. (4.17) turns to calculating $\partial \mathbf{G}_{(ij)} / \partial \mathbf{S}_{(mp)}$ as shown in Eq. (4.18),

$$\frac{\partial \mathbf{G}_{(ij)}}{\partial \mathbf{S}_{(mp)}} = -\frac{2\mathbf{G}_{(ij)}}{\sigma^2}\Big(\sum_{q=1}^{k}(\mathbf{X}_{(qi)} - \mathbf{X}_{(qj)})\Big)\Big(\frac{\partial \mathbf{X}_{(qi)}}{\partial \mathbf{S}_{(mp)}} - \frac{\partial \mathbf{X}_{(qj)}}{\partial \mathbf{S}_{(mp)}}\Big), \tag{4.18}$$

and

$$\frac{\partial \mathbf{X}_{(ij)}}{\partial \mathbf{S}_{(mp)}} = \frac{1}{E_1^2}\sum_{q=1}^{\overline{q}_{ij}} e^{\alpha d_{ijq}}((1+\alpha d_{ijq})E_1 - \alpha E_2)\frac{\partial d_{ijq}}{\partial \mathbf{S}_{(mp)}}, \tag{4.19}$$

where $E_1 = \sum_{q=1}^{\overline{q}_{ij}} e^{\alpha d_{ijq}}$, $E_2 = \sum_{q=1}^{\overline{q}_{ij}} d_{ijq} e^{\alpha d_{ijq}}$, $\overline{q}_{ij} = q_j - l_i + 1$ and $d_{ijq} = \frac{1}{l_i}\sum_{h=1}^{l_i}(\mathbf{t}_{j(q+h-1)} - \mathbf{s}_{i(h)})$.

$$\frac{\partial d_{ijq}}{\partial \mathbf{S}_{(mp)}} = \left\{ \begin{array}{ll} 0 & if \quad i \neq m \\ \frac{2}{l_m}(\mathbf{S}_{(mp)} - T_{j,q+p-1}) & if \quad i = m \end{array} \right\} \tag{4.20}$$

The second term in Eq. (4.17) turns to calculating Eq. (4.21),

$$\frac{\partial \mathbf{H}_{(ij)}}{\partial \mathbf{S}_{(mp)}} = -\frac{2}{\sigma^2}\tilde{d}_{ij} e^{-\frac{1}{\sigma^2}\tilde{d}_{ij}^2}\frac{\partial \tilde{d}_{ij}}{\partial \mathbf{S}_{(mp)}} \tag{4.21}$$

where $\tilde{d}_{ij}$ is the distance between shapelets $\mathbf{s}_i$ and $\mathbf{s}_j$. The calculation of $\tilde{d}_{ij}$ and $\frac{\partial \tilde{d}_{ij}}{\partial \mathbf{S}_{(mp)}}$ is similar to $\mathbf{X}_{(ij)}$ and $\frac{\partial \mathbf{X}_{(ij)}}{\partial \mathbf{S}_{(mp)}}$ respectively.

In summary, we can calculate the gradient $\nabla \mathbf{S}_i = \frac{\partial \mathbf{F}(\mathbf{S})}{\partial \mathbf{S}_i}$ using Eqs. (4.17)-(4.21). In the following, we discuss the convergence of the coordinate descent

Table 4.1: Statistics of the benchmark time series data sets

No.	Data set	# Train\Test	Length	classes
1	ArrowHead	36\175	251	3
2	Beef	30\30	470	5
3	BeetleFly	20\20	512	2
4	BirdChicken	20\20	512	2
5	Car	60\60	577	4
6	ChlorineConcentration	467\3840	166	3
7	Coffee	28\28	286	2
8	DiatomSizeReduction	16\306	345	4
9	DistalPhalanxOutlineAgeGroup	400\139	80	3
10	DistalPhalanxOutlineCorrect	600\276	80	2
11	ECG200	100\100	96	2
12	ECGFiveDays	23\861	136	2
13	GunPoint	50\150	150	2
14	Ham	109\105	431	2
15	Herring	64\64	512	2
16	Lightning2	60\61	637	2
17	Meat	60\60	448	3
18	MiddlePhalanxOutlineAgeGroup	400\154	80	3
19	MiddlePhalanxOutlineCorrect	600\291	80	2
20	MiddlePhalanxTW	399\154	80	6
21	MoteStrain	20\1252	84	2
22	OSULeaf	200\242	427	6
23	Plane	105\105	144	7
24	ProximalPhalanxOutlineAgeGroup	400\205	80	3
25	ProximalPhalanxTW	400\205	80	6
26	SonyAIBORobotSurface1	20\601	70	2
27	SonyAIBORobotSurface2	27\953	65	2
28	SwedishLeaf	500\625	128	15
29	Symbols	25\995	398	6
30	ToeSegmentation1	40\228	277	2
31	ToeSegmentation2	36\130	343	2
32	TwoLeadECG	23\1139	82	2
33	TwoPatterns	1000\4000	128	4
34	Wafer	1000\6164	152	2
35	Wine	57\54	234	2
36	WordsSynonyms	267\638	270	25

algorithm in solving Eq. (4.7).

4.4.2 Convergence

In Algorithm 2, convergence depends on the stepwise decent. When updating $\mathbf{Y}$ or $\mathbf{W}$, we know that $\mathbf{Y}_{t+1} = \mathbf{Y}^*(\mathbf{W}_t, \mathbf{S}_t)$ and $\mathbf{W}_{t+1} = \mathbf{W}^*(\mathbf{Y}_{t+1}, \mathbf{S}_t)$, thus the objective functions in problems (4.8) and (4.12) keep decreasing in each iteration. When updating $\mathbf{S}$, the objective function in Eq. (4.16) is not convex and a closed-form derivative is difficult to obtain. So, we introduce an internal iterative process (line 10-13) using a gradient descent algorithm to update $\mathbf{S}$. In the internal iterations, as long as we set an appropriate learning rate η that is usually very small, the objective function in Eq. (4.16) will decrease to convergence. Therefore, Algorithm 2 converges to its optima as long as it adopts a small learning rate.

In addition, the objective function in Eq. (4.7) is non-convex but has a lower bound of 0 because it is non-negative, which means Algorithm 2 can only converges to the local optima. In the experiments, we run the algorithm several times under different initializations and choose the best solution as the output.

4.4.3 Initialization

Since Algorithm 2 only converges to its local optima, we discuss initialization skills here to improve the performance. The algorithm expects to initialize $\mathbf{S}_0$, $\mathbf{Y}_0$ and $\mathbf{W}_0$. Specifically, we initialize $\mathbf{S}_0$ using the centroids of the segments with the same length as the shapelets, since the centroids can typically represent the main patterns behind the data. Based on $\mathbf{S_0}$, we transfer the original time series into the shapelet-based space and obtain the shapelet-transformed matrix $\mathbf{X}(\mathbf{S_0})$. In shapelet-based space, we initialize $\mathbf{Y}_0$ with the cluster labels and $\mathbf{W}_0$ with the centers of the clusters obtained by k-means. This initialization enables fast convergence.

4.4.4 Complexity analysis

In this section, we discuss the computational cost of the proposed UDSM algorithm .

As start in Algorithm 2, UDSM takes a constant amount of time for the initialization (line 1). Then, it is solved iteratively. When matrix $\mathbf{X_t}$ is calculated in each iteration (line 3), suppose the maximum length of the time series in T is q, the maximum length of shapelets is l, it takes $O(qnkl)$, where n is the size of T and k is the number of shapelets learned. Similarly, when calculate matrix $\mathbf{L_{G_t}}$ (line 4) and $\mathbf{H_t}$ (line 5), it takes $O(kn^2)$ and $O(l^2k^2)$ respectively. When update pseudo-class label matrix $\mathbf{Y_{t+1}}$ and pseudo classifier $\mathbf{W_{t+1}}$ (line 6-8), it takes $O(ckn + cn^2 + n^3)$ and $O(nk^2 + k^3 + nkc + k^2c)$ respectively where c is the number of clusters. When update S_{t+1} (line 9-14), it takes $O(i_{max}(q^2n^2 + n^2c + k^2l^2 + k^3 + ckn))$ where i_{max} is the maximum number of the internal iterations. To sum up, the total worst-case time complexity of Algorithm 2 is $O(I(qnlk + kn^2 + lk^2 + cnk^2 + n^3 + kn + cn^2 + +k^3 + k^2c + i_{max}(q^2n^2 + n^2c + k^2l^2 + k^3 + ckn))$ where I suppose to be the maximum number of iterations of the whole algorithm. Since $l \ll q$, $k \ll n$ and $c \ll n$, the total time complexity cost of the proposed method is $O(I(n^3 + q^2n^2i_{max}))$.

4.5 Experiments

A series of experiments are carried out to validate the performance of UDSM. All experiments are conducted on a Windows 7 machine with 3.00GHz CPU and 8GB memory.

4.5.1 Data sets

We use 36 time series benchmark data sets downloaded from the UCR time series archive [66] [13] to validate the performance of proposed algorithm in this section. Each data set contains between 40 and 7164 sequences. The sequences in each data set are in equal length, while the sequence length varies between 80 and 637 from one data set to another. These data sets are annotated and every sequence belongs to only one class. The data sets are split into training and test sets. We

use the training sets for tuning the parameters and the test sets for evaluation. The details are shown in Table 4.1.

4.5.2 Measures

Existing measures, used to evaluate the performance of time series clustering, include the Rand Index (RI) [103], Jaccard score [87], Normalized Mutual Information (NMI) [158], and the Folkes and Mallow index [49] [155]. Among these, the Rand Index and Normalized Mutual Information are the most popular, while the remaining measures can be considered variants. As such, we use the *Rand Index* and *Normalized Mutual Information* as our evaluation methods.

Table 4.2: Rand Index (RI) comparisons on 36 time series data sets.

Data set	k-means	UDFS	NDFS	RUFS	RSFS	KSC	KDBA	k-shape	u-shapelte	UDSM
ArrowHead	0.6905	0.7254	0.7381	**0.7476**	0.7108	0.7254	0.7222	0.7254	0.6460	0.7159
Beef	0.6713	0.6759	0.7034	**0.7149**	0.6975	0.7057	0.6713	0.5402	0.6966	0.6966
BeetleFly	0.4789	0.4947	0.5579	0.6053	0.6516	0.6053	0.6053	0.6053	0.7316	**0.8105**
BirdChicken	0.4947	0.4947	0.7316	0.5579	0.6632	0.7316	0.6053	0.6632	0.5579	**0.8105**
Car	0.6345	0.6757	0.6260	0.6667	0.6708	0.6898	0.6254	0.7028	0.6418	**0.7345**
ChlorineConcentration	0.5241	0.5282	0.5225	**0.5330**	0.5316	0.5256	0.5300	0.4111	0.5318	0.4997
Coffee	0.7460	0.8624	**1.0000**	0.5476	**1.0000**	**1.0000**	0.4841	**1.0000**	**1.0000**	**1.0000**
DiatomSizeReduction	0.9583	0.9583	0.9583	0.9333	0.9167	**1.0000**	0.9583	**1.0000**	0.7083	**1.0000**
Dist.Phal.Outl.AgeGroup	0.6171	0.6531	0.6239	0.6252	0.6539	0.6535	**0.6750**	0.6020	0.6273	0.6650
Dist.Phal.Outl.Correct	0.5252	0.5362	0.5383	0.5252	0.5327	0.5235	0.5203	0.5252	0.5098	**0.5962**
ECG200	0.6315	0.6533	0.6315	0.7018	0.6916	0.6315	0.6018	0.7018	0.5758	**0.7285**
ECGFiveDays	0.4783	0.5020	0.5573	0.5020	0.5953	0.5257	0.5573	0.5020	0.5968	**0.8340**
GunPoint	0.4971	0.5029	0.5102	0.6498	0.4994	0.4971	0.5420	0.6278	0.6278	**0.7257**
Ham	0.5025	0.5219	0.5362	0.5107	0.5127	0.5362	0.5141	0.5311	0.5362	**0.6393**
Herring	0.4965	0.5099	0.5164	0.5238	0.5151	0.4940	0.5164	0.4965	0.5417	**0.6190**
Lighting2	0.4966	0.5119	0.5373	0.5729	0.5269	0.6263	0.5119	0.6548	0.5192	**0.6955**
Meat	0.6595	0.6483	0.6635	0.6578	0.6657	0.6723	0.6816	0.6575	0.6742	**0.7740**
Mid.Phal.Outl.AgeGroup	0.5351	0.5269	0.5350	0.5315	0.5473	0.5364	0.5513	0.5105	0.5396	**0.5807**
Mid.Phal.Outl.Correct	0.5000	0.5431	0.5047	0.5114	0.5149	0.5014	0.5563	0.5114	0.5218	**0.6635**
Mid.Phal.TW	0.0983	0.1225	0.1919	0.7920	0.8062	**0.8187**	0.8046	0.6213	0.7920	0.7920
MoteStrain	0.4947	0.5579	0.6053	0.5579	0.6168	0.6632	0.4789	0.6053	0.4789	**0.8105**
OSULeaf	0.5615	0.5372	0.5622	0.5497	0.5665	0.5714	0.5541	0.5538	0.5525	**0.6551**
Plane	0.9081	0.8949	0.8954	0.9220	0.9314	0.9603	0.9225	0.9901	**1.0000**	**1.0000**
Prox.Phal.Outl.AgeGroup	0.5288	0.4997	0.5463	0.5780	0.5384	0.5305	0.5192	0.5617	0.5206	**0.7939**
Prox.Phal.TW	0.4789	0.4947	0.6053	0.5579	0.5211	0.6053	0.5211	0.5211	0.4789	**0.7282**

Table 4.3: Rand Index (RI) comparisons on 36 time series data sets (continued with Table 4.2).

Data set	k-means	UDFS	NDFS	RUFS	RSFS	KSC	KDBA	k-shape	u-shapelte	UDSM
SonyAIBORobotSurface	0.7721	0.7695	0.7721	0.7787	0.7928	0.7726	0.7988	0.8084	0.7639	**0.8105**
SonyAIBORobotSurfaceII	0.8697	0.8745	0.8865	0.8756	0.8948	**0.9039**	0.8684	0.5617	0.8770	0.8575
SwedishLeaf	0.4987	0.4923	0.5500	0.5192	0.5038	0.4923	0.5500	0.5333	0.6154	**0.8547**
Symbols	0.8810	0.8548	0.8562	0.8525	0.9060	0.8982	**0.9774**	0.8373	0.9603	0.9200
ToeSegmentation1	0.4873	0.4921	0.5873	0.5429	0.4968	0.5000	0.6143	0.6143	0.5873	**0.6718**
ToeSegmentation2	0.5257	0.5257	0.5968	0.5968	0.5826	0.5257	0.5573	0.5257	0.5020	**0.6778**
TwoPatterns	0.8529	0.8259	0.8530	0.8385	**0.8588**	0.8585	0.8446	0.8046	0.7757	0.8318
TwoLeadECG	0.5476	0.5495	0.6328	0.8246	0.5635	0.5464	0.5476	0.8246	0.5404	**0.8628**
wafer	0.4925	0.4925	0.5263	0.5263	0.4925	0.4925	0.4925	0.4925	0.4925	**0.8246**
Wine	0.4984	0.4987	0.5123	0.5021	0.5033	0.5006	0.5064	0.5001	0.5033	**0.8985**
WordsSynonyms	0.8775	0.8697	0.8760	**0.8861**	0.8817	0.8727	0.8159	0.7844	0.8230	0.8540
Average	0.5975	0.6077	0.6402	0.6478	0.6543	0.6582	0.6334	0.6419	0.6402	**0.7676**
(improvement)	28.47%	26.31%	19.90%	18.49%	17.32%	16.62%	21.19%	19.58%	19.90%	

Table 4.4: Normalized Mutual Information (NMI) comparisons on 36 time series data sets.

Data set	k-means	UDFS	NDFS	RUFS	RSFS	KSC	KDBA	k-shape	u-shapelte	UDSM
ArrowHead	0.4816	0.5240	0.4997	0.5975	0.5104	0.5240	0.4816	0.5240	0.3522	**0.6322**
Beef	0.2925	0.2718	0.3647	0.3799	0.3597	**0.3828**	0.3340	0.3338	0.3413	0.3338
BeetleFly	0.0073	0.0371	0.1264	0.1919	0.2795	0.2215	0.2783	0.3456	0.5105	**0.5310**
BirdChicken	0.0371	0.0371	0.3988	0.1187	0.3002	0.3988	0.2167	0.3456	0.2783	**0.6190**
Car	0.2540	0.2319	0.2361	0.2511	0.2920	0.2719	0.2691	0.3771	0.3655	**0.4650**
ChlorineConcentration	0.0129	0.0138	0.0075	**0.0254**	0.0159	0.0147	0.0164	0.0000	0.0135	0.0133
Coffee	0.5246	0.6945	**1.0000**	0.2513	**1.0000**	**1.0000**	0.0778	**1.0000**	**1.0000**	**1.0000**
DiatomSizeReduction	0.9300	0.9300	0.9300	0.8734	0.8761	**1.0000**	0.9300	**1.0000**	0.4849	**1.0000**
Dist.Phal.Outl.AgeGroup	0.1880	0.3262	0.1943	0.2762	0.3548	0.3331	**0.4261**	0.2911	0.2577	0.3846
Dist.Phal.Outl.Correct	0.0278	0.0473	0.0567	0.1071	0.0782	0.0261	0.0199	0.0527	0.0063	**0.1626**
ECG200	0.1403	0.1854	0.1403	0.2668	0.2918	0.1403	0.1886	0.3682	0.1323	**0.3776**
ECGFiveDays	0.0002	0.0600	0.1296	0.0352	0.1760	0.0682	0.1983	0.0002	0.1498	**0.6502**
GunPoint	0.0126	0.0220	0.0334	0.2405	0.0152	0.0126	0.1288	0.3653	0.3653	**0.4878**
Ham	0.0093	0.0389	0.0595	0.0980	0.0256	0.0595	0.0265	0.0517	0.0619	**0.3411**
Herring	0.0013	0.0253	0.0225	0.0518	0.0236	0.0027	0.0000	0.0027	0.1324	**0.1718**
Lighting2	0.0038	0.0047	0.0851	0.1426	0.0326	0.1979	0.0850	0.2670	0.0144	**0.3727**
Meat	0.2510	0.2832	0.2416	0.1943	0.3016	0.2846	0.3661	0.2254	0.2716	**0.9085**
Mid.Phal.Outl.AgeGroup	0.0219	0.1105	0.0416	0.1595	0.0968	0.1061	0.1148	0.0722	0.1491	**0.2780**
Mid.Phal.Outl.Correct	0.0024	0.0713	0.0150	0.0443	0.0321	0.0053	0.0760	0.0349	0.0253	**0.2503**
MiddlePhalanxTW	0.4134	0.4276	0.4149	0.5366	0.4217	0.4486	0.4497	0.5229	0.4065	**0.9202**
MoteStrain	0.0551	0.1187	0.1919	0.1264	0.2373	0.3002	0.0970	0.2215	0.0082	**0.5310**
OSULeaf	0.0208	0.0200	0.0352	0.0246	0.0463	0.0421	0.0327	0.0126	0.0203	**0.3353**
Plane	0.8598	0.8046	0.8414	0.8675	0.8736	0.9218	0.8784	0.9642	**1.0000**	**1.0000**
Prox.Phal.Outl.AgeGroup	0.0635	0.0182	0.0830	0.0726	0.0938	0.0682	0.0377	0.0110	0.0332	**0.6813**
Prox.Phal.TW	0.0082	0.0308	0.2215	0.1187	0.0809	0.1919	0.2167	0.1577	0.0107	**1.0000**

Table 4.5: Normalized Mutual Information (NMI) comparisons on 36 time series data sets (continued with Table 4.4)

Data set	k-means	UDFS	NDFS	RUFS	RSFS	KSC	KDBA	k-shape	u-shapelte	UDSM
SonyAIBORobotSurface	0.6112	0.6122	0.6112	0.6278	0.6368	0.6129	0.5516	**0.7107**	0.5803	0.5597
SonyAIBORobotSurfaceII	0.5444	0.4802	0.5413	0.5107	0.5406	0.5619	0.5481	0.0110	0.5903	**0.6858**
SwedishLeaf	0.0168	0.0082	0.0934	0.0457	0.0269	0.0073	0.1277	0.1041	0.3456	**0.9186**
Symbols	0.7780	0.7277	0.7593	0.7174	0.8027	0.8264	**0.9388**	0.6366	0.8691	0.8821
ToeSegmentation1	0.0022	0.0089	0.2141	0.0880	0.0174	0.0202	0.2712	0.3073	0.3073	**0.3351**
ToeSegmentation2	0.0863	0.0727	0.1713	0.1713	0.1625	0.0863	0.2627	0.0863	0.1519	**0.4308**
TwoPatterns	0.4696	0.3393	0.4351	0.4678	0.4608	0.4705	0.4419	0.3949	0.2979	**0.4911**
TwoLeadECG	0.0000	0.0004	0.1353	0.1238	0.0829	0.0011	0.0103	0.0000	0.0529	**0.5471**
wafer	0.0010	0.0010	0.0546	**0.0746**	0.0194	0.0010	0.0000	0.0010	0.0010	0.0492
Wine	0.0031	0.0045	0.0259	0.0065	0.0096	0.0094	0.0211	0.0119	0.0171	**0.7511**
WordsSynonyms	0.5435	0.4745	0.5396	0.5623	**0.5462**	0.4874	0.4527	0.4154	0.3933	0.4984
Average	0.2132	0.2240	0.2764	0.2624	0.2812	0.2808	0.2659	0.2841	0.2777	**0.5443**
(Improvement)	155.30%	142.99%	96.92%	107.43%	93.56%	93.84%	104.70%	91.59%	96.00%	

To calculate the Rand Index, we compare the cluster labels $\mathbf{Y}^*$ generated by the clustering algorithm with the genuine class labels $\mathbf{L}_{true}$

$$Rand\ index = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4.22}$$

where TP is the number of time series pairs that are assigned to the same cluster in $\mathbf{Y}^*$, and belong to the same class in $\mathbf{L}_{true}$, TN is the number of time series pairs that are assigned to different clusters in $\mathbf{Y}^*$ and belong to different classes in $\mathbf{L}_{true}$, FP is the number of time series pairs that are assigned to the same cluster in $\mathbf{Y}^*$ but belong to different classes in $\mathbf{L}_{true}$, and FN is the number of time series pairs that are assigned to different clusters in $\mathbf{Y}^*$ but belong to the same class in $\mathbf{L}_{true}$.

Normalized Mutual Information, NMI, can be calculated using the following equation:

$$NMI = \frac{\sum_{i=1}^{M}\sum_{j=1}^{M} N_{ij} log(\frac{N \cdot N_{ij}}{|G_i||A_j|})}{\sqrt{(\sum_{i=1}^{M} |G_i| log\frac{|G_i|}{N})(\sum_{j=1}^{M} |A_j| log\frac{A_j}{N})}} \tag{4.23}$$

where N is the total number of time series in the data set. $|G_i|$ denotes the number of time series in cluster G_i, $|A_j|$ is the number of time series in cluster A_j, and $N_{ij} = |G_i \cap A_j|$represents the number of time series in the intersection of the two sets.

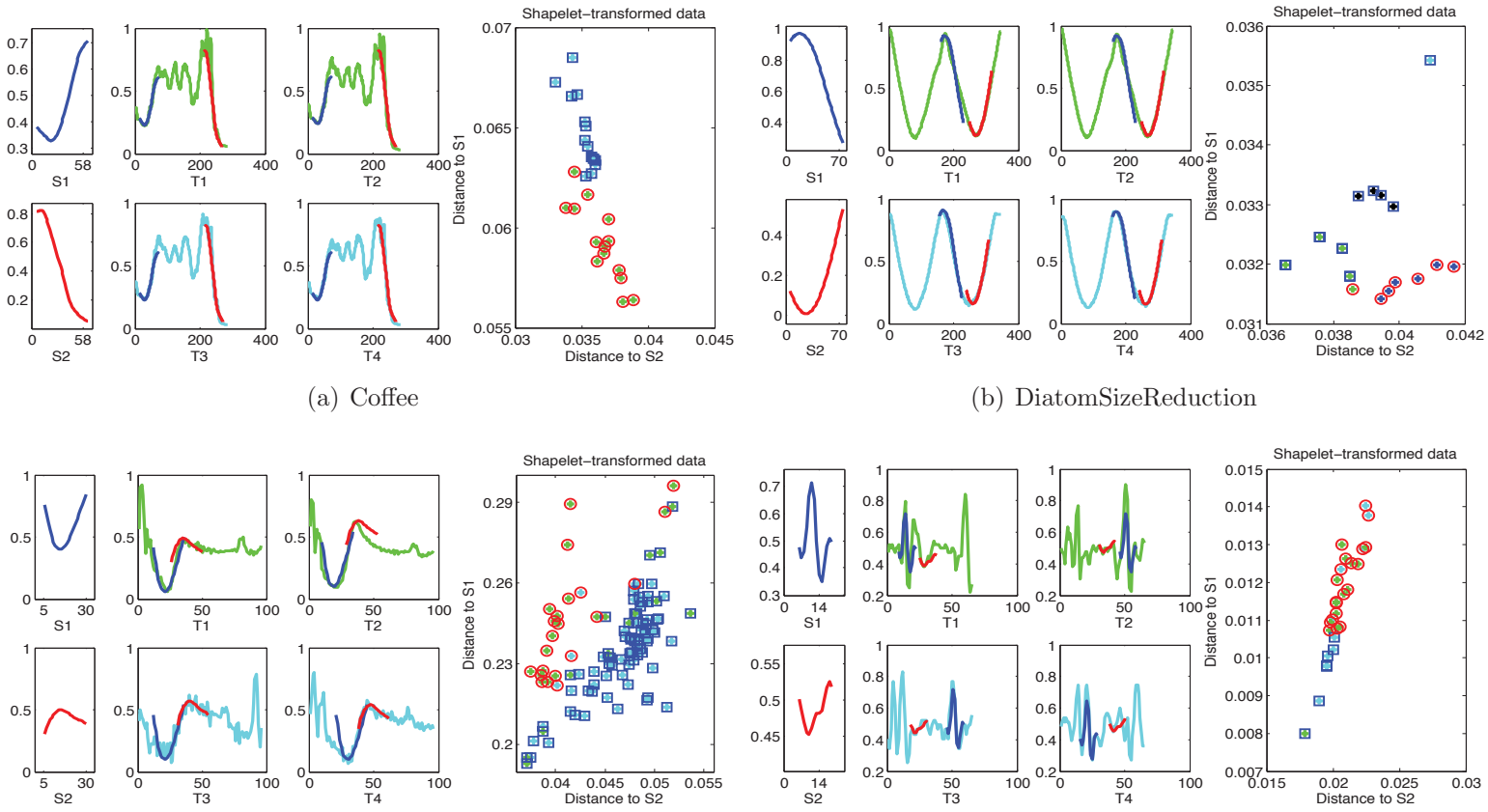
In these two measures, values close to 1 indicates high quality clustering [155] [93].

