Machine Learning and Data Mining Methods for Recommender Systems and Chemical Informatics

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Xia Ning

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Dr. George Karypis, Advisor

July, 2012

© Xia Ning 2012
ALL RIGHTS RESERVED

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school. First and foremost, I would like to dedicate this thesis to my parents. During the time of my graduate study, they had suffered the biggest pains physically and emotionally, but they freed me from all the responsibilities as a daughter and instead held all the weights on their own shoulders. Sorry, Dad and Mom, I wish I would have done something for you. I hope all my years of concentration and hard work could make you a little bit happier and satisfied.

I owe the deepest debt of gratitude to my advisor Prof. George Karypis. He is much more than just an academic advisor, a mentor and a colleague to me. Over the years, he has not only provided me with the best learning and research environment, enlightening advises, the strongest support and numerous great opportunities, but also helped me open my eyes to different research areas, build up my confidence and self-esteem and lay down my career path. I am deeply graceful for all the encouragement, patience and cares he offers during my graduate study. I have learned from both of his professional and personal behaviors and attitudes much more than I ever imagined. He is a perfect role model of my life.

My special thanks go to the Olson family for their unconditional love and support. They are my family in the states. Their caring and company make me feel never lonely on each of the holidays. Thank you, Jerry, Barbara, Philip, Christina, Nancy, Mike and Priscilla. It has been a great blessing to have all of you in my life.

I also owe special thanks to Prof. Vipin Kumar, Prof. Michael Walters, Prof. John Riedl, Prof. Birgit Grund, Prof. Ravi Janardan and Prof. Douglas Hawkins for their invaluable assistance, feedback, criticisms, and advises at different stages of my graduate study.

It has been a fortune to work with all the intelligent people in the Karypis Lab: Asmaa El Badrawy, Ahmed Rezwan, Chris Kauffman, David Anastasiu, Dominique Lasalle, Evangelia Christakopoulou, Jeremy Iverson, Kevin DeRonne, Santosh Kabbur, Sara Morsy, Yevgeniy

Podolyan and Zhonghua Jiang. Thank you for all your help over the years. I have learned a lot from you all. I also thank the previous Karypis lab members: Andrea Tagarelli, Christian Desrosiers, Huzefa Rangwala, Nikil Wale, Michihiro Kuramochi, Ying Zhao, Eui-Hong Han, Zi Lin, Fuzhen Zhuang and Aris Gkoulalas-Divanis, for all their suggestions, advises and help.

Last but not the least, thank you, Grandpa, for your unconditional and endless love since I was a little kid. I wish I could have seen you one last time. May you have peace in Heaven.

Dedication

To the anonymous kidney donor

Abstract

This thesis focuses on machine learning and data mining methods for problems arising primarily in recommender systems and chemical informatics. Although these two areas represent dramatically different application domains, many of the underlying problems have common characteristics, which allows the transfer of ideas and methods between them.

The first part of this thesis focuses on recommender systems. Recommender systems represent a set of computational methods that produce recommendations of interesting entities (e.g., products) from a large collection of such entities by retrieving/filtering/learning information from their own properties (e.g., product attributes) and/or the interactions with other parties (e.g., user-product ratings).

We have addressed the two core tasks for recommender systems, that is, *top-N recommendation* and *rating prediction*. We have developed 1). *a novel sparse linear method for top-N recommendation,* which utilizes regularized linear regression with sparsity constraints to model user-item purchase patterns; 2). *a set of novel sparse linear methods with side information for top-N recommendation,* which use side information to regularize sparse linear models or use side information to model user-item purchase behaviors; and 3). *a multi-task learning method for rating prediction*, which uses multi-task learning methodologies to model user communities and predict personalized ratings.

The second part of this thesis is dedicated to chemical informatics, which is an interdisciplinary research area where computational and information technologies are developed to aid the investigation of chemical problems.

We have developed computational methods to build two important models in chemical informatics, that is, *Structure-Activity-Relationship (SAR)* model and *Structure-Selectivity-Relationship (SSR)* model. We have developed 1). *a multi-assay-based SAR model*, which leverages information from different protein families; and 2). *a set of computational methods for better SSR models*, which use various learning methodologies including multi-class classification and multi-task learning.