4.5.3 Baseline methods

We compare UDSM with nine representative unsupervised feature selection and feature learning methods for time series. They consist of three type of algorithms: original-timeseries-based unsupervised feature selection methods, including k-means, UDFS, NDFS, RUFS and RSFS; distance-based unsupervised time series clustering methods, including KSC, k-DBA, k-shape and shapelet-based unsupervised feature learning methods, i.e. the u-shapelet method. The details are as follow:

- K-means: Using k-means on the entire time series.

• UDFS [149]: Unsupervised discriminative feature selection which simultaneously explores local discriminative information, feature correlations and the manifold structure.
• NDFS [76]: Nonnegative discriminative feature selection which adopts $l_{2,1}$ regularized regression and nonnegative spectral analysis as a joint framework to select features.
• RUFS [98]: Robust unsupervised feature selection which uses local learning regularized robust orthogonal nonnegative matrix factorization to perform robust label learning and $l_{2,1}$-norms minimization to perform robust feature learning jointly.
• RSFS [115]: Robust spectral learning for unsupervised feature selection, which joins sparse spectral regression with the robustness of graph embedding.
• KSC [142]: Using k-means for clustering by adopting a distance measure for pairwise scaling and shifting the time series and computing the spectral norm of a matrix for centroid computation.
• k-DBA [97]: k-means adopts dynamic time warping distance as a distance measure and uses a DBA method for centroid computation.
• k-shape [93]: uses a normalized version of the cross-correlation measure to consider the shapes of time series and relies on a scalable iterative refinement procedure which is efficient and accurate.
• u-shapelet [155]: a time series clustering method that deliberately ignores the rest of the data and uses local patterns to cluster the time series.



(a) Coffee

(b) DiatomSizeReduction

(c) ECG200

(d) SonyAIBoRobotSurfaceII

Figure 4.3: An illustration of UDSM on the four data sets: Coffee, DiatomSizeReduction, ECG200 and SonyAIBoRobotSurfaceII. The left part of each figure shows the two shapelets S1 and S2 learned from the original time series. The middle shows the closest match of the learned shapelets to the original time series examples, where T_1 and T_2 are drawn from one of the classes and T_3 and T_4 are drawn from the other class. We can see that the learned shapelets are discriminative features. The right shows the shapelet-transformed plots of the original time series data set. Different colors show different class labels. The dots in red circles and blue squares are the labels obtained by UDSM, which closely matches the original class labels.

We tune the parameters of these benchmark methods according to the descriptions in their original papers. To establish the best parameters for our novel method, UDSM, we tune a series of hyper-parameters. We use a grid search approach: the number of shapelets k is searched in a range of $\{1, 2, \ldots, 5\}$, and the minimum length of shapelets is searched in the range of $l_{min} \in \{0.05, 0.1, \ldots, 0.25\}$, which represents a fraction of the length of time series, e.g. $l_{min} = 0.05$ means 5% of the original time series length. Parameter r is searched from $\{1, 2, 3\}$. The weight parameters $\lambda_1, \lambda_2, \lambda_3$ are searched from $\{10^{-8}, 10^{-6}, \ldots, 10^{8}\}$. The learning rate is kept fixed at $\eta = 0.01$, and the maximum number of iterations is set to 50 which has been experimentally proven to be sufficiently large. The number of clusters is set to be as same as the number of classes of original data to evaluate the performance. Additionally, we set $\alpha = -100$ by following [45] and set $\sigma = 1$ experimentally.

4.5.4 Comparison

We compare our method UDSM against the selected baselines using RI and NMI measures. Experimental results on the 36 data sets are summarized in Tables 4.2 to 4.4. The best result for each data set is highlighted in bold.

From the two tables, we observe that the UDSM approach is effective, and performance is significantly improved. UDSM achieves the best RI results on 27 data sets and achieves the best NMI results on 29 data sets of the 36. The average improvement in terms of RI compared with the nine baselines is 20.85% (greatest 28.47%, least 16.62%), while the average improvement in terms of NMI is 109.15% (greatest 155.30%, least 91.59%). This verifies that the proposed UDSM is able to select discriminative shapelets as features with good clustering results. We attribute the significant improvement to the following reasons. First, UDSM learns the pseudo class label indicators and the best shapelets simultaneously, which enables UDSM to learn the most discriminative features in a supervised learning way. Second, the spectral structure of the data and the similarity of the learned shapelets are explored simultaneously, which means that similar time series share the same pseudo-class label and the learned shapelets are devised, thus the clustering performance is improved. Third, the shapelets are learned

from the original time series rather than selected from the candidate segments which avoids redundant noise.

Fig. 4.3 shows an illustration of UDSM on four data sets: Coffee, DiatomSizeReduction, ECG200 and SonyAIBORobotSurfaceII, due to space limitations. For each data set, we illustrate two learned shapelets first, then show the mapping of the original data in the two-dimensional shapelet-based space. We see UDSM can extract the most significant shapelets for clustering from the original data which makes the time series distinguishable from each other. For all four data sets, after mapping the original time series to the shapelet-transformed space, clear pseudo classification boundaries are obvious. The dots in red circles and blue squares are the cluster labels obtained by UDSM, which closely match the original class labels.

4.5.5 Time series with unequal length

In this section, we show the generalizability of the proposed algorithm using a synthetic data set which is generated by following the work of [114] and [155].

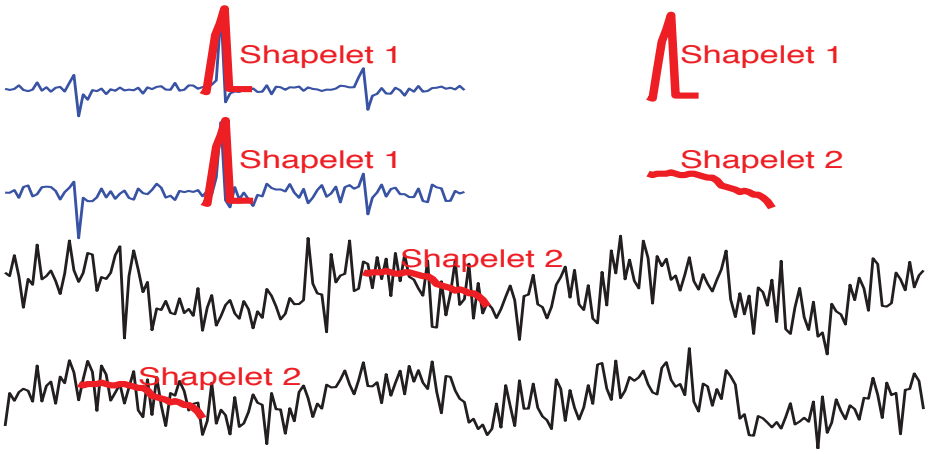
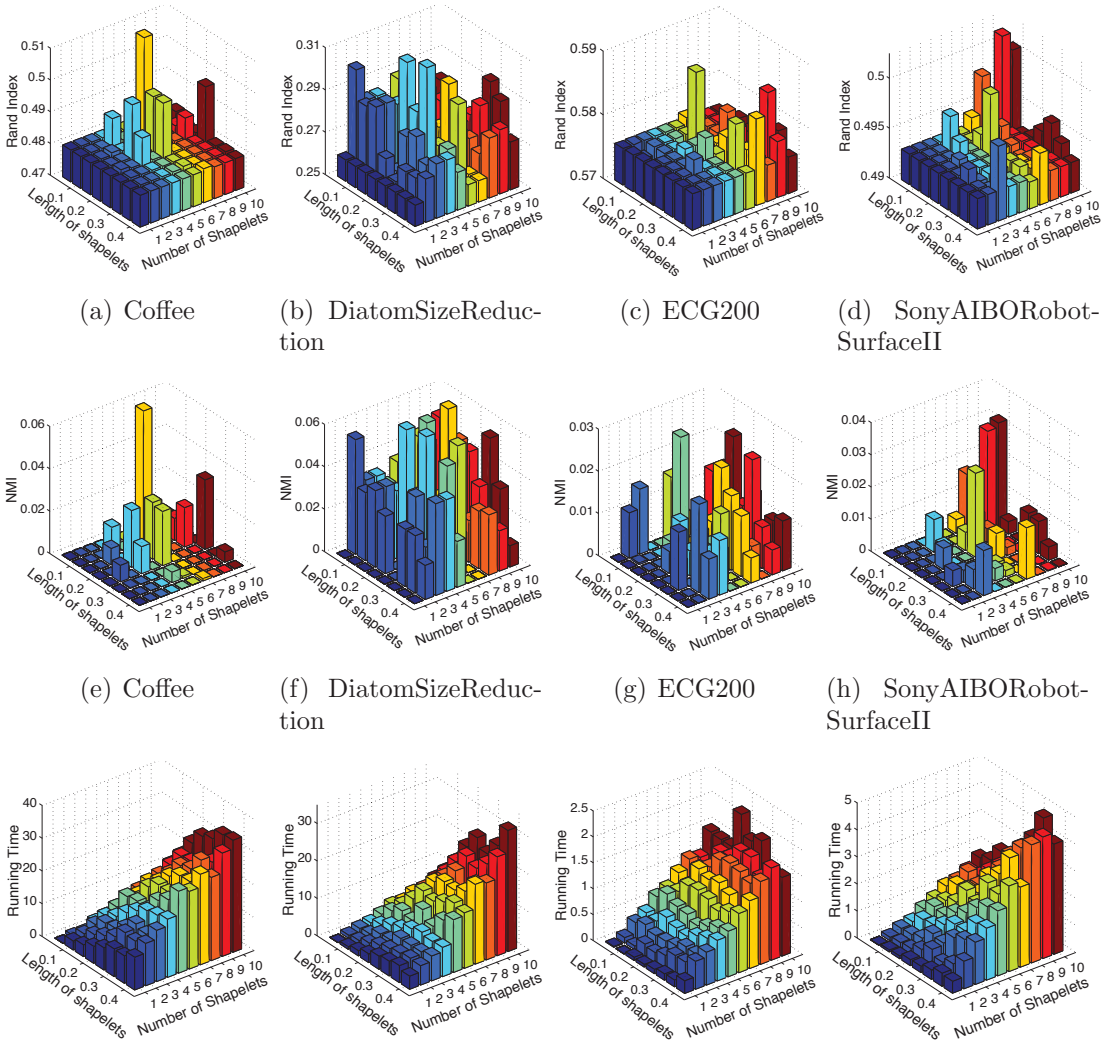


Figure 4.4: An example of the generalizability of UDSM on sinusoidal signals and rectangular signals. The top left blue curves are rectangular signals with Gaussian noise of length 100 while the bottom black curves are sinusoidal signals with Gaussian noise of length 200. The top right short curves marked in bold red are two learned shapelets of lengths 12 and 30 respectively. We see that UDSM can generally handle both the time series and shapelets of different length. The RI and NMI obtained by UDSM for this data set are both 1, which means perfect clustering.

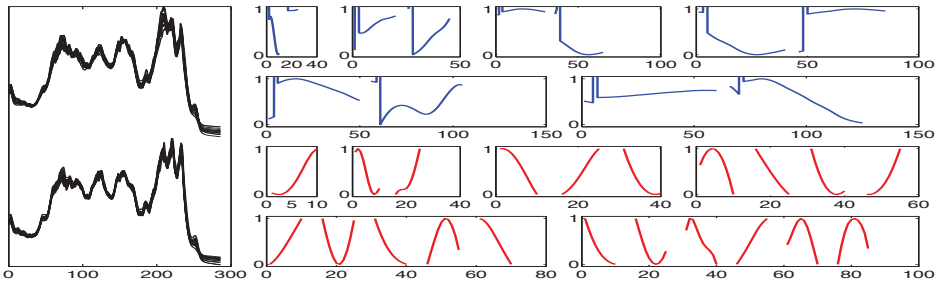


(a) Coffee (b) DiatomSizeReduction (c) ECG200 (d) SonyAIBORobotSurfaceII

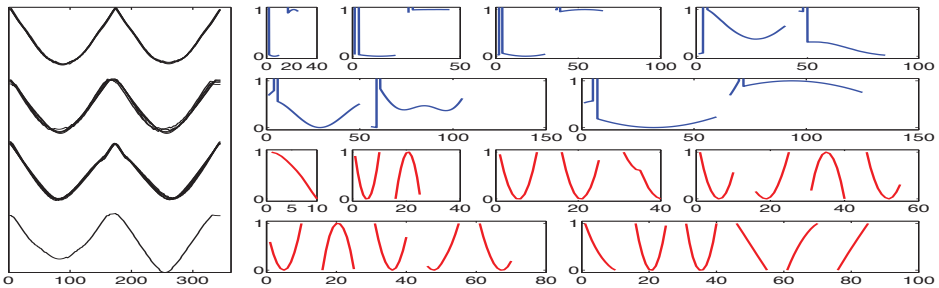
(e) Coffee (f) DiatomSizeReduction (g) ECG200 (h) SonyAIBORobotSurfaceII

(i) Coffee (j) DiatomSizeReduction (k) ECG200 (l) SonyAIBORobotSurfaceII

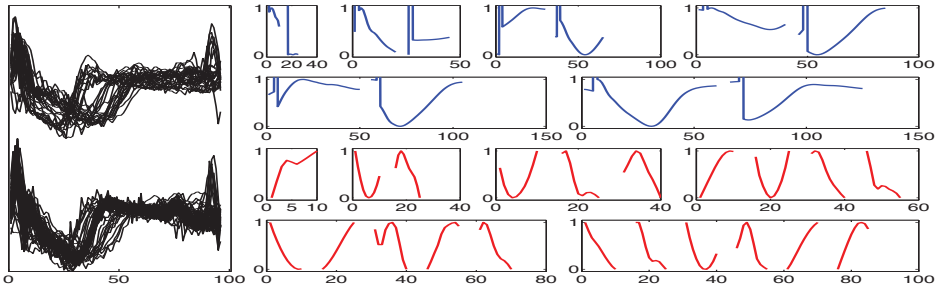
Figure 4.5: The performance of UDSM with respect to the number and the length of the learned shapelets (measured by the Rand index, the NMI and the running time). In terms of RI and NMI, UDSM reaches its best performance at the top-middle or top-right corners. And the running time continuously increases with an increase in the number and the length of the shapelets. This shows that the UDSM does not need to learn long shapelets since short shapelets achieve sufficient or better clustering performance and also save time. In terms of the number of shapelets, it is not necessary to set a large value. In most of the data sets too many shapelets would lead to repeating or overlapping. For the time series sets with complex lines, like SonyAIBORobotSurfaceII, a reasonably large number of shapelets can be set to learn the numerous distinct shapes of the original time series. In summary, the length and the number of shapelets, in practice, does not need to be large values.



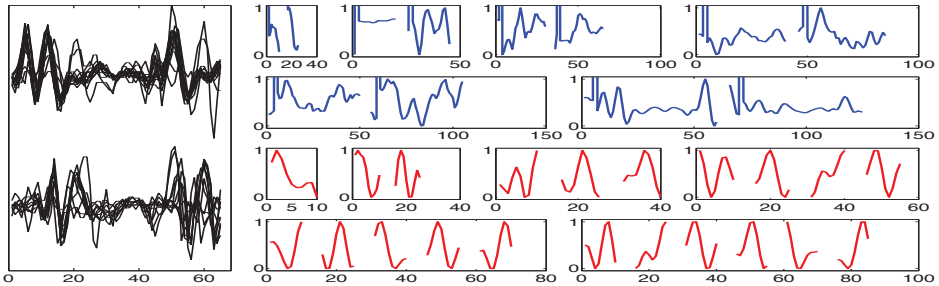
(a) Coffee



(b) DiatomSizeReduction



(c) ECG200



(d) SonyAIBORobotSurfaceII

Figure 4.6: An illustration of the shapelets learned by UDSM on the four real world data sets. The left shows time series examples of the original data sets. Time series from the same class are shown together. The right shows the learned shapelets. Shapelets in blue are learned by increasing the length from 10 to 60 with a step 10. Shapelets in red are learned by increasing the number from 1 to 6 with a step of 1. We see when increasing the length of shapelets, there is a heavy overlap in learned shapelets. When varying the shapelet number, the learned shapelets change but they still overlap. These observations confirm that in practice the length and the number of shapelets does not need to be large values.

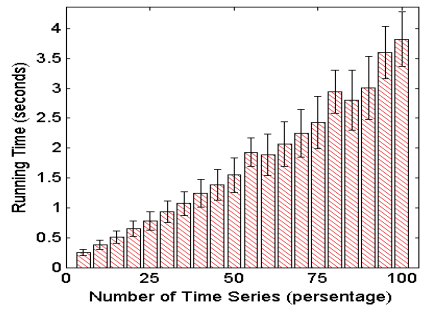
It consists of ten examples from two classes. The first class contains sinusoidal signals of length 200. The second class contains rectangular signals of length 100. We randomly embed Gaussian noise in the data. Heterogeneous noise is generated from five Gaussian processes with a mean of $\nu \in [0, 2]$ and a variance of $\sigma^2 \in [0, 10]$ chosen uniformly at random.

Fig. 4.4 shows an example of two shapelets learned by UDSM. *Shapelet 1* is a sharp spike of length 12 which matches the most prominent spikes of the rectangular signal. *Shapelet 2* is a subsequence with a length of 30, which is very similar to the standard shape of a sinusoidal signal. From the results, we observe that UDSM can automatically-learn representative shapelets from unlabelled time series data. The results in Fig. 4.4 show the great generalizability of UDSM by handling both time series and shapelets of unequal length. In addition, UDSM produces the highest RI and NMI value of 1 on this data set, which equate to perfect clustering result. By contrast, the work in [114] reaches 0.9 even when using class label information during training.

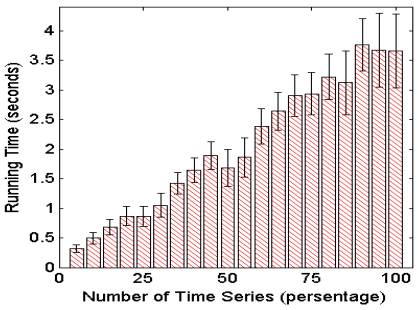
4.5.6 Parameter study

In this section, we test two key parameters in UDSM, i.e. the number of shapelets k and the minimum length of shapelets l_{min}. We varies k from 1 to 10 with a step of 1 and l_{min} from 0.05 to 0.4 of the length of original time series with a step of 0.05. The parameter r is set to 1 for simplicity, which means in this experiment we only learn shapelets of the same length (l_{min}).

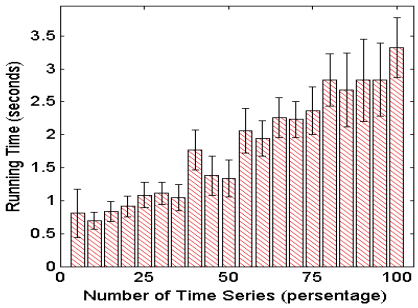
Fig. 4.5 shows the variations in the RI, NMI and running time with respect to the different lengths and different number of shapelets learned on the four data sets. We observe that the RI and NMI values reach their peak at the top-middle area on the first three data sets, and at the top-right corner on the SonyAIBORobotSurfaceII data set. The running time continuously increases with an increase in the number and length of the shapelets. This shows that UDSM does not need to learn long shapelets: short shapelets achieve sufficient or better clustering performance and also save time. It is also not necessary to learn too many shapelets, as in most data sets this leads to repeating or overlapping. For time series sets with complex lines, like SonyAIBORobotSurfaceII, a reasonably



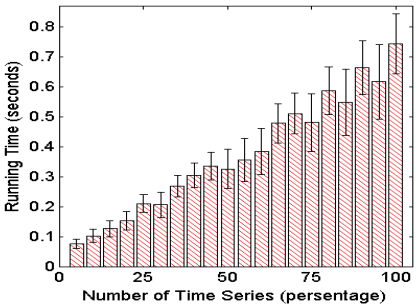
(a) Coffee



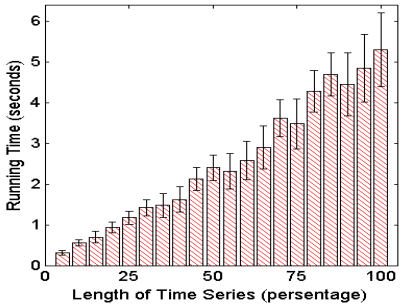
(b) DiatomSizeReduction



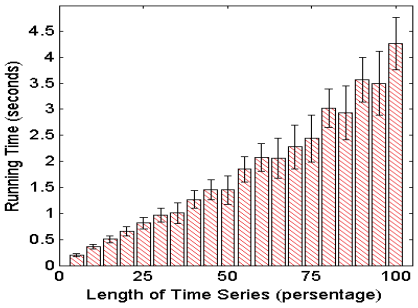
(c) ECG200



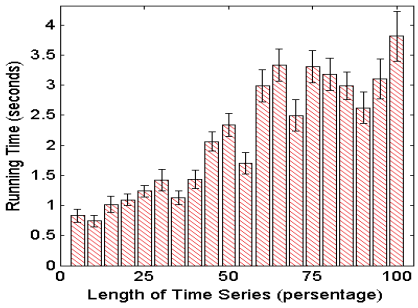
(d) SonyAIBORobotSurfaceII



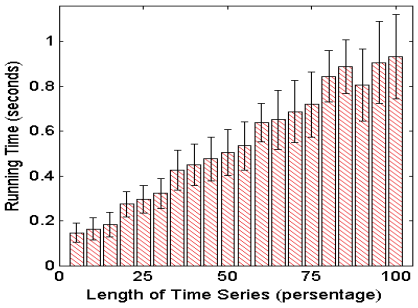
(e) Coffee



(f) DiatomSizeReduction



(g) ECG200



(h) SonyAIBORobotSurfaceII

Figure 4.7: Running time with respect to the length and number of time series.

large number of shapelets can be set to learn the numerous distinct shapes of the original time series.

Fig. 4.6 illustrates the shapelets learned by UDSM on the four real-world data sets. The length of shapelets is varied from 10 to 60 with a step of 10 while the number of shapelets is fixed at 2 at first. The learned shapelets are shown in blue. Subsequently, we vary the number of shapelets from 1 to 6 and fix the length of each shapelet to 10. These learned shapelets are shown in red. The results show that when increasing the length of shapelets, there is a heavy overlap in learned shapelets. When varying the shapelet number, the learned shapelets change but they still overlap. These observations confirm the previous conclusions that, in practice, the length and the number of shapelets does not need to be large values.

4.5.7 Running time and convergence curve

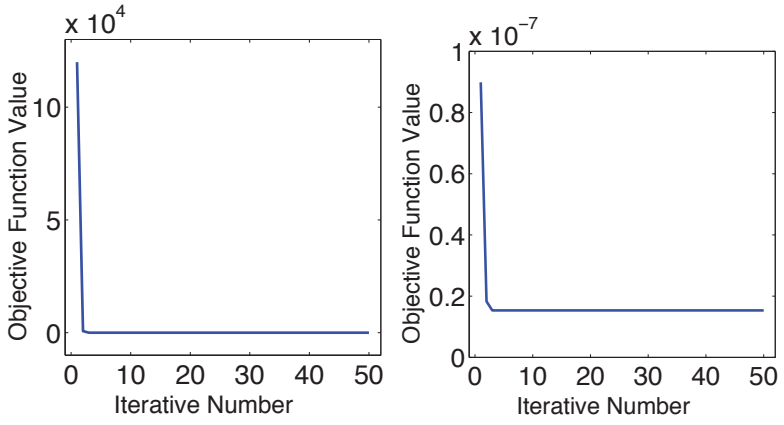
This section explores the variation in running time with respect to the length of the original time series, the size of the data sets and the convergence curves of the UDSM.

In the first experiment, we vary the length of training time series from 5% to 100% of the original ones with a step size of 5%. In the second experiment, we varies the number of training time series from 5% to 100% of the total number of time series in the data set with a step of 5%. The remaining parameters are fixed. The reported running time is an average of ten repeated experiments. Fig. 4.7 shows the experimental results. We observe that the running time increases approximately linearly with respect to both the length of training time series and the size of the data set, which means it scales well to large data sets.

In the third experiment, we explore the convergence curves of UDSM. As shown in Algorithm 2, the algorithm is an iterative process. The convergence curves of the iterative process in UDSM are presented in Fig. 4.8 over the four data sets. From these figures, we can see that the proposed algorithm UDSM is effective and converges quickly.

4.6 Conclusions

In this chapter we propose a new optimization model for unsupervised discriminative subsequence mining for time series by integrating the strengths of pseudo-class labels, spectral analysis, shapelet regularization, and regularized least-squares minimization that can automatically-learn the most discriminative shapelets from unlabelled times series data. Experiments on both synthetic and real-world data sets show that the UDSM model can learn meaningful shapelets with promising performance results compared to state-of-the-art unsupervised time series learn-



(a) Coffee (b) DiatomSizeReduction

(c) ECG200 (d) SonyAIBORobotSurfaceII

Figure 4.8: Convergence curves of UDSM over Coffee, DiatomSizeReduction, ECG200 and SonyAIBORobotSurfaceII data sets.

ing models.

Part III

Online Learning with double-streaming data

Chapter 5

Online Learning from Trapezoidal Data Streams

5.1 Introduction

Recently we have witnessed an increasing number of applications on doubly-streaming data where both data volume and data dimensions increase with time. For example, in graph node classification, both the number of graph nodes and the node features (e.g., the ego-network structure of a social network node) often change dynamically. In text classification and clustering, both the number of documents and text vocabulary increase over time, such as the *infinite vocabulary topic model* [157] to allow the addition, invention and increased prominence of new terms to be captured. Fig. 5.1 gives an example of doubly-streaming text data where both new documents and new text vocabulary arrive over time.

We refer to the above doubly-streaming data as *trapezoidal data streams* where data dynamically change in both volume and feature dimension. The problem of learning from trapezoidal data streams is much more difficult than existing data stream mining and online learning problems [159, 160]. The main challenge of learning from trapezoidal data streams is how to design highly dynamic classifiers that can learn from increasing training data with an expanding feature space. Obviously, existing online learning [68, 70], online feature selection [132] and streaming feature selection algorithms [136] cannot be directly used to handle

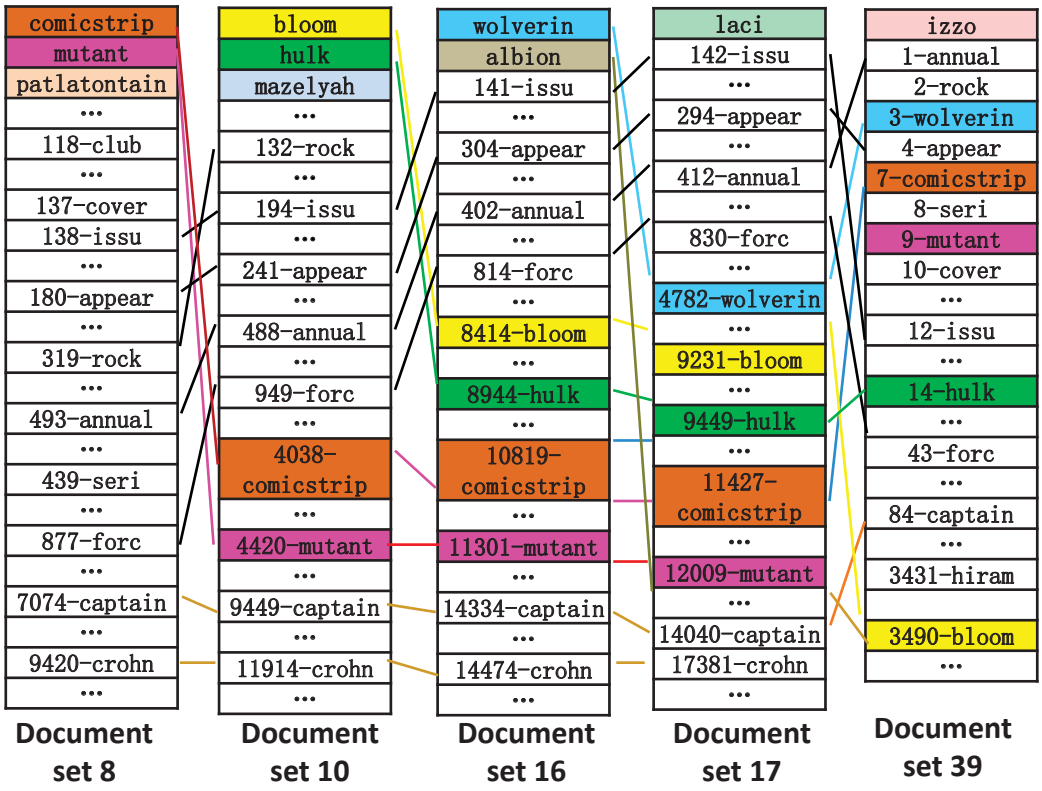


Figure 5.1: Each column is a document set. We observe document sets continuously arrive as a stream. In each column, words in colored boxes are new words introduced by document sets and the number associates with each word is the importance rank for classification. For example, in document set 16, the word "wolverin" in the blue box was first observed and then became one of the most important words for classification in Document set 39.

the problem because they are not designed to deal with the simultaneous change of data volume and data dimension.

Online learning algorithms [70] were proposed to solve the problem where training instances arrive one by one but the feature space is fixed and known a prior before learning. The algorithms update classifiers using incoming instances and allow the sum of training loss gradually to be bounded [70]. To date, online learning algorithms, such as the Perceptron algorithm [107], the Passive Aggressive algorithm [19] and the Confidence-Weighted algorithm [20], are commonly used in data-driven optimizations, but cannot be directly used to handle a dynamic feature space.

Online feature selection algorithms [70, 132] were proposed to perform feature

selection in data streams where data arrive sequentially with a fixed feature space. Online feature selectors are only allowed to maintain a small number of active features for learning [132]. These algorithms use sparse strategies, such as feature truncation, to select representative features. Sparse online learning via truncated gradient [70] and the OFS algorithm [132] are typical algorithms. However, these algorithms cannot solve the trapezoidal data stream mining problem because they assume the feature space is fixed.

Online streaming feature selection algorithms [136] were proposed to select features in a dynamic feature space where features arrive continuously as streams. Each new feature is processed upon its arrival and the goal is to select a "best so far" set of features to train an efficient learning model. It, in some ways, can be seen as the dual problem of online learning [136]. Typical algorithms include the online streaming feature selection (OSFS) algorithm [137] and the fast-OSFS [136] algorithm. However, these algorithms consider only a fixed training set where the number of training instances is given in advance before learning.

In this chapter, we propose a new Online Learning with Streaming Features (OL_{SF}) algorithm and its two variants OL_{SF}-I and OL_{SF}-II for mining trapezoidal data streams. OL_{SF} and its variants combine online learning and streaming feature selection to continuously learn from trapezoidal data streams. Specifically, when new training instances carrying new features arrive, a classifier updates existing features by following the passive-aggressive update rule used in online learning and updates the new features by following the structural risk minimization principle. Then, feature sparsity is introduced by using feature projected truncation. Theoretical and empirical studies validate the performance of the proposed algorithms. The **contributions** of the chapter are summarized as follows:

1. We study a new problem of learning from trapezoidal data streams where training data change in both data volume and feature space;
2. We propose a new learning algorithm OL_{SF} and its two variants. OL_{SF} combines the merits of online learning and streaming feature selection methods to learn from doubly-streaming data;
3. We theoretically analyze the performance bounds of the proposed algo-

rithms;

4. We empirically validate the performance of the algorithms extensively on 14 real-world data sets.

The remainder of the chapter is organized as follows: Section 2 introduces the setting of the learning problem. Section 3 discusses the proposed OL_{SF} algorithm and its variants. Section 4 analyzes the performance bounds. Section 5 conducts experiments and Section 6 concludes the chapter.

5.2 Problem setting

We consider the binary classification problem on trapezoidal data streams. Let $\{(x_t, y_t)|t = 1, \ldots, T\}$ be a sequence of input training data. Each $x_t \in \mathbb{R}^{d_t}$ is a d_t dimension vector where $d_{t-1} \leq d_t$ and class label $y_t \in \{-1, +1\}$ for all t. At each round, the classifier uses information on the current instance to predict its label to be either $+1$ or -1. After the prediction is made, the true label of the instance is revealed and the algorithm suffers an instantaneous loss which reflects the degree of infelicity of the prediction [19]. At the end of each round, the algorithm uses the newly obtained instance-label pair to improve its prediction rule for the rounds to come.

We restrict the discussion to a linear classifier based on a vector of weights w which is the common setting in online learning. The magnitude $|w \cdot x|$ is interpreted as the degree of confidence in the prediction. $w_t \in \mathbb{R}^{d_{t-1}}$ denotes the classifier, i.e., the vector we aim to solve in the algorithm at round t. w_t has the same dimension of the instance x_{t-1}, and has either the same or less dimension as the current instance x_t, for all $t = 2, \ldots, T$, and w_1 is initialized with the same dimension of x_1. For the loss function, we choose the hinge loss. Specifically, $l(w, (x_t, y_t)) = \max\{0, 1 - y_t(w \cdot x_t)\}$, where w and x_t are in the same dimension. In our study, the ultimate dimension d_T is very large, so we also introduce feature selection into our learning algorithm. Table 5.1 demonstrates the symbols and notations used in this chapter.

Table 5.1: Symbols and Notations.

Symbol	Description
B	$B \in [0, 1]$, proportion of selected features (projected feature space)
C	$C > 0$, tradeoff in the objective function of OL$_{SF}$-I and OL$_{SF}$-II
$d_t, t = 1, \ldots, T$	$d_t \leq d_{t+1}$, dimension of instance x_t
$d_{w_t}, t = 1, \ldots, T$	dimension of classifier w_t
λ	$\lambda > 0$, regularization parameter
$l_t, t = 1, \ldots, T$	$l_t = l(w, (x_t, y_t))$, hinge loss on instance (x_t, y_t)
$l_t^*, t = 1, \ldots, T$	$l_t^* = l(\Pi_{x_t} u; (x_t, y_t))$, hinge loss on instance (x_t, y_t) based on the classifier $u \in \mathbb{R}^{d_t}$
L_T	$L_T = \sqrt{\sum_{t=1}^{T} l_t^2}$
M	the number of false predictions by OL$_{SF}$-I in Theorem 3
$\nabla_t, t = 1, \ldots, T-1$	$\nabla_t = \|w_t - \Pi_{w_t} u\|^2 - \|w_{t+1} - \Pi_{w_{t+1}} u\|^2$
R	upper bound for the $L1$-norm of $x_t, t = 1, \ldots, T$
T	$T \in N^+$, total number of instances
u	$u \in \mathbb{R}^{d_T}$, arbitrary vector in $\mathbb{R}^{d_T}$
U_T	$U_T = \sqrt{\sum_{t=1}^{T} (l_t^*)^2}$
$w_t, t = 1, \ldots, T$	$w_t \in \mathbb{R}^{d_{t-1}}, t = 2, \ldots, T, w_1 \in \mathbb{R}^{d_1}$, classifier built at round t
$\|w_t \cdot x_t\|, t = 1, \ldots, T$	confidence degree of x_t with respect to classifier w_t
$\tilde{w}_{t+1}, t = 1, \ldots, T-1$	$\tilde{w}_{t+1} = \Pi_{w_t} w_{t+1}$, vector of elements w_{t+1} projected to feature space w_t
$\hat{w}_{t+1}, t = 1, \ldots, T-1$	$\hat{w}_{t+1} = \Pi_{\neg w_t} w_{t+1}$, vector of elements w_{t+1} not projected to the feature space w_t
$\bar{w}_{t+1}, t = 1, \ldots, T-1$	intermediate variable of new classifier after the update operation
$\check{w}_{t+1}, t = 1, \ldots, T-1$	intermediate variable of new classifier on the L_1 ball without truncation
$\Pi_{w_{t+1}/w_t} u, t = 1, \ldots, T-1$	vector of elements u projected to the feature space of w_{t+1} but not to w_t
$x_t, t = 1, \ldots, T$	$x_t \in \mathbb{R}^{d_t}$, input training instance at time t in d_t dimensions
$\tilde{x}_t, t = 2, \ldots, T$	$\tilde{x}_t = \Pi_{w_t} x_t, x_t \in \mathbb{R}^{d_t}, w_t \in \mathbb{R}^{d_{t-1}}, d_{t-1} \leq d_t$, vector of elements x_t projected to the feature space of w_t
$\hat{x}_t, t = 2, \ldots, T$	$\hat{x}_t = \Pi_{\neg w_t} x_t, x_t \in \mathbb{R}^{d_t}, w_t \in \mathbb{R}^{d_{t-1}}, d_{t-1} \leq d_t$, vector of elements x_t not projected to the feature space of w_t
$\{(x_t, y_t) \mid t = 1, 2, \ldots, T\}$	sequence of input training data
ξ	slack variable
$y_t, t = 1, \ldots, T$	$y_t \in \{-1, +1\}$, real label of instance x_t
$\hat{y}_t, t = 1, \ldots, T$	$\hat{y}_t = sign(w_t \cdot \Pi_{w_t} x_t)$, predicted label of instance x_t
$\tau_t, t = 1, \ldots, T$	learning rate variable

5.3 Online learning with trapezoidal data streams

In this section we present the Online Learning with Streaming Features algorithm (OL_{SF}) and its two variants for mining trapezoidal data streams. There are two challenges to be addressed by the algorithms. The first challenge is to update the classifier with an augmenting feature space. The classifier update strategy is able to learn from new features. We build the update strategy based on the margin-maximum principle. The second challenge is to build a feature selection method to achieve a sparse but efficient model. As the dimension increases over time, it is essential to use feature selection to prune redundant features. We use a truncation strategy to obtain sparsity. Also, in order to improve the truncation, a projection step is introduced before the truncation.

The pseudo-codes for the OL_{SF} algorithm and its two variants are given in Algorithms 1, 2 and 3 (OL_{SF}-I and OL_{SF}-II are different to OL_{SF} in parameter τ_t during updates). The vector w_1 is initialized to a zero vector with dimension d_1, i.e., $w_1 = (0, \ldots, 0) \in \mathbb{R}^{d_1}$ for all the three algorithms, where d_1 is the dimension of the first instance for each algorithm. Then, online learning is divided into the update step and the sparsity step.

The update strategy

The three algorithms are different in their update strategy. We first focus on the update strategy of the basic algorithm. At round t, with the classifier $w_t \in \mathbb{R}^{d_{t-1}}$, the new classifier $w_{t+1} = [\tilde{w}_{t+1}, \hat{w}_{t+1}] \in \mathbb{R}^{d_t}$ is obtained as the solution to the constrained optimization problem in Eq.(2), where $\tilde{w} = \Pi_{w_t} w_{t+1} \in \mathbb{R}^{d_{t-1}}$ represents a projection of the feature space from dimension d_t to dimension d_{t-1}, it is a vector consisting of elements of w_{t+1} which are in the same feature space of w_t, and $\hat{w} = \Pi_{\neg w_t} w_{t+1} \in \mathbb{R}^{d_t - d_{t_1}}$ denotes the vector consisting of elements of w_{t+1} which are not in the feature space of w_t,

$$\begin{aligned} w_{t+1} &= [\tilde{w}_{t+1}, \hat{w}_{t+1}] \\ &= \underset{\substack{w = [\tilde{w}, \hat{w}]: \\ l_t = 0}}{\text{argmin}} \quad \frac{1}{2}\|\tilde{w} - w_t\|^2 + \frac{1}{2}\|\hat{w}\|^2 \end{aligned} \tag{5.1}$$

Algorithm 3 The OL_{SF} algorithm and its two variants OL_{SF}-I and OL_{SF}-II

Require:

$C > 0$: the tradeoff parameter of OL_{SF}-I and OL_{SF}-II

$\lambda > 0$: the regularization parameter

$B \in (0, 1]$: the proportion of selected features

Ensure:

W^*: the optimal classifier

Initialize $w_1 = (0, \ldots, 0) \in \mathbb{R}^{d_1}$

For $t = 1, 2, \ldots$ **do**

  receive instance: $x_t \in \mathbb{R}^{d_t}$

  predict: $\hat{y}_t = sign(w_t \cdot \Pi_{w_t} x_t)$

  receive correct label: $y_t \in \{+1, -1\}$

  suffer loss: $l_t = max\{0, 1 - y_t(w_t \cdot \Pi_{w_t} x_t)\}$

  update step:

    set parameter :

      $\tau_t = Parameter\_Set(x_t, l_t, C)$ (See Algorithm 4)

    update w_t to $\bar{w}_{t+1}$:

      $\bar{w}_{t+1} = [w_t + \tau_t y_t \Pi_{w_t} x_t, \tau_t y_t \Pi_{\neg w_t} x_t]$

  sparsity step:

    project $\bar{w}_{t+1}$ to a L_1 ball:

      $\check{w}_{t+1} = \min\{1, \frac{\lambda}{\|\bar{w}_{t+1}\|_1}\}\bar{w}_{t+1}$

    truncate $\check{w}_{t+1}$ to w_{t+1}:

      $w_{t+1} = Truncate(\check{w}_{t+1}, B)$ (See Algorithm 5)

endfor

return $W^* = W_{t+1}$;

Algorithm 4 Update strategy: $\tau_t = Parameter\_Set(x_t, l_t, C)$

if OL_{SF}:

  $\tau_t = \frac{l_t}{\|x_t\|^2}$

else if OL_{SF}-I:

  $\tau_t = min\{C, \frac{l_t}{\|x_t\|^2}\}$

else if OL_{SF}-II:

  $\tau_t = \frac{l_t}{\|x_t\|^2 + \frac{1}{2C}}$

end if

where $l_t = l(w, (x_t, y_t))$ is the loss at round t , which can be written as,

$$l_t = l(w, (x_t, y_t)) = \max\{0, 1 - y_t(\tilde{w} \cdot \tilde{x}_t) - y_t(\hat{w} \cdot \hat{x}_t)\}. \tag{5.2}$$

Algorithm 5 Sparsity strategy: $w = Truncate(\check{w}, B)$

$\check{w} \in \mathbb{R}^{d_{\check{w}}}$
if $\|\check{w}\|_0 \geq B \cdot d_{\check{w}}$ **then**
    $w = \check{w}^B$; $\check{w}^B$ is a vector that remains the $\max\{1, floor(B \cdot d_{\check{w}})\}$ largest elements of $\check{w}$ and set others to zeros, where $floor\{x\}$ is the largest integer smaller then x.
else
    $w = \check{w}$
end if

Note that the definition of $\tilde{x}_t = \Pi_{\tilde{w}} x_t$ and $\hat{x}_t = \Pi_{\hat{w}} x_t$ are similar to the ones of $\tilde{w}$ and $\hat{w}$ respectively.

In the above constrained optimization problem, if the existing classifier w_t predicts the right label with the current instance x_t, i.e., $l_t = \max\{0, 1 - y_t(w_t \cdot \tilde{x}_t)\} = 0$, then we can easily know that the optimal solution is $\tilde{w} = w_t, \hat{w} = (0, \ldots, 0)$, that is, $w_{t+1} = [w_t, 0, \ldots, 0]$.

On the other hand, if the existing classifier makes a wrong prediction, the algorithm forces the updated classifier to satisfy the constraint in Eq. (5.1). At the same time, it also forces $\tilde{w}_{t+1}$ close to w_t in order to inherit information and let $\hat{w}_{t+1}$ be small to minimize structural risk and avoid overfitting. The solution to Eq. (5.1) has a simple closed form,

$$w_{t+1} = [w_t + \tau_t y_t \tilde{x}_t, \tau_t y_t \hat{x}_t], \quad where\ \tau_t = l_t / \|x_t\|^2 \tag{5.3}$$

We now discuss the derivation of the update strategy.

- In case that the dimension of the new classifier does not change, i.e., $d_t = d_{t-1}$, the problem degenerates to an online learning problem where $\hat{w}_{t+1}$ disappears and $w_{t+1} = \tilde{w}_{t+1}$.
- In case that $d_t > d_{t-1}$ and $l_t = 0$, the optimal solution is $\tilde{w}_{t+1} = w_t$ and $\hat{w}_{t+1} = (0, \cdots, 0)$.
- In case that $d_t > d_{t-1}$ and $l_t > 0$, we solve Eq. (5.1) to obtain the solution.

To solve Eq.(5.1), we use the Lagrangian function and the Karush-Khun-

Tucker conditions [8] on Eq.(2) and obtain

$$\begin{aligned} L(w,\tau) &= \frac{1}{2}\|\tilde{w} - w_t\|^2 + \frac{1}{2}\|\hat{w}\|^2 \\ &\quad + \tau(1 - y_t(\tilde{w}\cdot\tilde{x}_t) - y_t(\hat{w}\cdot\hat{x})), \\ \tilde{w} &= w_t + \tau y_t\tilde{x}_t; \qquad \hat{w} = \tau y_t\hat{x}_t \end{aligned} \tag{5.4}$$

where τ is a Lagrange multiplier. Plugging the last two equations into the first one, taking the derivative of $L(\tau)$ with respect to τ and setting it to zero, we can obtain

$$\begin{aligned} L(\tau) &= -\frac{1}{2}\tau^2\|\tilde{x}_t\|^2 - \frac{1}{2}\tau^2\|\hat{x}\|^2 + \tau - \tau y_t(w_t\cdot\tilde{x}) \\ \tau_t &= \frac{1 - y_t(w_t\cdot\tilde{x}_t)}{\|\tilde{x}\|^2 + \|\hat{x}_t\|^2} = \frac{l_t}{\|x_t\|^2} \end{aligned} \tag{5.5}$$

So, the update strategy is $w_{t+1} = [w_t + \tau_t y_t\tilde{x}_t, \tau_t y_t\hat{x}_t]$, where $\tau_t = l_t/\|x_t\|^2$. In addition, this update rule is also applied when $l_t = 0$. So we can take it as a general update rule.

From Eq. (5.1), we can see that the update strategy of the OL$_{SF}$ algorithm is rigorous because the new classifier needs to predict the current instance correctly. This may make the model sensitive to noise, especially label noise [19]. In order to avoid this drawback, we give two general updated variants of the OL$_{SF}$ algorithm which use the soft-margin technique by introducing a slack variable ξ into the optimization problem. The first one is abbreviated as OL$_{SF}$-I. Its objective function scales linearly with ξ, namely,

$$w_{t+1} = \underset{\substack{w=[\tilde{w},\hat{w}]:\\ l_t\le\xi;\xi\ge 0}}{\operatorname{argmin}} \quad \frac{1}{2}\|\tilde{w} - w_t\|^2 + \frac{1}{2}\|\hat{w}\|^2 + C\xi \tag{5.6}$$

The second one, OL$_{SF}$-II, is the same as OL$_{SF}$-I except that its objective function scales quadratically with the slack variable ξ, i.e.,

$$w_{t+1} = \underset{\substack{w=[\tilde{w},\hat{w}]:\\ l_t\le\xi}}{\operatorname{argmin}} \quad \frac{1}{2}\|\tilde{w} - w_t\|^2 + \frac{1}{2}\|\hat{w}\|^2 + C\xi^2 \tag{5.7}$$

In these two optimization problems, parameter C is a positive number which

is a tradeoff between rigidness and slackness. A larger value of C implies a more rigid update step.

The update strategy of OL$_{SF}$-I and OL$_{SF}$-II also shares the simple closed form $w_{t+1} = [w_t + \tau_t y_T \tilde{x}_t, \tau y_t \hat{x}_t]$, where

$$\tau_t = \min\{C, \frac{l_t}{\|x_t\|^2}\}\ (I) \quad or \quad \tau_t = \frac{l_t}{\|x_t\|^2 + \frac{1}{2C}}\ (II).$$

The update strategies of OL$_{SF}$-I and OL$_{SF}$-II are similar to the OL$_{SF}$ algorithm, so we omit their details due to space constraints.

The sparsity strategy

In many applications, the dimension of training instances increases rapidly and we need to select a relatively small number of features.

In our study, we introduce a parameter to control the proportion of the features used. For example, in each trial t, the learner presents a classifier $w_t \in \mathbb{R}^{d_{t-1}}$ to classify instance $x_t \in \mathbb{R}^{d_t}$ where $d_{t-1} \leq d_t$. After the update operation, a projection and a truncation are introduced to prune redundant features based on the parameter B, which locates in $[0, 1]$. Namely, we require the learner only retain at most a proportion of B nonzero elements of $w_t \in \mathbb{R}^{d_{w_t}}$, i.e. $\|w_t\|_0 \leq B \cdot d_{w_t}$. Specifically, if the resulting classifier w_t has more than a proportion of B nonzero elements, we will simply keep the proportion of B elements in w_t with the largest absolute weights, as demonstrated in Algorithm 3. In this way, at most a proportion of B features are used in the model and sparsity is introduced.

We introduce a projection step because one single truncation step does not work well. Although the truncation selects the B largest elements, this does not guarantee the numerical values of the unselected attributes are sufficiently small and may potentially lead to poor performance [132]. When projecting a vector to an L_1 ball, most of its numerical values are concentrated to its largest elements, and then removing the smallest elements will result in a small change to the original vector. Specifically, the projection is,

$$\check{w}_{t+1} = \min\{1, \frac{\lambda}{\|\bar{w}_{t+1}\|_1}\}\bar{w}_{t+1}, \tag{5.8}$$

where λ is the a positive regularization parameter.

5.4 Theoretical analysis

In this section, we derive performance bounds of the OL$_{SF}$ algorithm and its two variants OL$_{SF}$-I and OL$_{SF}$-II. There are four theorems and one lemma in this section. The first theorem discusses the upper bound of the cumulative squared hinge loss of OL$_{SF}$ when data are linearly separable, and the second derives the bound when data are linearly inseparable. The third and the fourth theorems relate to the upper bounds of the OL$_{SF}$-I and OL$_{SF}$-II algorithms respectively.

If instance x_t is falsely predicted, then $y_t(w_t \cdot \Pi_{w_t} x_t) < 0$, and the loss function $l_t > 1$. So the cumulative squared hinge loss $\sum_t l_t^2$ is an upper bound of the number of false predictions [19]. Therefore, the loss bound will be the upper bound of the total number of false predictions and the cumulative squared hinge loss. Our bounds essentially prove that our algorithms cannot do much worse than the best fixed prediction, which is chosen in hindsight for any sequence of instances.