The studies on recommender systems and chemical informatics show that these two areas have great analogies in terms of the data, the problem formulations and the underlying principles, and any advances in one area could contribute to that of the other.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis focuses on the development of machine learning and data mining methods for the problems that arise primarily in the areas of recommender systems and chemical informatics. Although these two areas represent dramatically different application domains (i.e., e-commerce systems and drug discovery), many of the underlying problems have common characteristics, which allows the transfer of ideas and methods between them. The high-level commonalities and connections of the two areas are identified and discussed in Section 2.3, but each of two areas is discussed within its own context. This thesis is explicitly divided into two parts, one for recommender systems and the other for chemical informatics.

Needs for Recommender Systems and Algorithms The emergence and fast growth of E-commerce have significantly changed people's traditional perspective on purchasing products by providing huge amounts of products and detailed product information, thus making online transactions much easier. However, as the number of products conforming to the customers' desires has dramatically increased, the problem has become how to effectively and efficiently help customers identify the products that best fit their personal tastes. This leads to the widely used recommender systems. Recommender systems have emerged as a key enabling technology for E-commerce. They perform as virtual experts who are keenly aware of the user preferences and desires, and correspondingly filter out vast amount of irrelevant information in order to identify and recommend the most relevant products to the users. During the last decade, there have been numerous algorithms developed for recommender systems. However, the state-of-the-art algorithms developed from academia either achieve high recommendation performance

with a big sacrifice of run-time performance, or achieve online recommendation efficiency but have low recommendation qualities. Both the effectiveness and efficiency of recommender systems are addressed in this thesis and new recommendation methods are developed to achieve both of them concurrently. In addition, the accessibility of other product information such as product reviews, product description and the social networks over users has provided a great opportunity for the improvement of recommender systems. Currently, a lot of efforts have been devoted to incorporating such information into conventional recommender systems. However, there still lacks effective methods that can achieve significant improvement. In this thesis, the problem of incorporating additional information into recommender systems is addressed and a set of effective methods are presented.

Needs for Computational Tools for Chemical Informatics Discovery, design and development of new drugs is an expensive and challenging process. A successful new drug should not only produce the desired response to the disease but should also do so with minimal side effects. Finding such drugs is a laborious and multi-step process whose success depends on a number of different factors. In this situation, computational methods that can help accurately analyze existing experimental results and generate reliable predictions are highly desired. On the other hands, with the availability and accessibility of different experimental data including high-throughput screening assays, dose-response assays and drug-target interaction networks, there is a great chance of building accurate computational models from such data and helping accelerate the process. This thesis addresses machine learning and data mining methods developed for aiding in the early stage of drug discovery, in particular, to computationally identify active and selective drug candidates that should further go though other optimization and test. New computational methods are developed which focus on utilizing additional information during learning for better predictive models.

1.1 Key Contributions

1.1.1 Recommender Systems

Sparse Linear Methods (SLIM) for *Top-N* Recommendation *Top-N* recommendations aim to identify a short list of items from a huge amount of items that most fit a user's personal interests and desires. In recent years, *top-N* recommendations have gained great popularities and

attentions due to the dramatic increase of products and services online that can satisfy people's needs from various perspectives. The development of efficient and effective *top-N* recommendation algorithms that scale to large real problems and generate high-quality recommendations efficiently is highly desired and also highly challenging.

In this thesis (Chapter 4), a new Sparse Linear Method (SLIM) for *top-N* recommendation is presented. The contribution from SLIM has two folds. First, it achieves significantly better recommendation accuracy than the state-of-the-art methods. SLIM is completely different from the existing models since it utilizes regularized linear regression with sparsity constraints to model the user-item purchase patterns such that it can better recover user preferences over items and generate high-quality recommendations. Second, due to the sparsity of SLIM model, it requires fewer computations to generate recommendations than the state-of-the-art methods, in addition to fewer space for model storage. This feature makes it very suitable for large-scale online recommendation tasks, and makes it more practically useful than other recommendation algorithms.

Sparse Linear Methods with Side Information (SSLIM) for *Top-N* Recommendation The increasing availability of additional information associated with items (referred to as *side information*) provides a great opportunity for recommender systems to get better recommendation performance by having side information incorporated. This is because the side information represents a rich source of information and knowledge that may better help understanding and reasoning of user-item purchase patterns.