For clarity, we use two abbreviations throughout the chapter. We denote by l_t the instantaneous loss suffered by our algorithm at round t. In addition, we denote by l_t^* the loss of an off-line predictor at round t. Formally, let $u \in \mathbb{R}^{d_T}$ be an arbitrary vector in $\mathbb{R}^{d_T}$, we define l_t and l_t^* as follows,

$$l_t = l(w_t; (\Pi_{w_t} x_t, y_t)) \quad \textit{and} \quad l_t^* = l(\Pi_{x_t} u; (x_t, y_t)) \tag{5.9}$$

Then, we have Lemma 1 as follows.

Lemma 1. Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of training instances, where $x_t \in \mathbb{R}^{d_t}, d_{t-1} \leq d_t$ and $y_t \in \{+1, -1\}$ for all t. Let the learning rate $\tau_t \in \left\{ \frac{l_t}{\|x_t\|^2}, min\{C, \frac{l_t}{\|x_t\|^2}\}, \frac{l_t}{\|x_t\|^2 + \frac{1}{2C}} \right\}$, as given in Algorithm 2. Then, the following bound holds for any $u \in \mathbb{R}^{d_T}$, $\sum_{t=1}^{T} \tau_t (2l_t - \tau_t \|x_t\|^2 - 2l_t^*) \leq \|u\|^2$

Proof. Define ∇_t to be $\|w_t - \Pi_{w_t} u\|^2 - \|w_{t+1} - \Pi_{w_{t+1}} u\|^2$. We prove the lemma by summing up all ∇_t over t in $1, \ldots, T$ and bounding this sum. Note that $\sum_t \nabla_t$

is a telescopic sum which collapses to

$$\begin{aligned}\sum_{t=1}^{T-1}\nabla_t &= \sum_{t=1}^{T-1}(\|w_t - \Pi_{w_t}u\|^2 - \|w_{t+1} - \Pi_{w_{t+1}}u\|^2) \\ &= \|w_1 - \Pi_{w_1}u\|^2 - \|w_T - \Pi_{w_T}u\|^2,\end{aligned} \tag{5.10}$$

where w_1 is initialized as a zero vector, and $\|w_T - \Pi_{w_T}u\|^2 \geq 0$ always holds. Thus, we can upper bound the right-hand side of the above equation by $\|\Pi_{w_1}u\|^2$,

$$\sum_{t=1}^{T-1}\nabla_t \leq \|\Pi_{w_1}u\|^2. \tag{5.11}$$

We now turn to bound every single ∇_t. If the minimum margin requirement is not violated on round t , i.e. l_t =0, then $\tau_t = 0$ and hence $\nabla_t \leq 0$. Now we only focus on rounds on which $l_t > 0$. With the update strategy $\bar{w}_{t+1} = [w_t + \tau_t y_t \Pi_{w_t}x_t, \tau_t y_t \Pi_{\neg w_t}x_t]$ where $\Pi_{\neg w_t}x_t$ is a vector consisting of elements in x which are not in the same feature space of w_t. And in light of the fact that $\check{w}_{t+1} \leq \bar{w}_{t+1}$ and $w_{t+1} \leq \check{w}_{t+1}$ we have

$$\begin{aligned}\nabla_t =& \|w_t - \Pi_{w_t}u\|^2 - \|w_{t+1} - \Pi_{w_{t+1}}u\|^2 \\ \geq& \|w_t - \Pi_{w_t}u\|^2 - \|w_t + \tau_t y_t \Pi_{w_t}x_t - \Pi_{w_t}u\|^2 \\ & - \|\tau_t y_t \Pi_{\neg w_t}x_t - \Pi_{w_{t+1}\ w_t}u\|^2 \\ =& -2\tau_t y_t \Pi_{w_t}x_t(w_t - \Pi_{w_t}u) - \tau_t^2\|\Pi_{w_t}x_t\|^2 \\ & - \|\tau_t y_t \Pi_{\neg w_t}x_t - \Pi_{w_{t+1}/w_t}u\|^2\end{aligned} \tag{5.12}$$

From $l_t = 1 - y_t(w_t \cdot \Pi_{w_t}x_t)$ and $l_t^* \geq 1 - y_t(\Pi_{x_t}u \cdot x_t)$, we have $y_t(w_t \cdot \Pi_{w_t}x_t) = 1 - l_t$ and $y_t(\Pi_{w_t}u \cdot \Pi_{w_t}x_t) + y_T(\Pi_{w_{t+1}/w_t}u \cdot \Pi_{w_{t+1}/w_t}x_t) \geq 1 - l_t^*$. Using these two facts in Eq. (5.12) gives,

$$\begin{aligned}\nabla_t \geq& 2\tau_t(y_t \Pi_{w_t}x_t \Pi_{w_t}u + y_t \Pi_{w_{t+1}/w_t}x_t \Pi_{w_{t+1}/w_t}u \\ & - y_t \Pi_{w_t}x_t w_t) - \tau_t^2\|x_t\|^2 - \|\Pi_{w_{t+1}/w_t}u\|^2 \\ \geq& \tau_t(2l_t - 2l_t^* - \tau_t\|x_t\|^2) - \|\Pi_{w_{t+1}/w_t}u\|^2.\end{aligned} \tag{5.13}$$

Summing up ∇_t over all t and comparing the lower bound of Eq. (5.13) with

the upper bound in Eq.(5.11), we can obtain

$$\sum_{t=1}^{T} \tau_t(2l_t - 2l_t^* - \tau_t\|x_t\|^2) \leq \|\Pi_{w_1} u\|^2 + \sum_{t=1}^{T-1} \|\Pi_{w_{t+1}/w_t} u\|^2.$$

The lemma is proved. □

Below we first prove a loss bound for the OL_{SF} algorithm in the linearly separable case. We assume that there is a classifier $u \in \mathbb{R}^{d_T}$ such that $y_t(\Pi_{x_t} u \cdot x_t) > 0$ for all $t \in \{1, \ldots, T\}$. Without loss of generality, we assume that classifier u is scaled such that $y_t(\Pi_{x_t} u \cdot x_t) \geq 1$. The loss of u is zero on all T instances in the sequence. Then, we have the following bound of the cumulative squared loss of OL_{SF}.

Theorem 1. *Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of instances where $x_t \in \mathbb{R}^{d_t}$, $d_{t-1} \leq d_t$, $y_t \in \{+1, -1\}$ and $\|x_t\| \leq R$ for all t. Assume that there exists a classifier u such that $l_t^* = 0$ for all t. Then, the cumulative squared loss of OL_{SF} on the sequence is bounded by*

$$\sum_{t=1}^{T} l_t^2 \leq \|u\|^2 R^2.$$

Proof. Since $l_t^* = 0$ for all t , Lemma 1 implies that,

$$\sum_{t=1}^{T} \tau_t(2l_t - \tau_t\|x_t\|^2) \leq \|u\|^2. \tag{5.14}$$

According to the definition $\tau_t = \frac{l_t}{\|x_t\|^2}$, we have

$$\sum_{t=1}^{T} \frac{l_t^2}{\|x_t\|^2} \leq \|u\|^2$$

and

$$\sum_{t=1}^{T} l_t^2 \leq \|u\|^2 R^2.$$

Hence, the theorem is proved. □

The following theorems generalize the linearly separable case. We consider that the classifier u cannot perfectly separate the training data. In addition, we assume that the input sequence is normalized so that $\|x_t\|^2 = 1$. Then, we have the following bounds of the cumulative squared loss of the OL_{SF} algorithm.

Theorem 2. *Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of instances where $x_t \in \mathbb{R}^{d_t}$, $d_{t-1} \leq d_t$, $y_t \in \{+1, -1\}$ and $\|x_t\|^2 = 1$ for all t. Then, for any vector $u \in \mathbb{R}^{d_T}$, the cumulative squared loss of OL_{SF} on the sequence is bounded by*

$$\sum_{t=1}^{T} l_t^2 \leq \left(\|u\| + 2\sqrt{\sum_{t=1}^{T} (l_t^*)^2} \right)^2 . \tag{5.15}$$

Proof. Since $\|x_t\|^2 = 1$, τ_t and l_t are equal, according to Lemma 1, we have $\sum_{t=1}^{T} l_t^2 \leq \|u\|^2 + \sum_{t+1}^{T} 2 l_t \cdot l_t^*$. Denote

$$L_T = \sqrt{\sum_{t=1}^{T} l_t^2} \quad \textit{and} \quad U_T = \sqrt{\sum_{t=1}^{T} (l_t^*)^2}.$$

By using the Cauchy-Schwartz inequality to bound the right-hand side of Eq.(5.15), we obtain

$$L_T^2 \leq \|u\|^2 + 2 L_T U_T.$$

Therefore, to obtain an upper bound of L_T, we need to find the largest solution of $L_T^2 - 2U_T L_T - \|u\|^2 = 0$, i.e.,

$$U_T + \sqrt{U_T^2 + \|u\|^2}.$$

Using the fact that $\sqrt{\alpha + \beta} \leq \sqrt{\alpha} + \sqrt{\beta}$, we have

$$L_T \leq \|u\| + 2U_T.$$

Furthermore, we can obtain

$$\sum_{t=1}^{T} l_t^2 \leq \left(\|u\| + 2\sqrt{\sum_{t=1}^{T} (l_t^*)^2} \right)^2$$

and the theorem is proved. □

Next we derive the bound for OL$_{SF}$-I. The following theorem provides an error rate bound of OL$_{SF}$-I based on the total number of falsely predicted instances that $y_t \neq sign(w_t \cdot \Pi_{w_t} x_t)$.

Theorem 3. *Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of instances, where $x_t \in \mathbb{R}^{d_t}$, $d_{t-1} \leq d_t$, $y_t \in \{+1, -1\}$ and $\|x_t\|^2 \leq R^2$ for all t. For any vector $u \in \mathbb{R}^{d_T}$, the number of false predictions by OL$_{SF}$-I is bounded by,*

$$\max\{R^2, \frac{1}{C}\} \left(\|u\|^2 + 2C \sum_{t=1}^{T} l_t^* \right),$$

where C is the parameter in OL$_{SF}$-I.

Proof. If OL$_{SF}$-I outputs a false prediction at round t, then $y_t(w_t \cdot \Pi_{w_t} x_t) \leq 0$, so $l_t \geq 1$. Under the assumption $\|x_t\|^2 \leq R^2$ and the definition $\tau_t = min\{l_t / \|x_t\|^2, C\}$, for the error occurring at round t, we have

$$\min\{\frac{1}{R^2}, C\} M \leq \sum_{t=1}^{T} \tau_t l_t,$$

where M is the number of false predictions by OL$_{SF}$-I.

Based on the definition of τ_t , we know that $\tau_t l_t^* \leq C l_t^*$ and $\tau_t \|x_t\|^2 \leq l_t$. Plugging these two inequalities into Lemma 1 gives the result,

$$\sum_{t=1}^{T} \tau_t l_t \leq \|u\|^2 + 2C \sum_{t=1}^{T} l_t^*.$$

Combining the above two inequations, we obtain that

$$\min\{1/R^2, C\}M \leq \|u\|^2 + 2C\sum_{t=1}^{T} l_t^*.$$

The theorem is proved by multiplying both sides of the above inequation with $\max\{R^2, 1/C\}$. □

Now, we turn to the bound analysis for OL$_{SF}$-II.

Theorem 4. *Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of instances where $x_t \in \mathbb{R}^{d_t}$, $d_{t-1} \leq d_t$, $y_t \in \{+1, -1\}$ and $\|x_t\|^2 \leq R$ for all t . Then, for any classifier (vector) $u \in \mathbb{R}^{d_T}$, the cumulative squared loss of OL$_{SF}$-II is bounded by,*

$$\sum_{t=1}^{T} l_t^2 \leq \left(R^2 + \frac{1}{2C}\right)\left(\|u\|^2 + 2C\sum_{t=1}^{T}(l_t^*)^2\right).$$

Proof. Lemma 1 states that

$$\|u\|^2 \geq \sum_{t=1}^{T}(2\tau_t l_t - \tau^2\|x_t\|^2 - 2\tau_t l_t^*).$$

Define $\alpha = 1/\sqrt{2C}$, and by subtracting the non-negative term $(\alpha\tau_t - l_t^*/\alpha)^2$ from each result on the right-hand side of the above inequality, we can obtain

$$\begin{aligned}\|u\|^2 &\geq \sum_{t=1}^{T}(2\tau_t l_t - \tau^2\|x_t\|^3 - 2\tau_t l_t^* - (\alpha\tau_t - l_t^*/\alpha)^2) \\ &= \sum_{t=1}^{T}(2\tau_t l_t - \tau^2(\|x_t\|^2 + \alpha^2) - (l_t^*)^2/\alpha^2).\end{aligned} \tag{5.16}$$

Plugging in the definition of α, and using the definition $\tau_t = l_t/(\|x_t\|^2 + 1/(2C))$, we can obtain the following lower bound,

$$\|u\|^2 \geq \sum_{t=1}^{T}\left(\frac{l_t^2}{\|x_t\|^2 + \frac{1}{2C}} - 2C(l_t^*)^2\right).$$

Replacing $\|x_t\|^2$ with its upper bound of R^2 and rearranging the terms gives

the desired bound. □

5.5 Experiments

In this section, we empirically evaluate the performance of OL$_{SF}$ and its two variants OL$_{SF}$-I and OL$_{SF}$-II[1].

The experiments are conducted from four aspects. Firstly, we evaluate the performance of the proposed three algorithms with respect to classification accuracy, projected feature space B, and tradeoff C in Section 6.1. Secondly, we evaluate the update strategy and the sparse strategy used in the three algorithms by comparing with three benchmark methods in Section 6.2. Thirdly, we compare the proposed algorithms with the state-of-the-art online feature selection algorithms in Section 6.3. Finally, we test the applications of the proposed algorithms on two real-world trapezoidal data streams in Section 6.4.

Experimental setup. We test on 12 UCI data sets and two real-world large-scale streams as listed in Table 5.2.

To simulate trapezoidal streams, we split the data sets into 10 chunks, where each chunk carries only 10% instances and a variant number of features. For example, the first data chunk carries the first 10% instances with the first 10% features. The second data chunk carries the second 10% instances with another 10% features (in total 20% features).

We measure the performance in terms of the average prediction accuracy. The experiments are repeated 20 times with a random permutation on the data sets. The results are reported by an average over the 20 repeats.

We set λ to be 30, C from 10^{-4} to 10^{4} with a step of 10^{1}, and B from 0 to 1. The parameters are chosen with cross validation.

[1]The Matlab source codes are available online at *https://github.com/BlindReview/onlineLearning*.

Table 5.2: The data sets used in the experiments

Dataset	♯ instances	♯ Dimensions
wpbc	198	34
ionosphere	351	35
wdbc	569	31
isolet	600	618
wbc	699	10
german	1,000	24
svmguide3	1,234	21
splice	3,175	60
HAPT	3,266	562
spambase	4,601	57
magic04	19,020	10
a8a	32,561	123
rcv1	697,641	47,236
URL	2,396,130	3,231,961

5.5.1 Experiment I: Comparisons between OL_{SF} and its two variants

In this part, we present the empirical results of the three algorithms on the 12 UCI benchmark data sets.

Table 5.3 summarizes the performance of the three algorithms on a projected feature space. We can observe that OL_{SF}-I performs the best on six data sets, a8a, german, HAPT, magic04, spambase, wpbc, OL_{SF}-II performs the best on the remaining six date sets, ionosphere, isolet, splice, svmguide3, wbc, wdbc. Among the three algorithms, OL_{SF} performs the worst on all the 12 data sets. This is because the 12 UCI data sets contain noise, OL_{SF} which relies on a strict update strategy overfits the noise and thus performs the worst. In contrast, OL_{SF}-I and OL_{SF}-II using a "soft" update strategy can avoid overfitting. Furthermore, we can see that OL_{SF}-I scales well on large data sets, while OL_{SF}-II performs the best on small data sets. This is because OLSF-I scales linearly with the slack variable.

Fig. 5.2 shows the error rate with respect to the streaming iterations on the 12 data sets. Similar to the above results, we can observe that both OL_{SF}-I

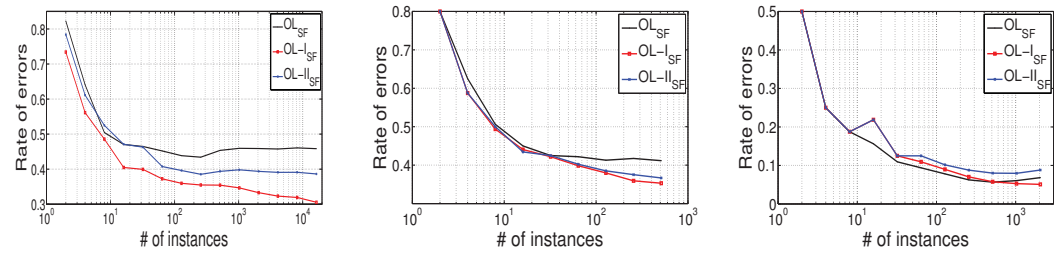
Table 5.3: The average number of prediction errors on the 12 UCI data sets

Algorithms	a8a	german	HAPT
OL_{SF}	12673.5 ± 75.9	415.9 ± 15.6	257.0 ± 8.5
OL_{SF}-I	**11204.7±713.1**	**366.9±8.8**	**167.0 ± 7.1**
OL_{SF}-II	11317.2±233.1	366.9±12.8	191.0 ± 9.9
Algorithms	ionosphere	isolet	magic04
OL_{SF}	55.0 ± 2.8	23.5 ±4.9	8051.3 ± 49.0
OL_{SF}-I	55.0 ±2.8	21.5±2.1	**6732.3±73.3**
OL_{SF}-II	**50.5 ± 6.4**	**18.0±4.2**	6924.5 ± 39.6
Algorithms	spambase	splice	svmguide3
OL_{SF}	1132.1 ± 29.7	1314.6 ± 30.3	396.7 ± 15.8
OL_{SF}-I	**1004.5±25.6**	1243.7 ± 13.6	359.1 ± 42.9
OL_{SF}-II	1013.2±26.1	**1238.8±16.8**	**357.5±26.9**
Algorithm	wbc	wdbc	wpbc
OL_{SF}	37.5 ± 0.7	43.5 ± 3.5	88.5 ± 2.1
OL_{SF}-I	35.5 ± 0.7	39.5 ± 0.7	**82.0 ± 8.5**
OL_{SF}-II	**34.0 ± 4.2**	**38.5 ± 4.9**	83.0 ± 1.4

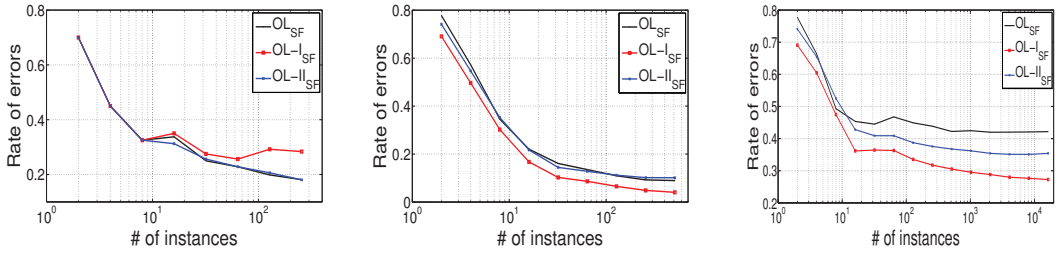
and OL_{SF}-II consistently outperform OL_{SF}. In addition, the performance gain of OLSF-I and OLSF-II raises with a large probability when new training instances arrive. This observation validates that OL_{SF}-I and OL_{SF}-II, by using slack variants to obtain soft update, can avoid overfitting to noise.

Fig. 5.3 shows the performance of the three algorithms under different projected feature space B. We can observe that OL_{SF}-I and OL_{SF}-II often outperform OL_{SF}. The results show the robustness of OL_{SF}-I and OL_{SF}-II under different subspace defined by B.

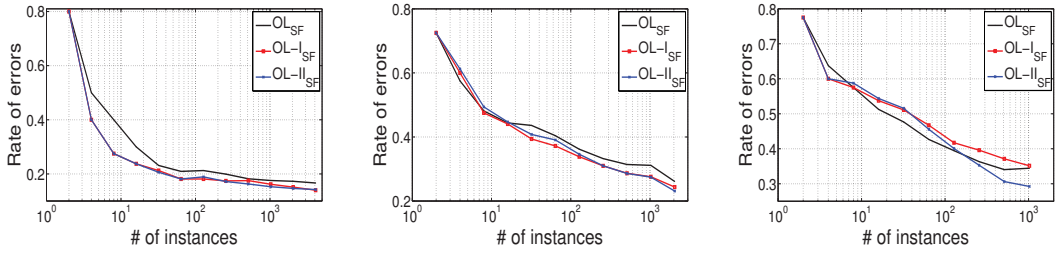
Fig. 5.4 shows the performance of the three algorithms under different tradeoff C. From the results, we can observe that varying the parameter C can alter the error rate of OL_{SF}-I and OL_{SF}-II. The larger of C, the closer of OL_{SF}-I to OL_{SF}. This is because the parameter τ_t in OL_{SF}-I is smaller than both parameter C and τ_t in OL_{SF}. When C is very large, OL_{SF}-I degenerates to OL_{SF}.



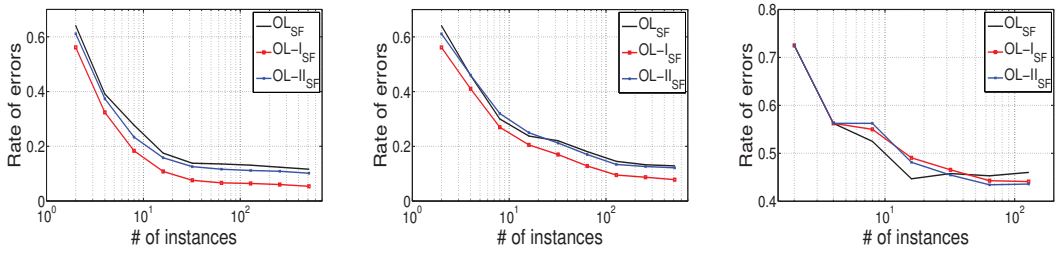
(a) a8a (b) german (c) HAPT



(d) ionosphere (e) isolet (f) magic04

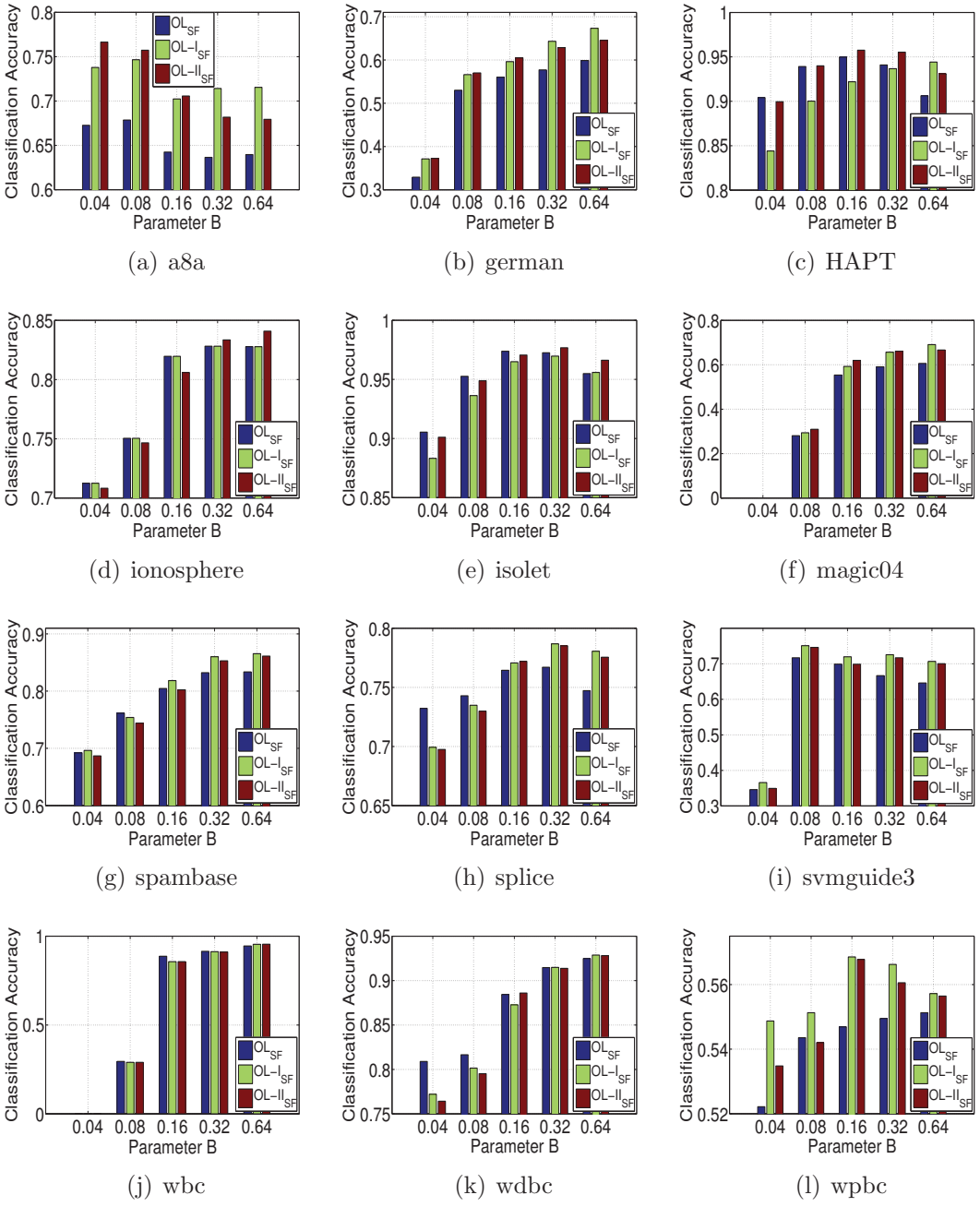


(g) spambase (h) splice (i) svmguide3



(j) wbc (k) wdbc (l) wpbc

Figure 5.2: Comparison among the proposed three algorithms OL$_{SF}$, OL$_{SF}$-I and OL$_{SF}$-II on the 12 UCI data sets. We can observe that OL$_{SF}$-I and OL$_{SF}$-II outperform OL$_{SF}$ because their "soft" update strategies can avoid overfitting to noise.



(a) a8a (b) german (c) HAPT

(d) ionosphere (e) isolet (f) magic04

(g) spambase (h) splice (i) svmguide3

(j) wbc (k) wdbc (l) wpbc

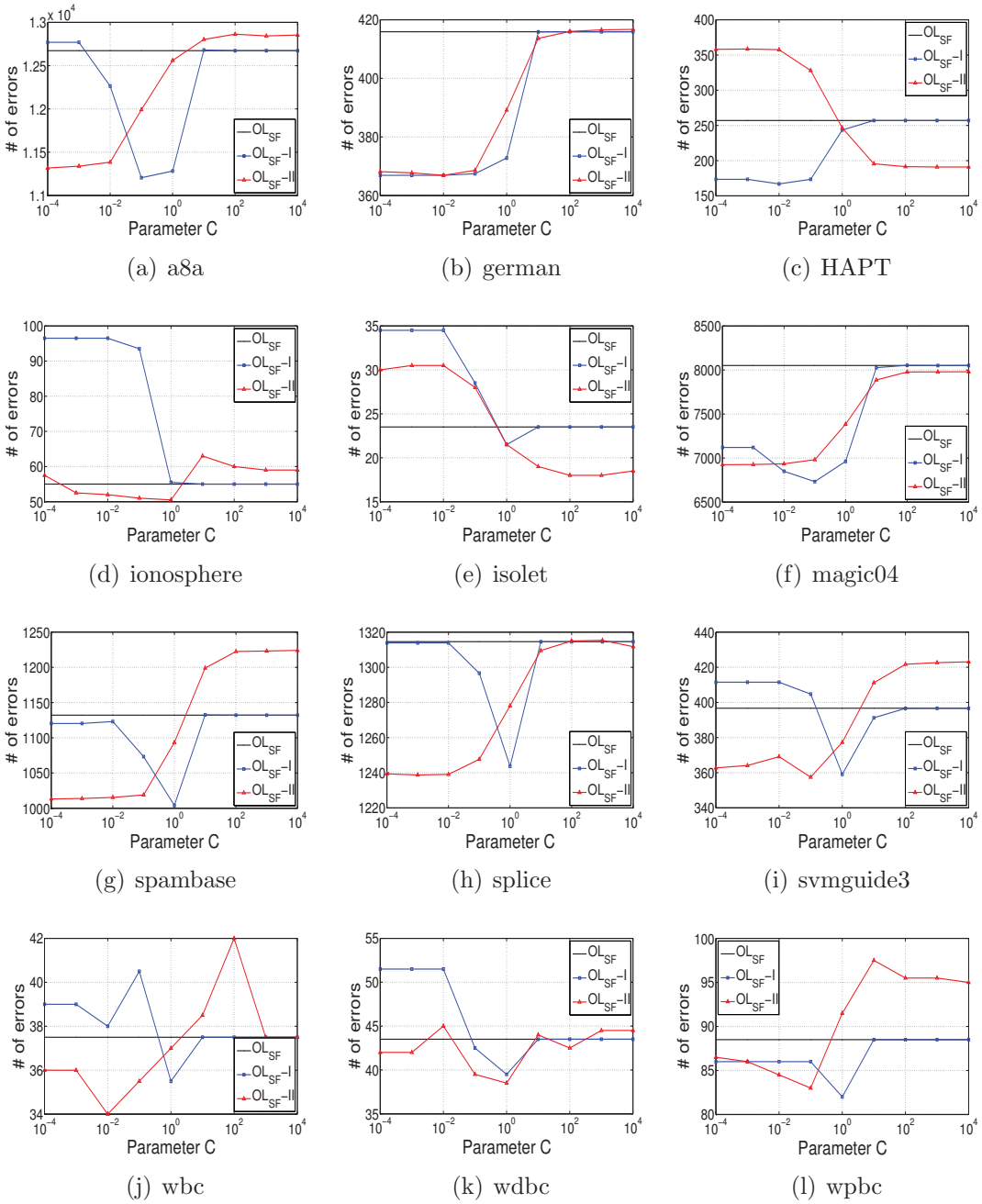
Figure 5.3: Performance comparison with respect to the projected feature space B. The results show that OL$_{SF}$-I and OL$_{SF}$-II are robust algorithms under different projected subspaces B.

Table 5.4: The average number of error predictions on the 12 UCI data sets with respect to parameter B.

	Algorithms	a8a	german	HAPT	magic04
$B = 0.04$	OL_{SF}-I	87.2 ± 8.8	56.8 ± 21.3	19020.0 ± 0.0	313.2 ± 55.2
	OL_{SF}I-rand	133.0 ± 4.5	190.2 ± 10.0	19020.0 ± 0.0	1318.6 ± 31.8
	OL_{SF}I-per	141.9 ± 6.3	92.5 ± 40.6	19020.0 ± 0.0	737.0 ± 171.8
$B = 0.16$	OL_{SF}-I	83.2 ± 6.9	**15.7 ± 3.9**	7379.2 ± 98.2	164.1 ± 14.8
	OL_{SF}I-rand	112.5 ± 6.4	144.4 ± 8.3	13107.3 ± 53.1	1158.1 ± 27.9
	OL_{SF}I-per	141.2 ± 5.8	35.4 ± 12.4	13143.5 ± 52.7	441.0 ± 51.1
$B = 0.64$	OL_{SF}-I	82.6 ± 3.7	25.6 ± 2.9	**5882.2 ± 105.3**	182.4 ± 41.9
	OL_{SF}I-rand	91.3 ± 4.8	69.7 ± 5.6	8361.8 ± 60.0	779.6 ± 15.2
	OL_{SF}I-per	83.4 ± 3.5	32.5 ± 3.1	6864.1 ± 49.5	420.6 ± 17.7
$B = 1.00$	OL_{SF}I-all	**79.3 ± 3.3**	16.7±1.8	6634.3 ± 35.2	**157.0 ± 8.9**

	Algorithms	spambase	splice	ionosphere	isolet
$B = 0.04$	OL_{SF}-I	788.4 ± 38.2	628.6 ± 42.5	108.7 ± 38.4	683.0 ± 0.0
	OL_{SF}I-rand	1041.0 ± 9.3	791.2 ± 11.1	375.8 ± 7.3	683.0 ± 0.0
	OL_{SF}I-per	1034.7 ± 29.0	868.3 ± 18.2	469.4 ± 14.1	683.0 ± 0.0
$B = 0.16$	OL_{SF}-I	**348.3 ± 49.4**	402.2 ± 39.7	65.8 ± 13.1	77.2 ± 15.7
	OL_{SF}I-rand	720.3 ± 13.3	582.4 ± 11.3	240.5 ± 10.4	405.5 ± 7.8
	OL_{SF}I-per	828.9 ± 51.2	711.9 ± 17.1	327.0 ± 14.1	529.6 ± 8.5
$B = 0.64$	OL_{SF}-I	364.4 ± 11.0	**326.7 ± 7.0**	**40.6 ± 4.3**	**31.3 ± 3.0**
	OL_{SF}I-rand	570.2 ± 20.3	456.0 ± 18.9	87.4 ± 7.0	79.2 ± 7.7
	OL_{SF}I-per	368.3 ± 51.7	365.0 ± 9.8	63.4 ± 3.8	85.9 ± 6.3
$B = 1.00$	OL_{SF}I-all	360.9 ± 7.2	344.1 ± 7.1	57.5 ± 3.7	82.9 ± 6.4

	Algorithms	svmguide3	wbc	wdbc	wpbc
$B = 0.04$	OL_{SF}-I	100.9 ± 12.4	850.2 ± 69.9	1397.2 ± 176.1	7488.4 ± 93.7
	OL_{SF}I-rand	234.7 ± 6.3	2026.5 ± 26.5	2808.7 ± 28.1	18249.4 ± 59.3
	OL_{SF}I-per	225.3 ± 8.6	2221.1 ± 43.5	2980.9 ± 80.9	20265.0 ± 285.0
$B = 0.16$	OL_{SF}-I	63.3 ± 8.7	**728.1 ± 20.7**	**835.3 ± 109.0**	**8680.3 ± 316.9**
	OL_{SF}I-rand	186.6 ± 6.4	1532.0 ± 30.4	1935.5 ± 38.0	16087.6 ± 102.8
	OL_{SF}I-per	220.5 ± 7.0	1308.8 ± 34.5	1248.8 ± 96.3	9659.6 ± 1444.9
$B = 0.64$	OL_{SF}-I	**54.8 ± 3.1**	683.7 ± 14.2	571.1 ± 15.3	9265.4 ± 115.4
	OL_{SF}I-rand	115.9± 8.5	1464.2 ± 33.4	1601.9 ± 23.4	15341.5 ± 71.0
	OL_{SF}I-per	60.0 ± 4.7	1244.8 ± 25.3	1003.7 ± 26.9	11325.6 ± 127.5
$B = 1.00$	OL_{SF}I-all	55.3 ± 2.7	1236.1 ± 29.5	983.6 ± 21.7	10243.0 ± 109.8



(a) a8a (b) german (c) HAPT

(d) ionosphere (e) isolet (f) magic04

(g) spambase (h) splice (i) svmguide3

(j) wbc (k) wdbc (l) wpbc

Figure 5.4: The average number of error predictions with respect to parameter C. We can choose the best parameter for the algorithms on the 12 UCI data sets.

5.5.2 Experiment II: Comparisons with benchmarks

We compare the proposed algorithms with three benchmark methods. According to the similar performance of OL$_{SF}$-I and OL$_{SF}$-II, we use OL$_{SF}$-I as the representative algorithm in this part.

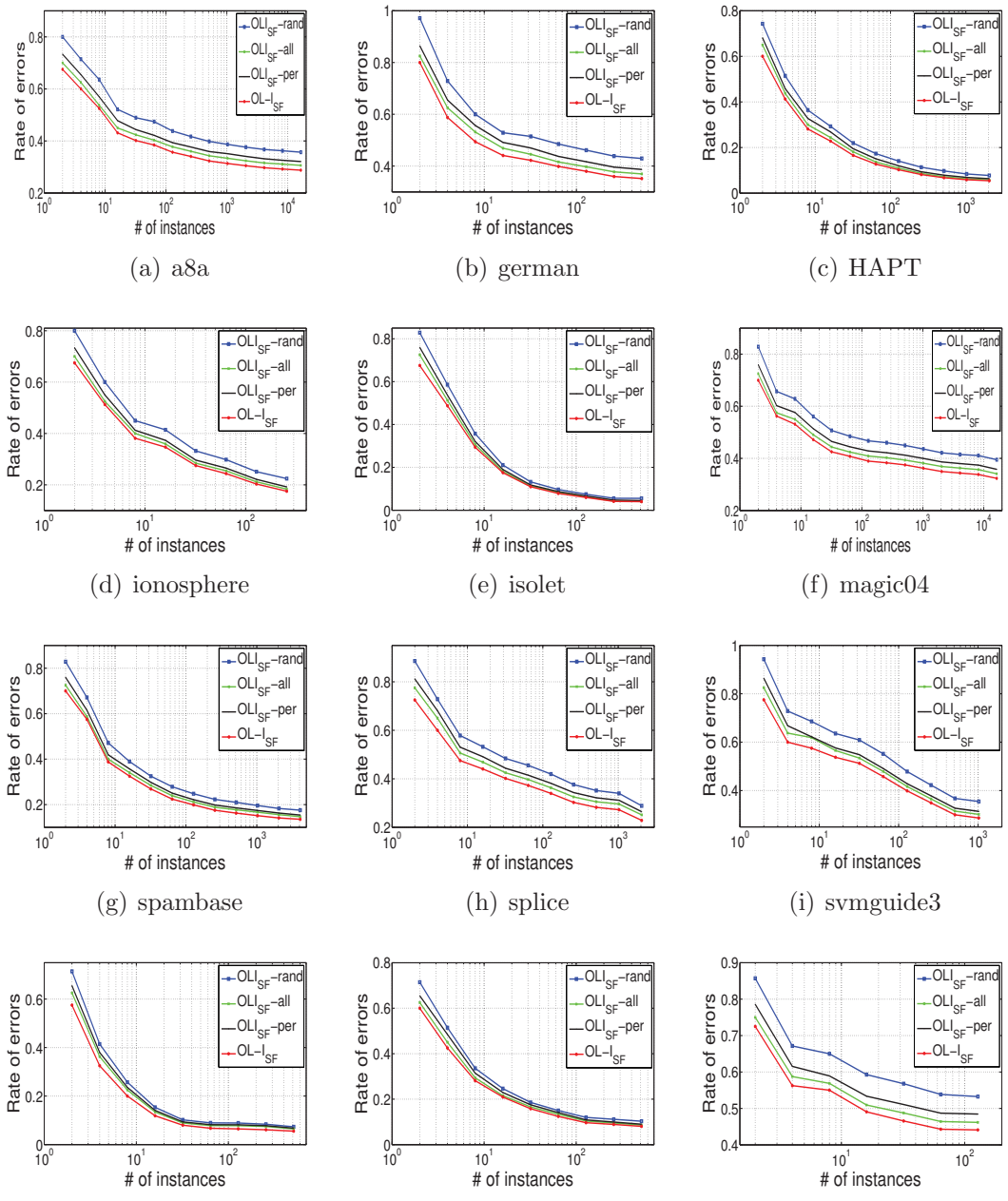
Now we introduce the three benchmark methods. The first algorithm is **OLI$_{SF}$-all**. Different from OL$_{SF}$-I that only uses a small projected feature space for learning, OLI$_{SF}$-all uses all features for learning. The second algorithm is **OLI$_{SF}$-rand** which uses randomly selected features for learning. The third algorithm is **OLI$_{SF}$-per** which uses the Perceptron update strategy for learning, i.e., $w_{t+1} = [w_t + y_t x_t, y_t x_t]$ [107]. We still use the 12 UCI data sets for our evaluation. The parameter settings are the same as in Experiments I.

Table 5.4 lists the average number of error predictions of the four algorithms on the 12 UCI data sets with different values of the parameter B. First, we can observe that OL$_{SF}$-I obtains the best results on 10 data sets out of 12. It even beats the OLI$_{SF}$-all algorithm which uses all the features for learning. Compared to the other two algorithms OLI$_{SF}$-rand and OLI$_{SF}$-per, OL$_{SF}$-I outperforms them under different B. The OL$_{SF}$I-rand algorithm randomly chooses a fixed proportion of features which receives the worst performance on all the 12 data sets. The OL$_{SF}$I-per algorithm which uses the Perceptron update strategy has higher error rates than OL$_{SF}$-I on all the 12 data sets, which shows that our update is better than the Perceptron update.

To sum up, the results show that the sparsity strategy in OL$_{SF}$-I can significantly improve the performance and our update strategy outperforms the Perceptron update strategy.

Fig. 5.5 shows the results of the online average error rates during the online learning on the 12 data sets. We can observe that the error rate of the algorithms decreases rapidly and becomes stable. OL$_{SF}$-I obtains the best results on all the data sets, which OLI$_{SF}$-rand obtains the worst results. The observation validate the results in Table 5.4.

To further examine the performance of these four algorithms, Fig. 5.6 shows the performance of the four algorithms with respect to different feature sets. The OL$_{SF}$-I algorithm outperforms the other three benchmark algorithms under



(a) a8a (b) german (c) HAPT

(d) ionosphere (e) isolet (f) magic04

(g) spambase (h) splice (i) svmguide3

(j) wbc (k) wdbc (l) wpbc

Figure 5.5: Comparison of the four algorithms under online learning setting. We can observe that OL$_{SF}$-I obtains the best results on all the 12 data sets because its sparsity strategy can significantly improve the performance and outperforms the Perceptron update strategy.

the same feature sets. In particular, OL$_{SF}$-I significantly outperforms the others when the subspace is very sparse, i.e., the parameter B is very small. The results show that the OL$_{SF}$-I algorithm can gain better sparsity and OL$_{SF}$-I performs well under a sparse feature space. This encouraging result verifies the efficacy of the proposed algorithms. Compared to OL$_{SF}$I-all that uses all features for learning, OL$_{SF}$-I achieves better results with sparser feature available.

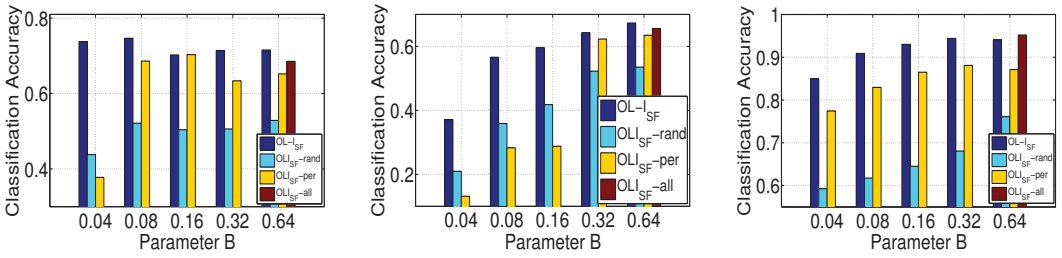
5.5.3 Experiment III: Comparisons with the state-of-the-art online feature selection algorithms

In this section, we compare the proposed OL$_{SF}$-I and OL$_{SF}$-II algorithms with the Online Feature Selection algorithms (OFS for short) proposed by J. Wang et al. [132] and its variant OFS$_P$, i.e., OFS with partial feature sets.

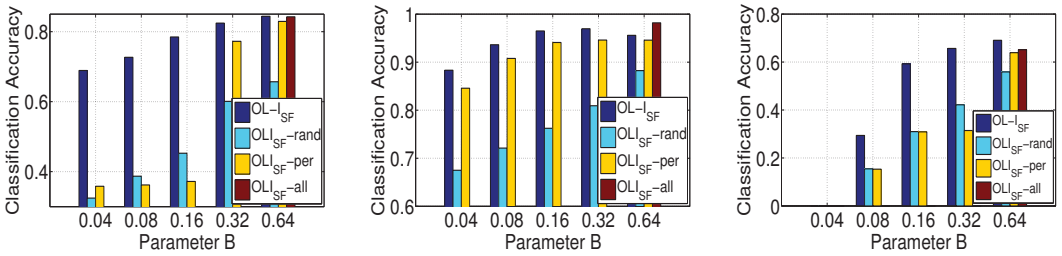
The OFS algorithm can access all the features for training and efficiently identify a fixed number of relevant features for prediction by using a gradient-based online learning update strategy and an l_2-norm projected truncation approach. OFS_P assumes only a partial number of features can be selected based on a Bernoulli distribution and then used for learning. The original codes of OFS and OFS$_P$ can be obtained online *http://OFS.stevenhoi.org/*.

In this part, we set the parameter $B = 0.1$, i.e., we use 10% features for learning at each round t. The tradeoff parameter C ranges from 10^{-4} to 10^4. OL$_{SF}$-I and OL$_{SF}$-II use 50% of the features for learning before 10% training instances are observed. Then, the algorithm continuously observes additional 10% features at each new data chunk.

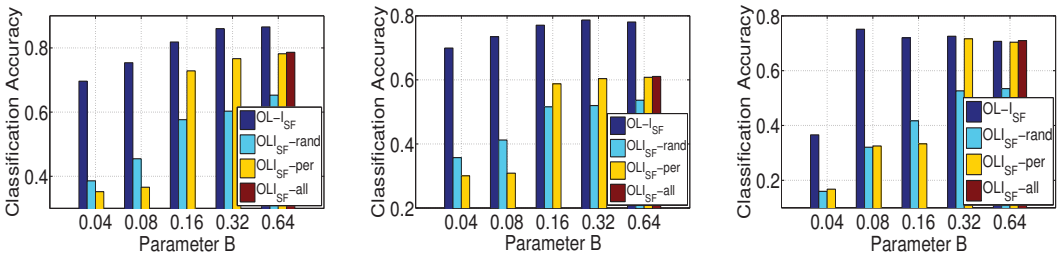
Table 5.5 and Fig. 5.7 show the average number of error predictions of the four algorithms. We can observe that OL$_{SF}$-I obtains the lowest error rate on the six data sets. Moreover, OL$_{SF}$-I significantly outperforms both OFS and OFS$_P$. When comparing OL$_{SF}$-II with OFS$_P$ and OFS, we can observe that OL$_{SF}$-II performs better on the six data sets than OFS$_P$. OL$_{SF}$-II also outperforms OFS on four data sets. This is because OL$_{SF}$-I and OL$_{SF}$-II have better update strategies than OFS and OFS$_P$ by adding a flexible learning rate τ_t. We can also observe that OL$_{SF}$-I and OL$_{SF}$-II are more stable because their standard deviations are significantly lower than OFS and OFS$_P$.



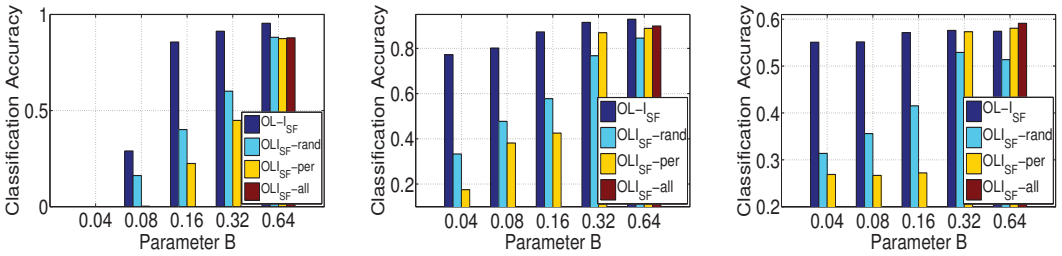
(a) a8a (b) german (c) HAPT



(d) ionosphere (e) isolet (f) magic04



(g) spambase (h) splice (i) svmguide3



(j) wbc (k) wdbc (l) wpbc

Figure 5.6: Online classification accuracy with respect to the parameter B. We can observe that OL$_{SF}$-I performs the best especially when the feature space is sparse, i.e., B is very small.

Table 5.5: Comparison with respect to the average number of error predictions.

Algorithms	a8a	german	magic04
OFS	9424.4 ± 2545.8	432.8 ± 13.6	6023.4 ± 1342.3
OFS_P	16931.0 ± 164.6	589.3 ± 33.9	10274.2 ± 172.1
OL_{SF}-I	**9322.7 ± 41.1**	**318.5 ± 7.3**	**5858.4 ± 29.6**
OL_{SF}-II	10709.3 ± 56.0	348.7 ± 11.6	5917.9 ± 55.9
Algorithms	spambase	splice	svmguide3
OFS	913.1 ± 157.8	735.4 ± 68.3	400.9 ± 66.8
OFS_P	1954.2 ± 78.7	1418.1 ± 70.5	701.5 ± 42.5
OL_{SF}-I	**616.6 ± 12.2**	**725.5 ± 18.8**	**374.2 ± 10.8**
OL_{SF}-II	690.7 ± 14.0	748.7 ± 16.0	382.1 ± 11.4

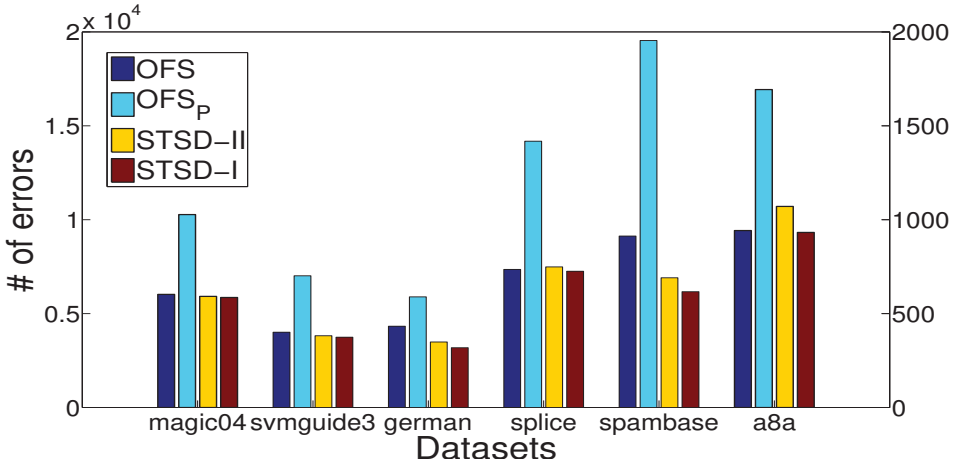
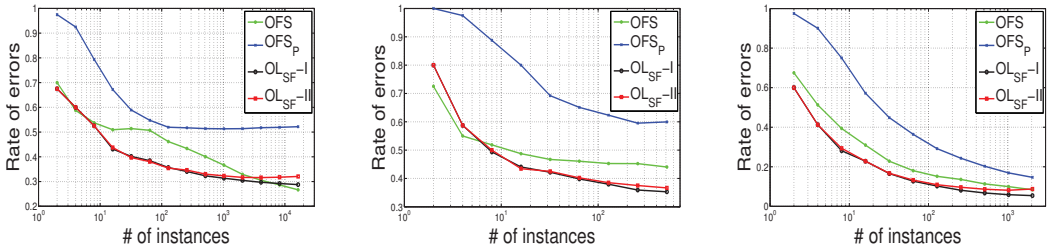
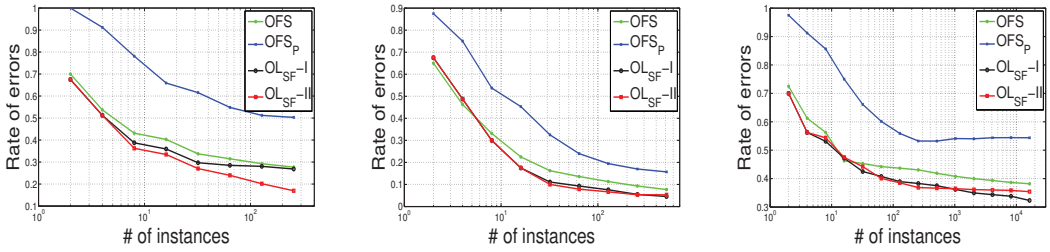


Figure 5.7: Comparison with respect to the average number of error predictions. We can observe that OL_{SF}-I and OL_{SF}-II performs better than OFS and OFS_P by adding a flexible learning rate τ_t.