In this thesis (Chapter 5), a set of new *top-N* recommendation methods (SSLIM) that effectively utilize side information is presented. These methods either use side information to regularize the SLIM method by exploring the relations between side information and user-item purchase patterns, or use side information within the SLIM model directly. These SSLIM methods are distinct from other existing recommendation methods that use side information in the ways of how they incorporate side information and how they bias the baseline recommendation methods with side information. It is demonstrated that SSLIM achieve better recommendation accuracy than the state-of-the-art methods using side information.

Multi-Task Learning for Rating Prediction Rating prediction in recommender systems aims to predict how a user likes a certain item, and it has been a very active research area over the

years. The state-of-the-art rating prediction methods tend to model the users and items in low-dimension latent space, which typically achieve good recommendation performance but significantly lack model interpretability. In addition, there are not many algorithms developed that address the local properties of user-item purchase patterns.

The contribution of this thesis (Chapter 6) on rating prediction focuses on that it presents a rating prediction method which explicitly utilizes user neighborhood and item neighborhood properties, and thus it provides high model interpretability. Additionally, it provides a multi-task learning framework which is able to effectively learn and incorporate the information from user neighbors and item neighbors that strengthens the signals that contribute to accurate rating predictions. Moreover, this multi-task learning framework can be easily adopted for problems that arise in chemical informatics.

1.1.2 Chemical Informatics

Multi-Assay-based Structure-Activity-Relationship Models Structure-Activity-Relationship (SAR) models have been serving as the major tool to identify the compounds that show desirable activity properties against a target under consideration. Over the years, many different computational SAR algorithms have been developed, but most of them are dominated by the chemogenomics methodology, that is, only the proteins from a same protein family as the target under consideration are somehow utilized, while all the information from different protein families is ignored.

The contribution of this thesis (Chapter 8) on SAR modeling is that a new SAR modelling (i.e., multi-assay based) methodology is presented. The method uses all related proteins and their SAR information together and utilizes multi-task learning methods to build better SAR models. The related proteins are identified based on their SAR information, and they can be from different protein families. In this way, a much larger protein space and thus a richer source of useful information is considered. The multi-assay based SAR method is the first method in literature that considers information outside protein families and it is demonstrated that it achieves significantly better results than the chemogenomics methods.

Improved Machine Learning Methods for Structure-Selectivity-Relationship Models Computational Structure-Selectivity-Relationship (SSR) models are important tools to find compounds that are not only active, but also selective against a target of consideration. Unlike

SAR models, there are very limited methods developed for building effective SSR models. The existing SSR models mainly focus on identifying selectivity from activity properties.

In this thesis (Chapter 9), a set of SSR models is presented. The major contribution of these new SSR models is that they consider and utilize both activity and selectivity properties, and a multi-task learning model is constructed such that the activity and the selectivity properties are differentiated during model learning such that more accurate selectivity predictions can be made. It is demonstrated that the multi-task based SSR models outperform the existing SSR models.

1.2 Outline

The thesis is organized as follows:

In Chapter 2, an introduction to recommender systems and chemical informatics, as well as related learning algorithms, is presented. This chapter provides background knowledge and overview of the problems that the rest of this thesis will focus on. In particular, it provides a description on the relations between recommender systems and chemical informatics, which connects the rest chapters.

The rest of the thesis is divided into two parts. The first part includes Chapters 3, 4, 5 and 6, and this part focuses on recommender systems. For this part, first in Chapter 3 a comprehensive literature review on recommender systems and algorithms is presented, and the existing methods are reviewed. In addition, the notations and evaluation metrics are discussed.

In Chapter 4, *top-N* recommendation is addressed. A novel Sparse Linear Method (`SLIM`) for *top-N* recommendation is presented with detailed analysis on the sparse models, the learning method and its experimental results. `SLIM` is compared with a comprehensive set of state-of-the-art methods and the results demonstrate that `SLIM` achieves better performance than the state-of-the-art.

In Chapter 5, *top-N* recommendation with side information is addressed. A set of Sparse Linear Methods (`SSLIM`) that incorporates side information into their models is presented, and the corresponding learning methods are provided in Appdendix C. The experimental results on `SSLIM` and the state-of-the-art recommendation algorithms using side information demonstrate that `SSLIM` outperforms the state-of-the-art methods.