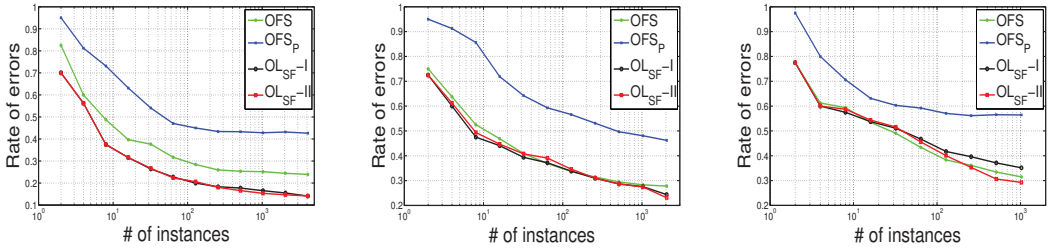
Furthermore, we compare the online prediction performance in Fig. 5.8. We can observe that the error rate varies at each iteration, where the curves of OL_{SF}-I and OL_{SF}-II descend much faster than those of OFS and OFS_P and eventually become stable with better results.



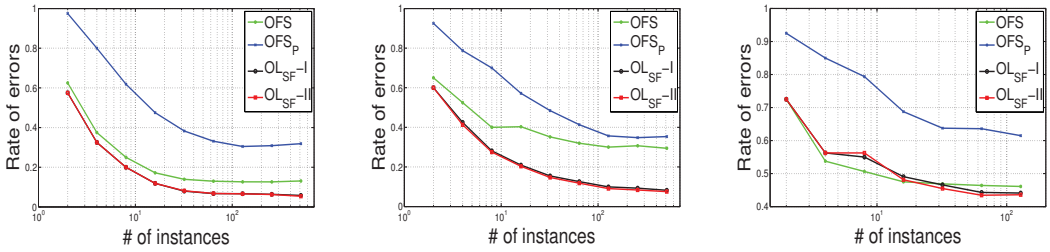
(a) a8a (b) german (c) HAPT



(d) ionosphere (e) isolet (f) magic04



(g) spambase (h) splice (i) svmguide3



(j) wbc (k) wdbc (l) wpbc

Figure 5.8: Comparison with respect to online prediction. We can observe that the curves of OL$_{SF}$-I and OL$_{SF}$-II descend much faster than those of OFS and OFS$_P$ and eventually become stable with lower error rates.

Table 5.6: Comparison with respect to the average number of error predictions ($B = 0.001$).

Algorithms	rcv1	URL
OL_{SF}-I	239582.0±1104.2	**599352.0±8888.1**
OL_{SF}I-all	**235280.8±1459.4**	607019.6±8051.6
OL_{SF}I-rand	482310.1±443.5	1520743.8±12546.3
OL_{SF}I-per	329572.6±1113.5	602546.8±9063.3

5.5.4 Experiment IV: Applications to real-world trapezoidal data streams

In this part, we evaluate the performance of the proposed algorithms on two real-world data streams. The data sets can be downloaded online [15].

The task of the URL dataset [86] is to detect malicious URLs from Webpage streams using lexical and host-based features of URLs. In the task, URLs arrive continuously as streams, where each URL carries lexical and host-based features that we have never seen before. The purpose is to continuously learn a URL classifier that can identify malicious Webpages from normal ones. Thus, the learning problem can be formulated as online learning from trapezoidal data streams. The task of rcv1 text classification is to categorize the JMLR articles into different groups. Because new articles are published continuously with new research topics, the problem can be also defined as online learning from trapezoidal data streams.

Table 5.6 shows the experimental results of the average number of error predictions of the four algorithms. We set the parameter $B = 0.001$. The tradeoff parameter $C = 0.1$. From the results, we can observe that OLSF-I that uses only 0.1% features performs similarly to OLSF-all that uses all the features on rcv1 dataset. Fig. 5.9 shows the performance of the algorithms with respect to the number of training instances when $B = 0.01$, i.e., using 1% features to learn. We can observe that OL_{SF}-I, OL_{SF}-all, OL_{SF}-per converge fast when the number of training instances increases. Moreover, OL_{SF}-I performs better than the other three algorithms and converges to the lowest error rates.

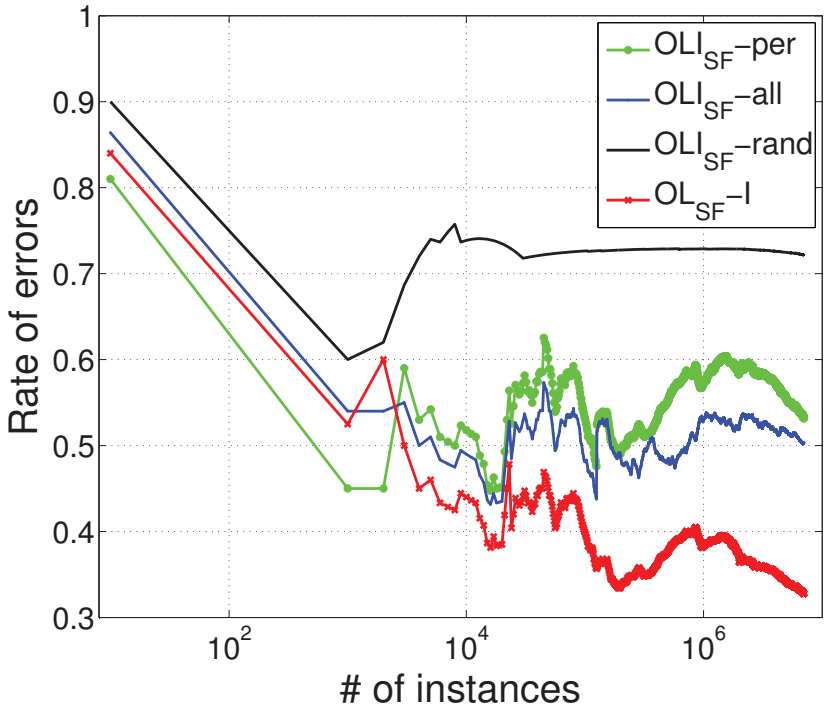
5.5.5 Discussions

Multi-class classification. There are two methods *One vs Rest* and *One vs One* [120] that can extend the proposed algorithms to multi-class classification by converting the problem to be multiple binary classification problems [7]. For a c-class problem in *One vs One*, it often requires to build $c(c-1)/2$ binary classifiers. From the model formulation perspective, we can directly extend the vector-based models to matrix-based models.

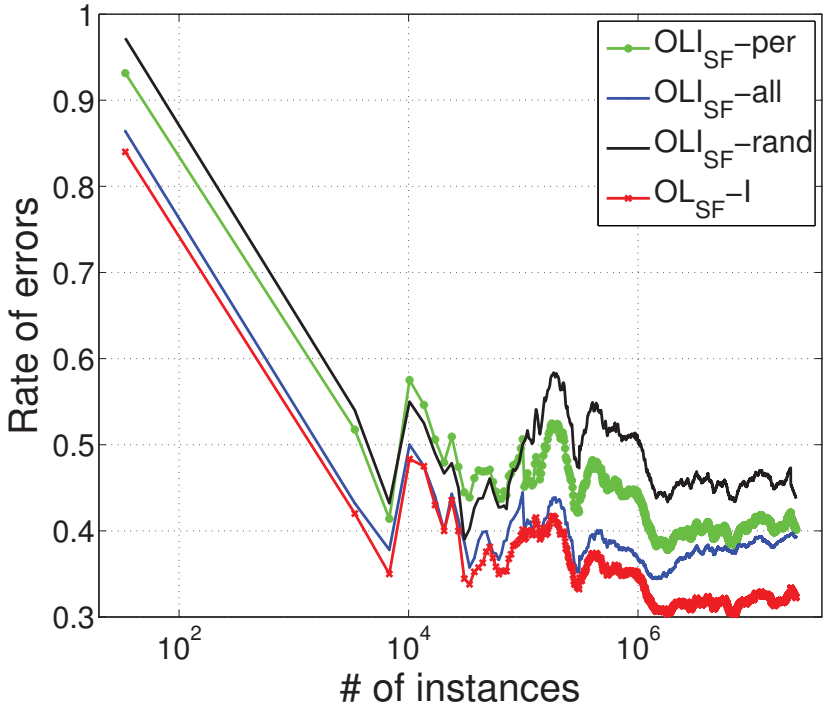
Semi-supervised classification. In many applications, labels are provided only for a few data points [128] [153]. Here, pseudo-labels can be used to enlarge a labelled training set. Specifically, we can use the classifiers trained from labelled examples to predict class labels (pseudo-labels) of unlabelled examples. Then, a semi-supervised learner can be built from both labelled and pseudo-labelled examples.

5.6 Conclusions

In this chapter we studied a new problem of online learning from trapezoidal data streams where both data volume and feature space increase by time. We proposed a new Online Learning with Streaming Features algorithm (OL_{SF}) and its two variants OL_{SF}-I and OL_{SF}-II as the solution. Theoretical and empirical analysis have shown the performance of the proposed algorithms.



(a) rcv1 text classification



(b) URL data set

Figure 5.9: Performance on real trapezoidal data streams($B = 0.01$).

Part IV

Dynamic implicit social recommendation

Chapter 6

Online dynamic implicit social recommendation

6.1 Introduction

Social recommendation is an everyday process that frequently touches people's lives. A general and dynamic social recommendation which can make timely and accurate personalized recommendations is urgently needed. The quicker the recommender system responds, the greater the likelihood that it will identify the user's current preferences and needs. For instance, after a user books a flight, hotels and places of interest in the destination city should be recommended to the user in a timely fashion. Also many recommendations should be given to users in advance due to capacity and budget issues, such as hotel recommendations which should be provided several days or months in advance so as to allow the user sufficient time to choose and book a hotel in a lower price. However there are still three main challenges in this area.

The first one is the drift of a user's preferences and needs over time. A recommender system needs to capture users' dynamic preferences and needs rapidly and deliver the right recommendations at the right time[162]. For instance, if the user has recently become pregnant, her preferences are likely to change dramatically to baby-related goods such as toys, buggies and car seats.

The second challenge is how to obtain the latent dynamic core social networks

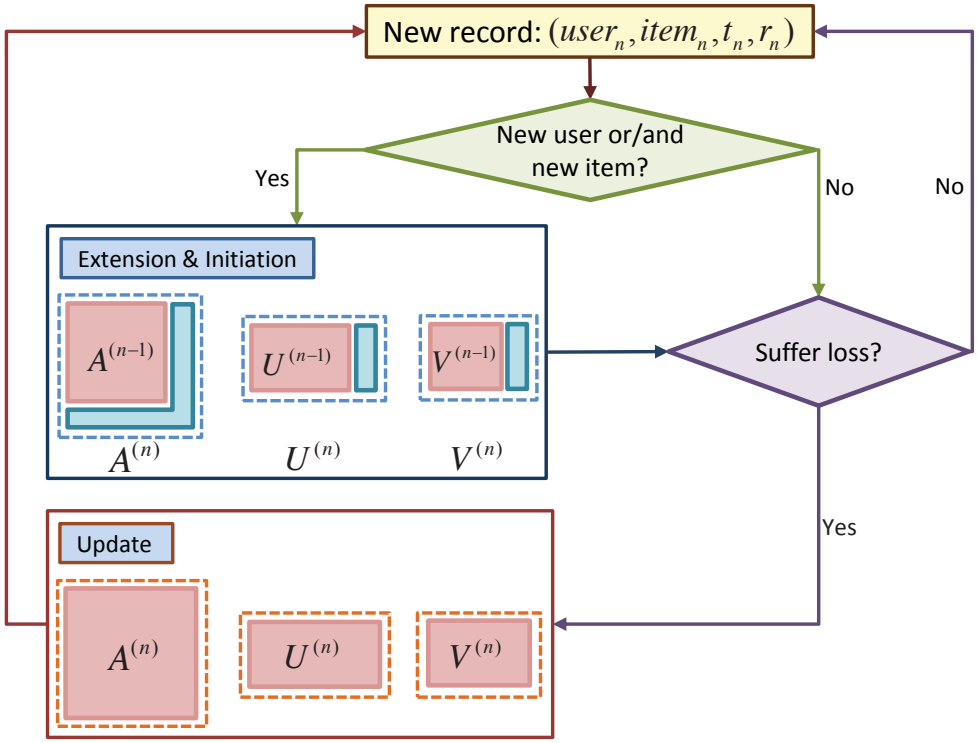


Figure 6.1: The framework of the General Online Dynamic Social Recommendation method. When a new record $(user_n, item_n, t_n, r_n)$ arrives, we first find out whether a new user or/and new item has appeared or not. If the answer is yes, we go to the extension and initiation step, otherwise we go to the loss calculation step directly. Then, if it suffers loss in this current record, we go to the update step to improve the model, otherwise we skip this record and go to the next loop with the following new record.

of users. A core social network is different from the general explicit social network, as it only contains the very close friends of the users who truly influence him. However, this is difficult to obtain since it is dynamic and implicit. For instance, supposing a user has recently moved to a new city, it is natural for him to ask for recommendations from his friends in the new city rather than other people. But this change is difficult to reflect in the whole large explicit social network which contains a large amount of redundant information and noise. If we consider the whole explicit social network, it will divert much attention from the network of real influence and harm the final performance. On the other side, most disclosed social data are normally binary decision values, e.g. whether two

people are friends or not, so the strength of their relationship is missing [35]. Also many signals of user's preferences are implicit [36], such as watching a video or clicking on a link, like *Google News* [23]. The complete core social network with weights is important for a social recommendation model and would improve the recommendations significantly.

The last challenge is how to deal with new users and new items which appear in the system. In reality, this problem is unavoidable. New users and new items enter the recommender system continuously, so how to involve them and how to handle this problem smoothly must be considered.

However, to the best of our knowledge, currently there is no research that tries to address these challenges simultaneously. Most of the existing research either builds a general model in a sample scenario such as ignoring the dynamic nature of the system and the problem of new users and items, or only focuses on some specific fields, such as dynamic restaurant recommendation [17] and timely news recommendation [23].

To this end, in this chapter, a novel General Online Dynamic Social Recommendation(GODSR) model is proposed to address these challenges simultaneously and effectively. Specifically, we combine network inference, online learning and collaborative filtering together in an iterative process. By inferring the latent dynamic core social networks from cascade data, identifying the drift of a user's preferences and involving new users and items in the online learning process, it reacts rapidly and make accurate recommendations to users when new data arrive. We illustrate the framework of the proposed model in Fig. 6.1. Experiments on three real-world datasets demonstrate the effectiveness of the proposed model compared with other state-of-the-art solutions.

The main contributions of the chapter are as follows:

1) We study a new problem of online dynamic social recommendation with latent core social networks, which addresses the concerns of the continuous drift of a user's preferences and needs, the dynamic core social network of users and the problem of newly appearing users and items to make timely and accurate recommendations.

2) We propose a new General Online Dynamic Social Recommendation method

which combines network inference, online learning and the collaborative filtering techniques together in an iterative process to make timely and accurate recommendation. It addresses the three aforementioned concerns simultaneously and effectively, which is different with most current researchers that tried to design the methods in a defined specific scenario.

3) We use two real-world datasets (Zomato and Douban movie data) we crawled, and the public MovieLens dataset to evaluate the performance of the proposed model. Experiments show that the proposed model outperforms the state-of-the-art solutions on these three real-world datasets.

The rest of this chapter is organized as follows: Section 6.2 introduces the details of the problem setting, and Section 6.3 introduces the algorithm. Section 6.4 describes the experiments and Section 6.5 concludes this chapter.

6.2 Problem setting

In this section, we consider the general online dynamic social recommendation problem. The records of users' actions appear one by one over time in the online learning scenario. We denote each record as a tuple $(user_n, item_n, t_n, r_n), n = 1, 2, \ldots$, which represents that $user_n$ rated $item_n$ at time t_n with the rate of r_n. $n = 1, 2, \ldots$ is the index of the record. Users and items can appear repeatedly. t_n is the time of the n-th record. Since we are not concerned about the specific time but the relative time of the records, we set $t_1 = 0$ and $t_n < t_{n+1}, n = 1, 2, \ldots$. rate r_n are normally positive numbers that in $[0, 5]$.

Historical data from a long time ago has little contribution to current recommendations, so we only use observations over a limited time period. Assume the observation window is $[0, T]$, thus $t_n \in [0, T], n = 1, 2, \ldots$. The records of the observed action time stamps by users on a certain item, such as item j, constitute to a cascade data $c^{(j)} = [(u_1^{(j)}, t_1^{(j)}); (u_2^{(j)}, t_2^{(j)}); \ldots; (u_{n_j}^{(j)}, t_{N_j}^{(j)})]$ where N_j is the length of the cascade. $u_p^{(j)}, p = 1, \ldots, N_j$ is the i-th user that rated item j at time $t_p^{(j)}$, and $t_p^{(j)} \leq t_{p+1}^{(j)}$ is always satisfied. Assume at the end of the observation window, i.e. at time T, there are N users and M items appeared in total. Thus there are M cascades in total, i.e. $C = \{c^{(1),} \ldots, c^{(M)}\}$. The length of each cascade can be

smaller than N because not all users rated the related item, or it can be larger than N since same user can rate a certain item at different times. Appropriately lengthening the observation window T increases the number of observed actions in a cascade $c^{(j)}$ and results in a more representative sample of the underlying dynamic networks [105].

Our goal is to build a social recommendation model to make timely and accurate recommendations and to address the three concerns previously raised, these being: the drift of a user's preferences and needs, the dynamic core social networks of users and the problem of newly appearing users and items.

6.3 General online dynamic social recommendation

In this section, we describe our general online dynamic social recommendation algorithm. During the observation window $[0, T]$, the records of the actions taken by users to the related items appear one by one. When a new record $(user\ i, item\ j, t_{ij}, r_{ij})$ arrives, we expect a small difference between the real rate r_{ij} and predicted one $\hat{r}_{ij} = U_i^T V_j$ where U_i, V_j are the latent feature vectors of user i and item j respectively. We assume K is the dimension of the latent feature space, so $U_i, V_j \in \mathbb{R}^{K\times 1}$. The smaller the difference, the more accurate the recommendation. By defining function $F_L = |r_{ij} - \hat{r}_{ij}|$, a reasonable loss function can be

$$\mathbb{L}(user\ i, item\ j, r_{ij}) = \begin{cases} 0, & \text{if } F_L \leq \epsilon \\ 1, & \text{if } F_L > \epsilon \end{cases} \tag{6.1}$$

where ϵ is a threshold value to control the strictness of the algorithm.

With the loss function, two main problems need to be addressed. The first one is how to update the parameter matrix, i.e. user feature matrix U, item feature matrix V and core social network matrix A, when a new record arrives without new users and new items. To address this problem, we adopt the idea of the probabilistic generation method to infer the implicit core social network and update the low-rank user matrix and item matrix by following the idea of the collaborative filtering approach. The second problem is when a new user

Algorithm 6 GODSR Algorithm

Require:

K: Latent feature space dimension parameter;
$\lambda_1, \lambda_2, \lambda_3, \lambda$: Tradeoff parameters and learning step size;
ϵ: Threshold parameter for loss function

Ensure:

U^*, V^*, A^*: User matrix, item matrix and Social network matrix;

Initialize $U^{(0)} = \Phi$, $V^{(0)} = \Phi$ and $A^{(0)} = \Phi$, where Φ is the empty matrix;
For $n = 1, 2, \ldots$ **do**
  receive data: $(user_n, item_n, t_n, r_n)$
  if $item_n$ is a new item
    extend $V^{(n-1)}$ to $V^{(n)} = [V^{(n-1)}, 1_K]$
    record the index of the $item_n$ as $j = N_n^{(V)}$
  else
    find the index j of $item_n$
  endif
  if $user_n$ is a new user
    extend $U^{(n-1)}$ to $U^{(n)} = [U^{(n-1)}, 1_K]$;
    extend $A^{(n-1)}$ to $A^{(n)}$ with Eq. (6.14)
    record the index of $user_n$ as $i = N_U^n$ in $H_U^{(j)}$
    record time t_n as the latest record of $user_i$ rated $item_j$ in $H_T^{(j)}$;
  else
    find the index i of $user_n$
  endif
  calculate loss: $\mathbb{L}_n = \mathbb{L}(r_n, U_i, V_j)$
  if suffer loss, i.e. $\mathbb{L}_n > 0$
    update A_i with Eq. (6.8)
    update U_i with Eq. (6.12)
    update V_j with Eq. (6.13)
  endif
endfor
return $U^* = U^{(n)}$, $V^* = V^{(n)}$, $A^* = A^{(n)}$;

or/and a new item appears with the coming record, how to involve them in the recommender system. To address this problem, we dynamically extend the three matrices, U, V, A, by adding the related latent user/item feature vector and network influence weight vector to the matrix and update them continuously.

The pseudo-codes for the GODSR algorithm are given in Algorithm 6 and the complexity analysis is provided in Section 6.3.3. The three main matrices, user matrix $U \in \mathbb{R}^{K \times N}$, item matrix $V \in \mathbb{R}^{K \times M}$ and social network matrix $A \in \mathbb{R}^{N \times N}$ are initialized as an empty matrix firstly, where K is the dimension of the latent feature space. N and M are the total number of users and items which appear at the end of the observation respectively. In the following, we firstly overview the general online update strategies, and then discuss the strategy taken in the scenario when a new user or/and a new item appears.

6.3.1 Online update strategy

We first consider the general scenario that neither a new user nor a new item appears when the n-th record $(user_n, item_n, t_n, r_n)$ arrives. Assume the indexes of $user_n$ and $item_n$ are i and j respectively, the n-th record can be rewritten as $(user\ i, item\ j, t_{ij}, r_{ij})$. Then, we go to the specific update strategies of the three matrices, core social network matrix A, user matrix U and item matrix V, respectively.

6.3.1.1 Update A

To infer the implicit core social network, we need to consider the pairwise interactions. With the cascade data, we assume that infections can occur at different rates over different edges of a network, and aim to infer the influence rates between pairs of users in the network [105]. The influence likelihood between user p and i depends on the action time pair (t_{pj}, t_{ij}) in cascade $c^{(j)}$, and the pair-wise influence rate $A_{p,i}$. A user cannot be influenced by a user who has never appeared before. In other words, user p who appeared at time t_p may influence user i at time t_i only if $t_p < t_i$. Following the work of [105], we define the conditional likelihood of the influence between user p and user i with the well-known exponential parametric likelihood function which is shown in Eq. (6.2),

$$f(t_{ij}|t_{pj}; A_{pi}) = \begin{cases} A_{pi} \cdot e^{-A_{pi}(t_{ij}-t_{pj})}, & \text{if } t_{pj} < t_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

where $A_{pi} \geq 0$ is the influence rate between user p and i. If $A_{pi} \to 0$, the likelihood of influence tends to zero and the expected transmission time becomes arbitrarily long.

When the n-th record $(user\ i, item\ j, t_{ij}, r_{ij})$ arrives, the users who previously rated item j may influence user i. We use $H_U^{(j)} = [u_1^{(j)}, u_2^{(j)}, \ldots, u_{N_j}^{(j)}]$ to record the indexes of the users who rated item j before time t_{ij} and $H_T^{(j)} = [t_1^{(j)}, t_2^{(j)}, \ldots, t_{N_j}^{(j)}]$ to record the related time stamps. N_j is the number of the users who have appeared in cascade $c^{(j)}$. We would like to maximize the joint likelihood of between the historical users $H_U^{(j)}$ and the current user i in Problem (6.3)

$$\max \prod_{p=1}^{N_j} f(t_{ij} | t_p^{(j)}, A_{u_p^{(j)},i}) \tag{6.3}$$

In combination with Eq. (6.2), the problem is equal to minimizing the following optimization problem, i.e. Eq. (6.4)

$$\max_{A>0} F_A = \sum_{p=1}^{N_j} ((t_{ij} - t_p^{(j)}) A_{u_p^{(j)},i} - \log A_{u_p^{(j)},i}) \tag{6.4}$$

We follow the idea of coordinate gradient decent to optimize the related elements in A. The derivative of $A_{u_p^{(j)},i}$ is

$$\frac{\nabla F_A}{\nabla A_{u_p^{(j)},i}} = t_{ij} - t_p^{(j)} - \frac{1}{A_{u_p^{(j)},i}} \tag{6.5}$$

where $p = 1, \ldots, N_j$. The final update strategy for $A_{u_p^{(j)},i}, p = 1, \ldots, N_j$ is

$$A_{u_p^{(j)},i} = \frac{1}{t_{ij} - t_p^{(j)}} \tag{6.6}$$

Further, we introduce the exponential technique to make it more smooth, i.e.

$$A_{u_p^{(j)},i} = \frac{1}{e^{t_{ij} - t_p^{(j)}}} \tag{6.7}$$

and the final update strategy for $A_{u_p^{(j)},i}, p = 1, \ldots, N_j$ is

$$A^{(l+1)}_{u_p^{(j)},i} = A^{(l)}_{u_p^{(j)},i} + \frac{1}{e^{t_{ij} - t_p^{(j)}}} \tag{6.8}$$

and the remaining elements of matrix A remain unchanged.