In Chapter 6, the rating prediction problem in recommender system is addressed. A multi-task-based rating prediction method is presented and the detailed experimental results on the bench-mark datasets are provided.

The second part focuses on chemical informatics. This part includes Chapter 7, 8 and 9. Similarly to the first part, a comprehensive literature review on the SAR and SSR problems is provided and the notations and metrics used in chemical informatics are presented in Chapter 7.

In Chapter 8, the Structure-Activity-Relationship (SAR) modeling in chemical informatics is addressed. A novel multi-assay-based SAR method is presented. An extensive comparison between the multi-assay-based SAR and the standard SAR methods demonstrates that the multi-assay-based SAR significantly outperforms the standard SAR. In addition, a comparison between the multi-assay-based SAR and the state-of-the-art chemogenomics method shows that the new SAR method performs better than the state-of-the-art chemogenomics method.

In Chapter 9, the Structure-Selectivity-Relationship (SSR) modeling in chemical informatics is addressed. Multiple SSR models based on multi-task learning, multi-class classification, etc, are presented. A comprehensive set of experimental results on real assay data demonstrate that the new SSR methods outperform the existing methods.

In Chapter 10, the conclusions are presented and several future research directions are proposed.

1.3 Related Publications

The work presented in this thesis has been published in leading journals and conferences in data mining, recommender systems and chemical informatics. The related publications are listed as follows:

- **Xia Ning**, Huzefa Rangwala, and George Karypis. Improved SAR models - exploiting the target-ligand relationships. Technical Report 08-011, University of Minnesota, 2008.
- **Xia Ning**, Huzefa Rangwala, and George Karypis. Multi-assay-based structure-activity relationship models: improving structure-activity relationship models by incorporating activity information from related targets. *Journal of Chemical Information and Modelling*, 49(11):2444–2456, Nov 2009.

- **Xia Ning** and George Karypis. The set classification problem and solution methods. In *SIAM International Conference on Data Mining*, pages 847–858. SIAM, 2009.

- **Xia Ning** and George Karypis. Multi-task learning for recommender systems. In *Journal of Machine Learning Research Workshop and Conference Proceedings (ACML2010)*, volume 13, pages 269–284. Microtome Publishing, 2010.

- Nikil Wale, **Xia Ning**, and George Karypis. Trends in chemical graph data mining. In Ahmed K. Elmagarmid, Charu C. Aggarwal, and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 581–606. Springer US, New York, NY, US, 2010.

- **Xia Ning** and George Karypis. `SLIM`: Sparse linear methods for top-n recommender systems. In *Proceedings of 11th IEEE International Conference on Data Mining*, pages 497–506, 2011.

- **Xia Ning**, Michael Walters and George Karypis. Improved machine learning models for predicting selective compounds. *Journal of Chemical Information and Modelling*,52 (1), pp 38-50, Nov 2011.

- **Xia Ning** and George Karypis. Sparse linear models with side-information for top-n recommender systems. WWW2012, accepted as poster, 2012.

- **Xia Ning** and George Karypis. Sparse linear models with side-information for top-n recommender systems. RecSys2012, accepted, 2012.

Chapter 2

Background

2.1 Recommender Systems

Recommender systems represent a set of computational methods that produce recommendations of interesting entities (e.g., products, friends, etc) from a large collection of such entities by retrieving/filtering/learning information from their own properties (e.g., product attributes, personal profiles, etc) and/or the interactions between these entities and other parties (e.g., user-product ratings, friend-friend trust relations, etc). Recommender systems are particularly important for E-commerce applications, where the overwhelming amount of items makes it extremely difficult for users to manually identify those items that best fit their personal preferences. There are two core tasks in recommender systems. The first is to generate a short ranked list of items that best suit a user's personal tastes. This is referred to as *top-N recommendation*. The second is to predict how a user likes an item in terms of a numerical rating. This is referred to as *rating prediction*.

The top-*N* recommendation and rating prediction problems are typically solved through two different classes of approaches. The first is purely content based and only the intrinsic information of the items and/or the users is used to make recommendations. In the content-based approaches, no relations between multiple users or multiple items are used. The second is via collaborative filtering, which generates recommendations for a user by taking into account the purchases/ratings of other users. Collaborative filtering approaches can be further divided into neighborhood based and model based, according to the methods used to derive the recommendations. The neighborhood-based methods explicitly construct user or item neighborhoods

from the data and use the information from the neighborhood to calculate recommendations. The model-based methods learn a model (e.g., regression model, matrix factorization model) from the data, and recommendations are made from the model.