6.3.1.2 Update U and V

For user matrix U and item matrix V, when record $(user\ i, item\ j, t_{ij}, r_{ij})$ arrives, we try to shrink the difference between the real rate r_{ij} and the estimated rate $\hat{r}_{ij} = U_i^T V_j$, where U_i is the i-th column of matrix U and V_j is the j-th column of the matrix V representing the latent feature vector of user i and item j respectively. With the inferred social network A, we also want to make the new latent feature vector of user i similar to the ones of their close friends'. Following the idea of collaborative filtering, we update the related feature vectors U_i, V_j by minimizing the following optimization problem (6.9)

$$\begin{aligned} \min_{U_i, V_j} \quad & F_R = \frac{1}{2}(r_{ij} - U_i^T V_j)^2 + \frac{\lambda_1}{2}\|U_i\|_2 \\ & + \frac{\lambda_2}{2}\|V_j\|_2 + \frac{\lambda_3}{2}\sum_{p=1}^{N_U} A_{pi}\|U_i - U_p\|^2 \end{aligned} \tag{6.9}$$

where N_U is the total number of the users who have appeared before time t_{ij} which is also the same as the size of social network matrix A. $\|\cdot\|_2$ is the l_2-norm of the vector. $\lambda_1, \lambda_2, \lambda_3 > 0$ are tradeoff parameters. The second and third terms are two regularization terms to avoid overfitting. The last term is the social regularization term based on the latent core social network inferred above.

Following the idea of the coordinate gradient descent algorithm, we update U_i with fixed V_j. Problem (6.9) degrades to problem (6.10)

$$\begin{aligned} \min_{U_i} \quad & F_U = \frac{1}{2}(r_{ij} - U_i^T V_j)^2 + \frac{\lambda_1}{2}\|U_i\|_2 \\ & + \frac{\lambda_3}{2}\sum_{p=1}^{N_U} A_{pi}\|U_i - U_p\|^2 \end{aligned} \tag{6.10}$$

where the problem is convex and consistent with variable U_i, and the derivative of U_i is shown in Eq. (6.11)

$$\begin{aligned}\frac{\nabla F_U}{\nabla U_i} =& - V_j(r_{ij} - U_i^T V_j) + \lambda_1 U_i \\ &+ \lambda_3 \sum_{p=1}^{N_U} A_{pi}(U_i - U_p)\end{aligned} \tag{6.11}$$

Making the derivative equal to zero, we can obtain the update strategy of U_i as

$$U_i^* = \frac{r_{ij}V_j + \lambda_3 \sum_{p=1}^{N_U} A_{pi}U_p}{V_j^T V_j + \lambda_1 + \lambda_3 \sum_{p=1}^{N_U} A_{pi}} \tag{6.12}$$

And the update strategy of V_j is similar to U_i which is shown in Eq. (6.13) directly by omitting the intermediate steps,

$$V_j^* = \frac{r_{ij}U_i}{U_i^T U_i + \lambda_2} \tag{6.13}$$

6.3.2 The scenario with newly appearing users and items

We first consider the scenario when a new user appears, i.e., $user_n$ appears for the first time with the n-th record $(user_n, item_n, t_n, r_n)$. For the historical data, we assume N_U^{n-1} users and N_V^{n-1} items have already appeared. Thus, the learned user matrix $U^{(n-1)}$ at the last step is in the size of $K \times N_U^{n-1}$, learned item matrix $V^{(n-1)}$ is in the size of $K \times N_V^{N-1}$ and learned social network matrix $A^{(n-1)}$ is in the size of $N_U^{n-1} \times N_U^{n-1}$. When new $user_n$ appears, we first initialize the latent feature vector of the new user with a column vector of one, i.e. extend user matrix $U^{(n-1)}$ to $U^{(n)} = [U^{(n-1)}, 1_K] \in \mathbb{R}^{K \times N_U^n}$ where 1_K is a column vector in dimension of K and all the elements are one and $N_U^n = N_U^{n-1} + 1$. Further we need to explore the core social network of the new user. We need to find out which users appear to influence the new user a lot and how the new user influences the next newly appearing users. So, we extend the existing network matrix $A^{(n-1)}$ to

$$A^{(n)} = \begin{bmatrix} A^{(n-1)} & O_{N_U^{n-1}} \\ O_{N_U^{n-1}}^T & 1 \end{bmatrix}, \tag{6.14}$$

where $0_{N_U^{n-1}}$ is a column vector in dimension of N_U^{n-1} and all the elements are zero. The size of the square matrix $A^{(n)}$ becomes N_U^n.

On the other hand, when $item_n$ is a new item, we need to initialize the related latent feature vector of the new item by extending the existing item matrix $V^{(n-1)}$ with a column of one, i.e. $V^{(n)} = [V^{(n-1)}, 1_K] \in \mathbb{R}^{K \times N_V^n}$ where $N_V^n = N_V^{n-1} + 1$.

After the extension and initialization of the related matrix, we can adopt the general update strategy introduced in section 6.3.1 to update the latent feature vectors and the related core social network of the new users and items.

6.3.3 Complexity analysis

In Algorithm 6, GODSR takes constant time for the initialization (line 1). In the n-th loop, the record $(user_n, item_n, t_n, r_n)$ arrives. It takes $O(N_V^{n-1})$ to find the index of $item_n$ (line 4-9) and $O(N_U^{n-1})$ to find the index of $user_n$ (line 10-17). In the loss function calculation, it takes $O(K)$ (line 19). In the update step, it takes $O(N_U^n)$ to update A_i (line 20), $O(K + N_U^n + K * N_U^n)$ to update U_i (line 21) and $O(K)$ to update V_j (line 22). To sum up, the total worst-case time complexity of algorithm 6 is $O(N^\epsilon(N + M + K + K * N))$, where N^ϵ is the total number of records that suffers loss and ϵ is the parameter for loss function to control the strictness of the algorithm. Since $K \ll \min\{M, N\}$, the complexity can be simplified to $O(N^\epsilon(M + N + M * N))$.

6.4 Experiments

In this section, we describe several experiments conducted to compare the recommendation qualities of our approaches with other state-of-the-art recommendation methods. We first introduce the details of the datasets and metrics we use. We then introduce the experimental setup and benchmark methods. Finally, we compare the performance of the proposed algorithm with respect to MAEs and RMSEs. All experiments are conducted on a Windows 8 machine with 3.00GHz CPU and 8GB memory.

Datasets. We evaluate the performance of the proposed GODSR model

Table 6.1: Performance Comparisons in terms of MAE

Dataset	Training	online-MF	online-SocialMF	online-reciprocal	GODSR
Douban	40%	2.1626 (24.57%)	1.8151 (10.13%)	2.4458 (33.31%)	**1.6312**
	60%	2.1483 (28.21%)	1.8036 (14.49%)	2.4444 (36.91%)	**1.5422**
	80%	2.0191 (26.06%)	1.7985 (16.99%)	2.4378 (38.76%)	**1.4929**
Movielens	40%	1.6456 (23.12%)	1.5183 (16.67%)	1.7103 (26.03%)	**1.2652**
	60%	1.6439 (24.52%)	1.4985 (17.20%)	1.6924 (26.68%)	**1.2408**
	80%	1.6187 (23.17%)	1.4937 (16.73%)	1.6734 (25.68%)	**1.2437**
Zomato	40%	2.4874 (21.14%)	2.3380 (16.10%)	2.5496 (23.07%)	**1.9615**
	60%	2.4163 (20.11%)	2.2395 (13.80%)	2.5507 (24.32%)	**1.9304**
	80%	2.3791 (25.88%)	2.2021 (19.92%)	2.5695 (31.37%)	**1.7635**

on three real-world datasets, *i.e.,* Douban movie [1], MovieLens [2] and Zomato [3] datasets. Of these, the Douban movie and Zomato datasets were crawled from the related websites by us while the MovieLens dataset is a public dataset which is popularly used in the research of recommender systems.

The Douban movie dataset contains the records of users' actions during the period from January 2015 to April 2016 on the 100 most popular movies. It includes 877,572 ratings (1-5) and the time stamp data given by 249,408 users with a sparsity of 0.0330 which is calculated as $1 - \frac{\text{nonzero entries}}{\text{total entries}}$. The MovieLens dataset was collected from the MovieLens website during a seven-month period

[1] http://movie.douban.com
[2] http://www.grouplens.org/node/73
[3] https://www.zomato.com

Table 6.2: Performance Comparisons in terms of RMSE

Dataset	Training	online-MF	online-SocialMF	online-reciprocal	GODSR
Douban	40%	2.2909 (19.09%)	2.0053 (7.57%)	2.5360 (26.91%)	**1.8536**
	60%	2.1753 (19.83%)	2.0007 (12.83%)	2.5386 (31.30%)	**1.7440**
	80%	2.1352 (18.39%)	1.9893 (12.40%)	2.5241 (30.96%)	**1.7426**
Movielens	40%	1.9161 (28.29%)	1.8902 (27.31%)	1.8550 (25.92%)	**1.3741**
	60%	1.9162 (28.07%)	1.8970 (27.34%)	1.8636 (26.04%)	**1.3783**
	80%	1.9019 (28.13%)	1.8876 (27.58%)	1.8398 (25.70%)	**1.3670**
Zomato	40%	2.5987 (20.77%)	2.4979 (17.58%)	2.6587 (22.56%)	**2.0589**
	60%	2.3681 (12.71%)	2.2942 (9.90%)	2.6370 (21.61%)	**2.0671**
	80%	2.3264 (12.60%)	2.2479 (9.54%)	2.6285 (22.64%)	**2.0334**

from September 1997 to April 1998. The dataset consists of 100,000 ratings (1-5) from 943 users on 1,682 movies and each user has rated at least 20 movies with a sparcity of 0.0630. Detailed descriptions of the data file can be found in [51]. The Zomato dataset includes records of 5,336 users' actions on the most popular 1,012 restaurants in Sydney from September 2008 to April 2016 with a sparsity rate of 0.0031.

Metrics. We use Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) as the metrics in this chapter, which are the most popular metrics used to evaluate the deviation of recommendations from their true user-specified values. Specifically, they are defined as $RMSE = \sqrt{\frac{1}{N_{test}} \sum_{i,j} (R_{ij} - \hat{R}_{ij})^2}$ and $MAE = \frac{1}{N_{test}} \sum_{i,j} |R_{ij} - \hat{R}_{ij}|$ where r_{ij} denotes the real rate user i gives to item j, $\hat{r}_{ij}$ denotes the predicted rate user i gives to item j. N_{test} denotes the number of

rates in the test set. The lower the values of RMSE and MAE, the more accurate the predictions of the recommendation engine.

Experimental setup. In the general online dynamic social recommendation algorithm, only historical data but not future data can be used for current parameter learning. Following [71] [126], we split the training and testing data based on time and evaluate the performance. Specifically, we sort the entire data set in normal time order, and use the earlier part $(40\%, 60\%, 80\%$, respectively) as the training set and the remaining part as the testing set.

For parameter selection, we choose the best value of K through a grid search from 5 to 50 in steps of 20. And we choose the best value for the tradeoff parameters λ_1, λ_2, and λ_3 with grid search in $[10^{-4}, 10^4]$ with the step of 10^1. The learning step size is set to 0.01 empirically and the threshold parameter for the loss function is fixed as 1.

Benchmark methods. In order to show the effectiveness of our proposed recommendation approach, we compare the recommendation results of the following methods:

- **online-MF** is based on the baseline matrix factorization approach proposed in [110], which does not take the social network into account. We introduce online learning technique to it, turning it to online-MF;

- **online-SocialMF** is based on the method proposed in [64] which improves the recommendation accuracy of baseline matrix factorization approach by taking into account the social trust between users. It always uses all social links available in the dataset. We also introduce online learning technique to it, make it to online-SocialMF;

- **general online with reciprocal** is the baseline method which is same with GODSR except the implicit social network is built with the reciprocal technique but not the exponential technique.

- **GODSR**, which is proposed in this chapter, combines the social network inference, online learning, and collaborative filtering in a joint iterative process to make timely and accurate recommendations.

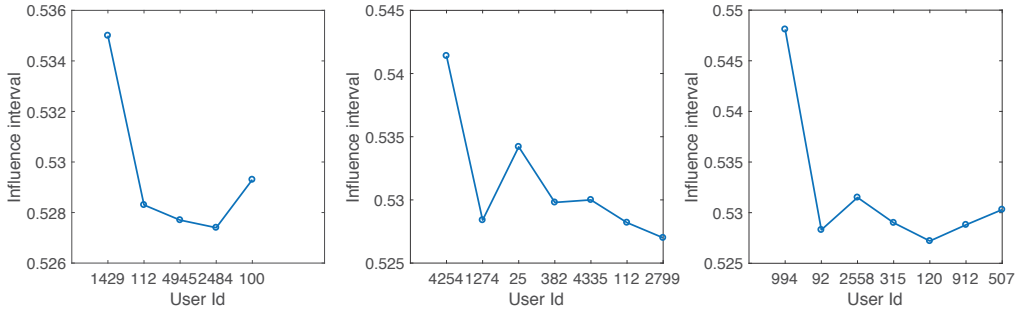
Result comparisons. Table 6.1 and 6.2 list the details of the performance comparison between the proposed GODSR algorithm and the benchmark methods. We can observe that our method consistently outperforms the others on the three datasets. Firstly, we can see our algorithm generates significantly better results than the Online-MF method which does not take the social network into account. There is an average 24.09% improvement in MAEs (max 28.21%, min 20.11%), and an average 20.87% improvement in RMSEs (max28.29%, min 12.60%) on the three datasets. This observation illustrates that employing social networks helps to increase the recommendation quality.

Furthermore, our method performs better than Online-SocialMF and Online-reciprocal. Specifically, the proposed GODSR achieves 15.78% improvement on average (max 19.92%, min 10.13%) in MAEs and 16.89% improvement on average (max 27.58%, min 7.57%) in RMSEs compared with Online-SocialMF. Compared with Online-reciprocal, our method achieves on average 29.57% (max 38.76%, min 23.07%) and 25.96% (max 31.30%, min 21.61%) improvement in MAEs and RMSEs, respectively. These observations demonstrate that general online dynamic social recommendation can explore a user's preferences and their core social network much more accurately while avoiding the noise contained in explicit social networks. Furthermore, timely feedback from the recommender system improves the performance of the recommendations.

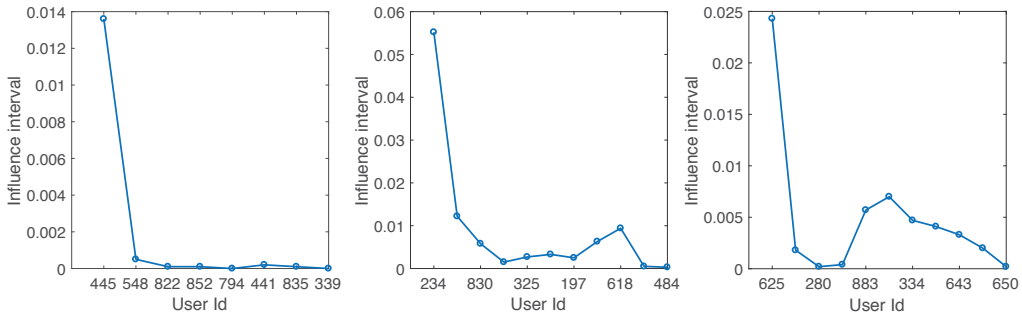
Further, we use the method proposed in Chapter II to learn the influence path among users. Fig 6.2 shows some samples on the three data sets. Confirmed with the inferred network, we can find the users that in the same influence paths have higher similarities among them.

6.5 Conclusion

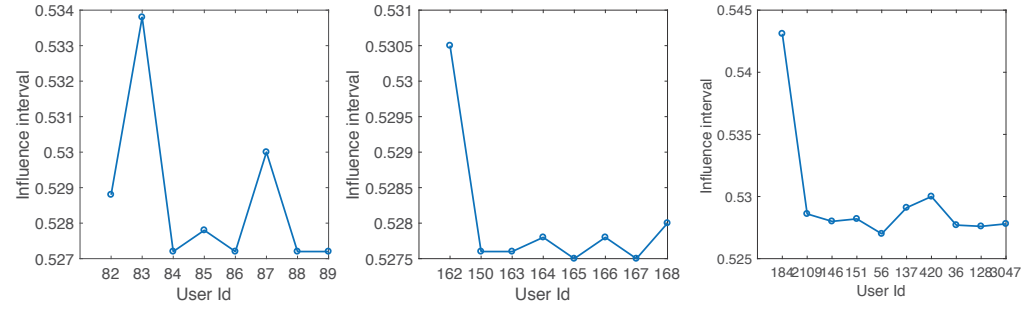
In this chapter, we aim to make timely and accurate recommendations with an inferred dynamic core social networks and to address the three concerns described in this chapter: the continuous drift of a user's preferences and needs, the dynamic core social networks of users and the problem of newly appearing users and items. A new General Online Dynamic Social Recommendation algorithm is proposed, which combines network inference, online learning and collaborative



(a) Douban



(b) Movielens



(c) Zomato

Figure 6.2: Examples for influence paths on the three data sets

filtering in an iterative process. It is different to most of the existing social recommendation approaches which tried to solve this problem in a defined specific scenario. Experiments on three real-world datasets demonstrate the effectiveness of the proposed model compared with other state-of-the-art solutions.

Part V

Conclusions and future works

Chapter 7

Conclusions and Future Works

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

7.1 Summary

Due to the potential value of social relations [125] [34], social recommendation has attracted a lot of attention recently in the research communities of information retrieval [147], machine learning [172] and data mining [99].

Modelling time drifting data is a central problem in social recommendation. Often, data is changing over time, and up to date modelling should be continuously updated to reflect its present nature. The analysis of such data needs to find the right balance between discounting temporary effects that have very low impact on future behavior, while capturing longer-term trends that reflect the inherent nature of the data[113] [134].

Especially, modelling temporal changes in customer preferences brings unique challenges. One kind of concept drift in this setup is the emergence of new products or services that change the focus of customers. Related to this are seasonal changes, or specific holidays, which lead to characteristic shopping patterns. All those changes influence the whole population, and are within the realm of traditional studies on concept drift. However, many of the changes in user behavior are driven by localized factors. For example, a change in the family structure can drastically change shopping patterns. Likewise, individuals gradually change

their taste in movies and music. All those changes cannot be captured by methods that seek a global concept drift. Instead, for each customer we are looking at different types of concept drifts, each occurs at a distinct time frame and is driven towards a different direction.

Hence, we propose a new Implicit Social Recommendation model in Chapter 3, which infers latent social network from cascade data. It can sufficiently mine the information contained in time by mining the cascade data and identify the dynamic changes in the users in time by using the latest updated social network to make recommendations. Experiments and comparisons on three real-world datasets show that the proposed model outperforms the state-of-the-art solutions in both explicit and implicit scenarios.

Another challenge is how to deal with the continuously appeared new users and new items in the system. In reality, this problem is unavoidable. New users and new items enter the recommender system continuously, so how to involve them and how to handle this problem smoothly must be considered. Most of the existing research either builds a general model in a sample scenario such as ignoring the dynamic nature of the system and the problem of new users and items, or only focuses on some specific fields, such as dynamic restaurant recommendation and timely news recommendation.

To this end, in this thesis, a novel General Online Dynamic Social Recommendation model is proposed in Chapter 6 to address these challenges simultaneously and effectively. Specifically, we combine network inference, online learning with the double-streaming data introduced in Chapter 5 and collaborative filtering together in an iterative process. In addition, we also apply the unsupervised time series feature learning algorithm introduced in Chapter 4 to learn the influence path among users to further explore the core social networks and improve the performance of the recommendation. By inferring the latent dynamic core social networks from cascade data, identifying the drift of a users preferences and involving new users and items in the online learning process, it reacts rapidly and make accurate recommendations to users when new data arrive. Experiments on three real-world datasets demonstrate the effectiveness of the proposed model compared with other state-of-the-art solutions.

7.2 Future Work

As more and more emerged applications of big data involve large volume, high velocity, and a variety of data sources, social recommendation has drawn and will continue to draw more and more attention in the research communities. Here, we outline some social recommendation problems that remains unexplored in the research community, from the perspective of dynamic social recommendation, as follows:

- **Detecting communities and the evolutions in dynamic social networks** Although a large body of work is devoted to finding communities in static social networks, only a few studies examined the dynamics of communities in evolving social networks. Dynamic stochastic block model for finding communities and their evolution in a dynamic social network should be explored. Unlike the existing approaches [144] [62] for modelling social networks that estimate parameters by their most likely values (i.e., point estimation), the model should capture the evolution of communities by explicitly modelling the transition of community memberships for individual nodes in the network.

- **Personalized recommendation on dynamic content** In Web-based services of dynamic content (such as news articles), recommender systems face the difficulty of timely identifying new items of high-quality and providing recommendations for new users [18]. Feature-based machine learning approach to personalized recommendation that is capable of handling the cold-start issue effectively should be further explored. The profiles of content of interest, in which temporal characteristics of the content, e.g. popularity and freshness, are updated in real-time manner, and also the profiles of users including demographic information and a summary of user activities. Based on all features in user and content profiles, the models to provide accurate personalized recommendations of new items for both existing and new users shuold be proposed and can be general and flexible for other personalized tasks.

- **Prediction and recommendation across heterogeneous social net-**

works Social network analysis is a fundamental problem in prediction and recommendation problem. The key challenge comes from the sparsity of networks due to the strong disproportion of links that they have potential to form to links that do form. Most previous work tries to solve the problem in single network, few research focus on capturing the general principles of link formation across heterogeneous networks. However, due to the intuition that people make friends in different networks with similar principles [27], the recommendation across heterogeneous networks should be explored which will effectively improve the predictive performance.

References

[1] N. Vasconcelos A.B. Chan and G.R.G. Lanckriet. Direct convex relaxations of sparse svm. *Proc. 24th Int'l Conf. Machine Learning*, pages 145–153, 2007. (Cited on page 24.)

[2] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, **53**:16–38, 2015. (Cited on page 22.)

[3] Ching-man Au Yeung and Tomoharu Iwata. Strength of social influence in trust networks in product review sites. In *WSDM*, pages 495–504, 2011. (Cited on pages 5 and 29.)

[4] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, **27**[9]:2522–2535, 2015. (Cited on pages 7 and 53.)

[5] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. A bag-of-features framework to classify time series. *IEEE transactions on pattern analysis and machine intelligence*, **35**[11]:2796–2802, 2013. (Cited on pages 6 and 52.)

[6] J. Bi, K.P. Bennett, M.J. Embrechts, C.M. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *J. Machine Learning Research*, **3**:1229–1243, 2003. (Cited on page 24.)

[7] V. Boln-Canedo, N. Snchez-Maroo, and A. Alonso-Betanzos. Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset. *Expert Systems with Applications*, **38**[5]:5947–5957, 2011. (Cited on page 114.)

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. (Cited on page 92.)

[9] Guy Bresler, George H Chen, and Devavrat Shah. A latent source model for online collaborative filtering. In *Advances in Neural Information Processing Systems(NIPS)*, pages 3347–3355, 2014. (Cited on pages 25 and 26.)

[10] Deng Cai, Chiyuan Zhang, and Xiaofei He. Unsupervised feature selection for multi-cluster data. In *KDD*, pages 333–342, 2010. (Cited on page 21.)

[11] Yuhan Cai and Raymond Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *SIGMOD*, pages 599–610, 2004. (Cited on page 52.)

[12] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *SIAM J. Comput.*, **34(3)**:640–668, 2005. (Cited on page 23.)

[13] Mustafa S Cetin, Abdullah Mueen, and Vince D Calhoun. Shapelet ensemble for multi-dimensional time series. *Quebec: SIAM SDM*, 2015. (Cited on page 65.)

[14] Allison JB Chaney, David M Blei, and Tina Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In *RecSys*, pages 43–50, 2015. (Cited on page 19.)

[15] Chih-Chung Chang and Chih-Jen Lin. Libsvm : a library for support vector machines. *ACM Trans. on Intell. Syst. and Tech.*, **2(27)**:1–27, 2011. (Cited on page 113.)

[16] Kai-Wei Chang, Biplab Deka, Wen-Mei W Hwu, and Dan Roth. Efficient pattern-based time series classification on gpu. In *ICDM*, pages 131–140, 2012. (Cited on pages 6 and 53.)

[17] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 815–824. ACM, 2016. (Cited on pages 10, 25, 26, and 119.)

[18] Wei Chu and Seung-Taek Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th international conference on World wide web*, pages 691–700. ACM, 2009. (Cited on page 137.)

[19] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *J. Machine Learning Research*, **7**:6551–6585, 2006. (Cited on pages 8, 23, 85, 87, 92, and 94.)

[20] K. Crammer, M. Dredze, and A. Kulesza. Multi-class confidence weighted algorithms. *In Empirical Methods in Natural Language Proc.*, pages 496–504, 2009. (Cited on pages 8, 23, and 85.)

[21] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. *in Proc. Adv. Neural Inf. Process. Syst.*, pages 414–422, 2009. (Cited on page 23.)

[22] K. Crammer and D. D. Lee. Learning via gaussian herding. *in Proc. Adv. Neural Inf. Process. Syst.*, pages 451–459, 2010. (Cited on page 23.)

[23] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of international conference on World Wide Web(WWW)*, pages 271–280. ACM, 2007. (Cited on pages 10, 25, 26, and 119.)

[24] Hoang Anh Dau and Nurjahan Begum Eamonn Keogh. Semi-supervision dramatically improves time series clustering under dynamic time warping. In *CIKM*, pages 978–987, 2016. (Cited on page 22.)

[25] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, **22**[1]:143–177, 2004. (Cited on page 25.)

[26] William E Donath and Alan J Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, **17**[5]:420–425, 1973. (Cited on page 57.)

[27] Yuxiao Dong, Jie Tang, Sen Wu, Jilei Tian, Nitesh V Chawla, Jinghai Rao, and Huanhuan Cao. Link prediction and recommendation across heterogeneous social networks. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 181–190. IEEE, 2012. (Cited on page 138.)

[28] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, **38**[9]:1734–1747, 2016. (Cited on page 21.)

[29] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *J. Machine Learning Research*, **10**:2899–2934, 2009. (Cited on page 24.)

[30] R.O. Duda, P.E. Hart, and D.G. Stork. Pattern classification. *Wiley-Interscience*, 2012. (Cited on page 24.)

[31] J.G. Dy and C.E. Brodley. Feature subset selection and order identification for unsupervised learning. *Proc. 17th Int'l Conf. Machine Learning*, pages 247–254, 2000. (Cited on page 24.)

[32] J.G. Dy and C.E. Brodley. Feature selection for unsupervised learning. *J. Machine Learning Research*, **5**:845–889, 2004. (Cited on page 24.)

[33] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, **45**[1]:12, 2012. (Cited on page 20.)

[34] Martin Ester. Recommendation in social networks. In *RecSys*, pages 491–492, 2013. (Cited on pages vii, 1, and 135.)

[35] Hui Fang, Yang Bao, and Jie Zhang. Leveraging decomposed trust in probabilistic matrix factorization for effective recommendation. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 30–36, 2014. (Cited on pages 5, 9, 19, 30, and 119.)

[36] Soude Fazeli, Babak Loni, Alejandro Bellogin, Hendrik Drachsler, and Peter Sloep. Implicit vs. explicit trust in social matrix factorization. In *Proceedings of ACM Conference on Recommender systems*, pages 317–320, 2014. (Cited on pages 5, 10, 19, 30, and 119.)

[37] Jordan Frank, Shie Mannor, Joelle Pineau, and Doina Precup. Time series analysis using geometric template matching. *IEEE transactions on pattern analysis and machine intelligence*, **35**[3]:740–754, 2013. (Cited on page 22.)

[38] Y. Freund and R.E. Schapire. Large margin classification using the perceptron algorithm. *J. Machine Learning Research*, **37(3)**:277–296, 1999. (Cited on page 23.)

[39] Li Gao, Jia Wu, Hong Yang, Zhi Qiao, Chuan Zhou, and Yue Hu. Semi-data-driven network coarsening. In *IJCAI*, pages 1483–1489, 2016. (Cited on pages 5 and 30.)

[40] C. Gentile. A new approximate maximal margin classifiction algorithm. *J. Machine Learning Research*, **2**:213–242, 2001. (Cited on page 23.)

[41] K.A. Glocer, D. Eads, and J. Theiler. Online feature selection for pixel classification. *Proc. 22th Int'l Conf. Machine Learning*, pages 249–256, 2005. (Cited on page 25.)

[42] Dian Gong, Gerard Medioni, and Xuemei Zhao. Structured time series analysis for human action segmentation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, **36**[7]:1414–1427, 2014. (Cited on page 52.)

[43] Songjie Gong. A collaborative filtering recommendation algorithm based on user clustering and item clustering. *JSW*, **5**[7]:745–752, 2010. (Cited on page 25.)

[44] Ian J Goodfellow, Aaron Courville, and Yoshua Bengio. Scaling up spike-and-slab models for unsupervised feature learning. *IEEE transactions on pattern analysis and machine intelligence*, **35**[8]:1902–1914, 2013. (Cited on page 22.)

[45] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. In *KDD*, pages 392–401, 2014. (Cited on pages 7, 20, 21, 53, 55, 56, 57, and 74.)

[46] Guibing Guo, Jie Zhang, Daniel Thalmann, Anirban Basu, and Neil Yorke-Smith. From ratings to trust: an empirical study of implicit trust in recommender systems. In *SAC*, pages 248–253, 2014. (Cited on pages 5, 19, and 30.)

[47] Guibing Guo, Jie Zhang, Daniel Thalmann, and Neil Yorke-Smith. Etaf: An extended trust antecedents framework for trust prediction. In *ASONAM*, pages 540–547, 2014. (Cited on page 20.)

[48] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *AAAI*, pages 123–129, 2015. (Cited on page 19.)

[49] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems*, **17**[2-3]:107–145, 2001. (Cited on page 66.)

[50] Yuan Hao, Yanping Chen, Jesin Zakaria, Bing Hu, Thanawin Rakthanmanon, and Eamonn Keogh. Towards never-ending learning from time series streams. In *KDD*, pages 874–882, 2013. (Cited on page 21.)

[51] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, **5**[4]:19, 2015. (Cited on pages 1, 41, and 129.)

[52] Qing He, Fuzhen Zhuang, Tianfeng Shang, Zhongzhi Shi, et al. Fast time series classification based on infrequent shapelets. In *ICML*, **1**, pages 215–219, 2012. (Cited on page 21.)

[53] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. *in Proc. Adv. Neural Inf. Process. Syst.*, 2005. (Cited on page 24.)

[54] Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. In *NIPS*, pages 507–514, 2005. (Cited on page 21.)

[55] Patrick Héas and Mihai Datcu. Modeling trajectory of dynamic clusters in image time-series for spatio-temporal reasoning. *IEEE Transactions on Geoscience and Remote Sensing*, **43**[7]:1635–1647, 2005. (Cited on page 52.)

[56] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, **28**[4]:851–881, 2014. (Cited on page 20.)

[57] Shoji Hirano and Shusaku Tsumoto. Cluster analysis of time-series medical data based on the trajectory representation and multiscale comparison techniques. In *ICDM*, pages 896–901, 2006. (Cited on page 52.)

[58] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266. ACM, 2003. (Cited on page 25.)

[59] Thomas Hofmann and D Hartmann. Collaborative filtering with privacy via factor analysis. In *Proceedings of the ACM symposium on applied computing(SAC)*, pages 791–795, 2005. (Cited on page 25.)

[60] Steven C.H. Hoi, Jialei Wang, Peilin Zhao, and Ji Wan. Libol: A library for online learning algorithms. *Nanyang Technological Univ.*, 2012. (Cited on page 23.)

[61] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, **22**[1]:116–142, 2004. (Cited on pages 17 and 18.)

[62] Matthew O Jackson and Alison Watts. The evolution of social and economic networks. *Journal of Economic Theory*, **106**[2]:265–295, 2002. (Cited on page 137.)

[63] Mohsen Jamali and Martin Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *KDD*, pages 397–406, 2009. (Cited on pages 4, 19, and 28.)

[64] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the ACM conference on Recommender systems*, pages 135–142, 2010. (Cited on pages 4, 19, 28, 45, and 130.)

[65] Rong Jin, Joyce Y Chai, and Luo Si. An automatic weighting scheme for collaborative filtering. In *Proceedings of the international ACM SIGIR conference on Research and development in information retrieval(SIGIR)*, pages 337–344. ACM, 2004. (Cited on page 25.)

[66] E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering homepage www.cs.ucr.edu/∼eamonn/time_series_data/. (Cited on pages 7, 53, and 65.)

[67] Eamonn Keogh, Jessica Lin, Ada Waichee Fu, and Helga VanHerle. Finding unusual medical time-series subsequences: Algorithms and applications. *IEEE Transactions on Information Technology in Biomedicine*, **10**[3]:429–439, 2006. (Cited on page 52.)

[68] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, **132(1)**:1–63, 1997. (Cited on pages 8 and 84.)

[69] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, **53**[4]:89–97, 2010. (Cited on page 44.)

[70] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *J. Machine Learning Research*, **10**:777–801, 2009. (Cited on pages 8, 24, 84, 85, and 86.)

[71] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the ACM international conference on Web search and data mining(WSDM)*, pages 297–306, 2011. (Cited on pages 44 and 130.)

[72] Yuhong Li, Man Lung Yiu, Zhiguo Gong, et al. Efficient discovery of longest-lasting correlation in sequence databases. *The VLDB Journal*, pages 1–24. (Cited on page 22.)

[73] Zechao Li, Jing Liu, Jinhui Tang, and Hanqing Lu. Robust structured subspace learning for data representation. *IEEE transactions on pattern analysis and machine intelligence*, **37**[10]:2085–2098, 2015. (Cited on page 21.)

[74] Zechao Li, Jing Liu, Yi Yang, Xiaofang Zhou, and Hanqing Lu. Clustering-guided sparse structural learning for unsupervised feature selection. *IEEE Transactions on Knowledge and Data Engineering*, **26**[9]:2138–2150, 2014. (Cited on page 21.)

[75] Zechao Li and Jinhui Tang. Unsupervised feature selection via nonnegative spectral analysis and redundancy control. *IEEE Transactions on Image Processing*, **24**[12]:5343–5355, 2015. (Cited on page 21.)

[76] Zechao Li, Yi Yang, Jing Liu, Xiaofang Zhou, Hanqing Lu, et al. Unsupervised feature selection using nonnegative spectral analysis. In *AAAI*, pages 1026–1032, 2012. (Cited on pages 21 and 72.)

[77] Chen Lin, Runquan Xie, Xinjun Guan, Lei Li, and Tao Li. Personalized news recommendation via implicit social experts. *Information Sciences*, **254**:1–18, 2014. (Cited on pages 5, 20, and 30.)

[78] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. In *KDD*, pages 289–297, 2012. (Cited on pages 20, 56, and 57.)

[79] Guang Ling, Haiqin Yang, Irwin King, and Michael R Lyu. Online learning for collaborative filtering. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012. (Cited on page 6.)

[80] Nathan N Liu and Qiang Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *Proceedings of the international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90. ACM, 2008. (Cited on page 25.)

[81] Hao Ma. An experimental study on implicit social recommendation. In *SIGIR*, pages 73–82, 2013. (Cited on pages 1, 5, 20, 30, and 45.)

[82] Hao Ma, Irwin King, and Michael R Lyu. Learning to recommend with social trust ensemble. In *SIGIR*, pages 203–210, 2009. (Cited on page 19.)

[83] Hao Ma, Irwin King, and Michael R Lyu. Learning to recommend with explicit and implicit social relations. *ACM Transactions on Intelligent Systems and Technology (TIST)*, **2**[3]:29:1–29:19, 2011. (Cited on page 20.)

[84] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*, pages 931–940, 2008. (Cited on pages 4, 17, 18, 19, and 28.)

[85] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of ACM international conference on Web search and data mining (WSDM)*, pages 287–296, 2011. (Cited on pages 4, 5, 17, 18, 19, 25, 28, 30, 36, and 45.)

[86] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious urls: An application of large-scale online learning. *Proc. 26th Int'l Conf. Machine Learning*, 2009. (Cited on page 113.)

[87] Sara C Madeira, Miguel C Teixeira, Isabel Sa-Correia, and Arlindo L Oliveira. Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **7**[1]:153–165, 2010. (Cited on page 66.)

[88] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *RecSys*, pages 17–24, 2007. (Cited on pages 17 and 19.)

[89] P. Mitra, C. A. Murthy, and S. Pal. Unsupervised feature selection using feature similarity. *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**:301–312, 2002. (Cited on page 24.)

[90] Abdullah Mueen, Eamonn Keogh, and Neal Young. Logical-shapelets: an expressive primitive for time series classification. In *KDD*, pages 1154–1162, 2011. (Cited on pages 6, 21, and 53.)

[91] Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. Efficient and robust feature selection via joint l2, 1-norms minimization. In *NIPS*, pages 1813–1821, 2010. (Cited on page 22.)

[92] F. Orabona and K. Crammer. New adaptive algorithms for online classification. *in Proc. Adv. Neural Inf. Process. Syst.*, pages 1840–1848, 2010. (Cited on page 23.)

[93] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*, pages 1855–1870, 2015. (Cited on pages 6, 11, 22, 52, 54, 71, and 72.)

[94] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007. (Cited on page 25.)

[95] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1226–1238, 2005. (Cited on page 24.)

[96] Zhang Peng, Zhou Chuan, Wang Peng, Gao Byron J., Zhu Xingquan, and Guo Li. E-tree: An efficient indexing structure for ensemble models on data streams. *IEEE Transactions on Knowledge Data Engineering*, **27**[2]:461–474, 2015. (Cited on page 23.)

[97] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, **44**[3]:678–693, 2011. (Cited on page 72.)

[98] Mingjie Qian and Chengxiang Zhai. Robust unsupervised feature selection. In *IJCAI*, pages 1621–1627, 2013. (Cited on pages 22 and 72.)

[99] Xueming Qian, He Feng, Guoshuai Zhao, and Tao Mei. Personalized recommendation combining user interest and social circle. *IEEE Transactions on Knowledge and Data Engineering*, **26**[7]:1763–1777, 2014. (Cited on pages 1 and 135.)

[100] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. (Cited on page 24.)

[101] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *KDD*, pages 262–270, 2012. (Cited on page 20.)

[102] Thanawin Rakthanmanon and Eamonn Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *SDM*, pages 668–676, 2013. (Cited on pages 6, 21, and 53.)

[103] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, **66**[336]:846–850, 1971. (Cited on page 66.)

[104] Jasson DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, pages 713–719, 2005. (Cited on page 36.)

[105] Manuel Gomez Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697*, 2011. (Cited on pages 32, 34, 38, 121, and 123.)

[106] Adriana Romero, Petia Radeva, and Carlo Gatta. Meta-parameter free unsupervised sparse feature learning. *IEEE transactions on pattern analysis and machine intelligence*, **37**[8]:1716–1722, 2015. (Cited on page 21.)

[107] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Rev.*, **65**:386–407, 1958. (Cited on pages 8, 23, 85, and 107.)

[108] Eduardo J Ruiz, Vagelis Hristidis, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Correlating financial time series with micro-blogging activity. In *WSDM*, pages 513–522, 2012. (Cited on page 52.)

[109] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887, 2008. (Cited on page 36.)

[110] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems(NIPS)*, **20**, pages 1–8, 2011. (Cited on pages 18, 25, 36, 45, and 130.)

[111] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of international conference on World Wide Web(WWW)*, pages 285–295, 2001. (Cited on pages 25 and 44.)

[112] Ray E Schafer. Statistical models and methods for lifetime data. *Technometrics*, **25**[1]:111–112, 1983. (Cited on page 34.)

[113] Jeffrey C Schlimmer and Richard H Granger. Beyond incremental processing: Tracking concept drift. In *AAAI*, pages 502–507, 1986. (Cited on pages 2 and 135.)

[114] Shahriar Shariat and Vladimir Pavlovic. Isotonic cca for sequence alignment and activity recognition. In *ICCV*, pages 2572–2578, 2011. (Cited on pages 75 and 78.)

[115] Lei Shi, Liang Du, and Yi-Dong Shen. Robust spectral learning for unsupervised feature selection. In *ICDM*, pages 977–982, 2014. (Cited on pages 21 and 72.)

[116] Luo Si and Rong Jin. Flexible mixture model for collaborative filtering. In *Proceedings of tinternational conference on machine learning (ICML)*, **3**, pages 704–711, 2003. (Cited on page 25.)

[117] M. R. Sikonja and I. Kononenko. Theoretical and empirical analysis of relief and relieff. *J. Machine Learning Research*, **53**:23–69, 2003. (Cited on page 24.)

[118] L. Song, A. Smola, A. Fretton, K. Borgwardt, and J. Bedo. Supervised feature selection via dependence estimation. *Proc. 24th Int'l Conf. Machine Learning*, 2007. (Cited on page 24.)

[119] David H Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of international conference on World wide web (WWW)*, pages 111–120. ACM, 2009. (Cited on page 26.)

[120] Jos A. Sez, Mikel Galar, Julin Luengo, and Francisco Herrera. Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. *Knowledge and Information Systems*, **38**[1]:179–206, 2014. (Cited on page 114.)

[121] N.L. C. Talbot, G. C. Cawley, and M. Girolami. Sparse multimonial logistic regression via bayesian l1 regularisation. *in Proc. Adv. Neural Inf. Process. Syst.*, 2006. (Cited on page 24.)

[122] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data Classification: Algorithms and Applicaions Boca Raton, FL USA: CRC Press*, 2014. (Cited on page 24.)

[123] Jiliang Tang, Xia Hu, and Huan Liu. Social recommendation: a review. *Social Network Analysis and Mining*, **3**[4]:1113–1133, 2013. (Cited on pages 17 and 18.)

[124] Jiliang Tang and Huan Liu. An unsupervised feature selection framework for social media data. *IEEE Transactions on Knowledge and Data Engineering*, **26**[12]:2914–2927, 2014. (Cited on page 57.)

[125] Jiliang Tang, Jie Tang, and Huan Liu. Recommendation in social media: recent advances and new frontiers. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1977–1977. ACM, 2014. (Cited on pages vii, 1, and 135.)

[126] Xiangyu Tang and Jie Zhou. Dynamic personalized recommendation on sparse data. *IEEE Transactions on Knowledge and Data Engineering(TKDE)*, **25**[12]:2895–2899, 2013. (Cited on pages 5, 30, 44, and 130.)

[127] Stephen J Taylor. Modeling financial time series. *World Scientific Publishing*, 2007. (Cited on page 52.)

[128] Isaac Triguero, Salvador Garca, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, **42**[2]:245–284, 2015. (Cited on page 114.)

[129] Liudmila Ulanova, Nurjahan Begum, and Eamonn Keogh. Scalable clustering of time series with u-shapelets. In *SDM*, 2015. (Cited on page 22.)

[130] Pramod K Vemulapalli, Vishal Monga, and Sean N Brennan. Robust extrema features for time-series data analysis. *IEEE transactions on pattern analysis and machine intelligence*, **35**[6]:1464–1479, 2013. (Cited on pages 7 and 53.)

[131] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, **17**[4]:395–416, 2007. (Cited on page 57.)

[132] Jialei Wang, Peilin Zhao, Steven C.H. Hoi, and Ji Wan. Online feature selection and its applications. *IEEE Trans. Knowl. and Data Eng.*, **26(3)**:698–710, 2014. (Cited on pages 8, 23, 24, 84, 85, 86, 93, and 109.)

[133] J. Weston, A. Elisseff, B. Schoelkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *J. Machine Learning Research*, **3**:1439–1461, 2003. (Cited on page 24.)

[134] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, **23**[1]:69–101, 1996. (Cited on pages 2 and 135.)

[135] Martin Wistuba, Josif Grabocka, and Lars Schmidt-Thieme. Ultra-fast shapelets for time series classification. *arXiv preprint arXiv:1503.05018*, 2015. (Cited on page 20.)

[136] X. Wu, K. Yu, H. Wang, and W. Ding. Online streaming feature selection. *Proc. 27th Int'l Conf. Machine Learning*, pages 1159–1166, 2010. (Cited on pages 8, 25, 84, and 86.)

[137] Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. Online feature selection with streaming features. *IEEE Trans. Pattern Anal. Mach. Intell.*, **35(5)**:1178–1192, 2013. (Cited on pages 8, 25, and 86.)

[138] Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. Online feature selection with streaming features. *IEEE transactions on pattern analysis and machine intelligence*, **35**[5]:1178–1192, 2013. (Cited on page 22.)

[139] Xin Xin, Irwin King, Hongbo Deng, and Michael R Lyu. A social recommendation framework based on multi-scale continuous conditional random fields. In *CIKM*, pages 1247–1256, 2009. (Cited on page 19.)

[140] Z. Xu, R. Jin, J. Ye, M. Lyu, and I. King. Discriminative semi-supervised feature selection via manifold regularization. *Proc. 20th Int'l Joint Conf. Artificial Intelligence*, 2009. (Cited on page 24.)

[141] Z. Xu, R. Jin, J. Ye, M.R. Lyu, and I. King. Non-monotonic feature selection. *Proc. 26th Int'l Conf. Machine Learning*, page 144, 2009. (Cited on page 24.)

[142] Jaewon Yang and Jure Leskovec. Patterns of temporal variation in online media. In *WSDM*, pages 177–186, 2011. (Cited on page 72.)

[143] L. Yang, R. Jin, and J. Ye. Online learning by ellipsoid method. *Proc. 26th Int'l Conf. Machine Learning*, page 145, 2009. (Cited on page 23.)

[144] Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. Detecting communities and their evolutions in dynamic social net-

worksa bayesian approach. *Machine learning*, **82**[2]:157–189, 2011. (Cited on page 137.)

[145] Xiwang Yang, Yang Guo, Yong Liu, and Harald Steck. A survey of collaborative filtering based social recommender systems. *Computer Communications*, **41**:1–10, 2014. (Cited on page 18.)

[146] Xiwang Yang, Harald Steck, Yang Guo, and Yong Liu. On top-k recommendation using social networks. In *RecSys*, pages 67–74, 2012. (Cited on page 18.)

[147] Xiwang Yang, Harald Steck, and Yong Liu. Circle-based recommendation in online social networks. In *KDD*, pages 1267–1275, 2012. (Cited on pages 1, 18, and 135.)

[148] Y. Yang, H.T. Shen, Z. Ma, Z. Huang, and X. Zhou. $l_{2,1}$-norm regularized discriminative feature selection for unsupervised learning. *Proc. 22th Int'l Joint Conf. Artificial Intelligence*, pages 1589–1594, 2011. (Cited on page 24.)

[149] Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. l2, 1-norm regularized discriminative feature selection for unsupervised learning. In *IJCAI*, **22**, page 1589, 2011. (Cited on pages 21 and 72.)

[150] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *KDD*, pages 947–956, 2009. (Cited on pages 6, 20, 21, and 52.)

[151] Lexiang Ye and Eamonn Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data mining and knowledge discovery*, **22**[1-2]:149–182, 2011. (Cited on page 21.)

[152] Hilmi Yildirim and Mukkai S Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys*, pages 131–138, 2008. (Cited on page 18.)

[153] Guoxian Yu, Guoji Zhang, Zili Zhang, Zhiwen Yu, and Lin Deng. Semi-supervised classification based on subspace sparse representation. *Knowledge and Information Systems*, **43**[1]:81–101, 2015. (Cited on page 114.)

[154] Kai Yu, Shenghuo Zhu, John Lafferty, and Yihong Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *SIGIR*, pages 211–218, 2009. (Cited on page 36.)

[155] Jesin Zakaria, Abdullah Mueen, and Eamonn Keogh. Clustering time series using unsupervised-shapelets. In *ICDM*, pages 785–794, 2012. (Cited on pages 6, 11, 22, 52, 54, 66, 71, 72, and 75.)

[156] Jesin Zakaria, Abdullah Mueen, Eamonn Keogh, and Neal Young. Accelerating the discovery of unsupervised-shapelets. *Data mining and knowledge discovery*, **30**[1]:243–281, 2016. (Cited on page 22.)

[157] K. Zhai and J. Boyd-Graber. Online latent dirichlet allocation with infinite vocabulary. *Proc. 30th Int'l Conf. Machine Learning*, **28**, 2013. (Cited on pages 7 and 84.)

[158] Hui Zhang, Tu Bao Ho, Yang Zhang, and M-S Lin. Unsupervised feature extraction for time series clustering using orthogonal wavelet transform. *Informatica*, **30**[3], 2006. (Cited on page 66.)

[159] Peng Zhang, Byron J Gao, Ping Liu, Yong Shi, and Li Guo. A framework for application-driven classification of data streams. *Neurocomputing*, **92**:170–182, 2012. (Cited on pages 8 and 84.)

[160] Peng Zhang, Chuan Zhou, Peng Wang, Byron J Gao, Xingquan Zhu, and Li Guo. E-tree: An efficient indexing structure for ensemble models on data streams. *Knowledge and Data Engineering, IEEE Transactions on*, **27**[2]:461–474, 2015. (Cited on pages 8 and 84.)

[161] Qin Zhang, Jia Wu, Hong Yang, Yingjie Tian, and Chengqi Zhang. Unsupervised feature learning from time series. In *IJCAI*, pages 2322–2328, 2016. (Cited on pages 5, 13, 15, 20, and 30.)

[162] Qin Zhang, Jia Wu, Peng Zhang, Guodong Long, Ivor W. Tsang, and Chengqi Zhang. Inferring latent network from cascade data for dynamic social recommendation. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*, pages 669–678, 2016. (Cited on pages 9, 13, 15, and 117.)

[163] Qin Zhang, Peng Zhang, Guodong Long, Wei Ding, Chengqi Zhang, and Xindong Wu. Towards mining trapezoidal data streams. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 1111–1116. IEEE, 2015. (Cited on pages 14 and 15.)

[164] Qin Zhang, Peng Zhang, Guodong Long, Wei Ding, Chengqi Zhang, and Xindong Wu. Online learning from trapezoidal data streams. *IEEE Transactions on Knowledge and Data Engineering(TKDE)*, **28**[10]:2709–2723, 2016. (Cited on pages 5, 14, 15, and 30.)

[165] Yi Zhang and Jonathan Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of international ACM SIGIR conference on Research and development in information retrieval*, pages 47–54. ACM, 2007. (Cited on page 25.)

[166] Z. Zhao and H. Liu. Semi-supervised feature selection via spectral analysis. *Proc. SIAM Int'L Conf. Data Mining*, 2007. (Cited on page 24.)

[167] Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. *Proc. 24th Int'l Conf. Machine Learning*, pages 1151–1157, 2007. (Cited on page 24.)

[168] Zheng Zhao and Huan Liu. Spectral feature selection for supervised and unsupervised learning. In *ICML*, pages 1151–1157, 2007. (Cited on page 21.)

[169] Zheng Zhao, Lei Wang, and Huan Liu. Efficient spectral feature selection with minimum redundancy. In *AAAI*, pages 673–678, 2010. (Cited on page 21.)

[170] Feng Zhou and Fernando De la Torre. Generalized canonical time warping. *IEEE transactions on pattern analysis and machine intelligence*, **38**[2]:279–294, 2016. (Cited on page 52.)

[171] Y. Zhou, R. Jin, and S.C.H. Hoi. Exclusive lasso for multi-task feature seleciton. *J. Machine Learning Research*, **9**:988–995, 2010. (Cited on page 24.)

[172] Jianke Zhu, Hao Ma, Chun Chen, and Jiajun Bu. Social recommendation using low-rank semidefinite program. In *AAAI*, pages 158–163, 2011. (Cited on pages 1 and 135.)

[173] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. *Proc. 20th Int'l Conf. Machine Learning*, pages 928–936, 2003. (Cited on page 23.)