When building recommender systems, there are two major issues to consider. The first one is what data the recommender systems should rely on. There are many different data sources that can be used to build recommender systems. For the items, there are numerous product descriptions, specifications, article content, media attributes that describe item properties from different perspectives. For the users, there are profile data including demographic, social-economic and behavioral information, social/trust networks, etc, which provide implicit or explicit information about user personalities and preferences. Moreover, in terms of the transactions, there is a vast amount of user-item purchase history information and product ratings and reviews that describe whether a user likes an item or not. These data usually have certain characteristics that make the development of effective and efficient recommender systems non-trivial. One of the characteristics is that the data are usually extremely sparse and noisy. As to the user-item purchase/rating history, which is usually organized into a user-item purchase/rating matrix, it is typically of less than 1% density, with more than 99% of the purchase/rating activities not observed or missing. As to the user-item review information, only a small subset of the users tend to review a small subset of the items. In addition, it is almost unavoidable that the reviews are subjective with strong personal opinions, misspellings and grammatical errors.

The second major issue for building recommender systems is what the underlying principles are used by the recommendation methods. The first widely adopted principle is about the "match" between the users and the items. It is assumed that the users derive utilities from the purchases, whereas the items satisfy the users' needs and preferences. When there is a "match" between the users' desires and the items' characteristics, there can be a purchase activity, and thus a corresponding recommendation can be accepted. The second widely used principle is about the "similarity" among the users and items, respectively. That is, the similar users purchase similar items, whereas the similar items are purchased by similar users. In addition, a user tends to purchase similar items. Based on the "similarity" principle, a recommender system that effectively explores the user or item similarities could generate high-quality recommendations.

2.2 Chemical Informatics

Chemical informatics is an interdisciplinary research area in which computational and information technologies are developed and applied to aid the investigation of chemical problems. It is key to the management, analysis and use of chemical data generated by techniques such as combinatorial chemistry and high-throughput screening. Two areas in which chemical informatics plays a critical role are those of chemical genetics and small molecule drug discovery. Chemical genetics seeks to identify small organic molecules (referred to as chemical probes) that can be used to modulate the function of a protein and thus aid in the study and understanding of complex biological systems. Small molecule drug discovery seeks to identify small organic molecules that can modulate the function of pharmaceutically relevant proteins. Within these two areas, chemical informatics provides computational tools to inform the initial and secondary experimental screens and the iterative optimization of the initial compounds (referred to as lead compounds) in order to achieve the desired properties. This thesis focuses on chemical informatics methods for target-based small molecule drug discovery.

There are two important problems that require new computational models and approaches in target-based drug discovery. The first is to predict if a compound is *active* for a specific target (i.e., if it binds to the target with high affinity), and the second is to predict if an active compound is *selective* (i.e., if its binding affinities to any other targets are low). Accurate prediction of a compound's activity and selectivity can significantly reduce the time and cost associated with drug/probe development as it allows the lead optimization process to focus on compounds that have high probability of being potent while at the same time have minimal side effects.

The most successful approaches for building computational models to predict the activity and selectivity of compounds are based on supervised learning methods that take into account the molecular structure of the compounds. The models used to predict the activity of a compound are called Structure-Activity-Relationship (SAR) models, whereas the models used to predict the selectivity of a compound are called Structure-Selectivity-Relationship (SSR) models. The rationale of the SAR/SSR models is that the biological properties (e.g., activity, selectivity) of a chemical compound can be expressed as a function of its molecular structure and physicochemical properties. Since these models are built using supervised learning approaches, they utilize as training data compounds whose activity and selectivity is already known.

2.3 Relations between Recommender Systems & Chemical Informatics

Recommender systems and chemical informatics represent two different application domains and have different problem definitions and data sources. However, there are many commonalities between them, which allow them to be considered under similar methodologies. Table 2.1 summarizes the similarities/analogies between recommender systems and chemical informatics.

Table 2.1: Analogies between Recommender Systems and Chemical Informatics

	Recommender Systems	Chemical Informatics
Data	Users	Targets
	Items	Compounds
	Rating values	Activity values
	Recommendation	Selectivity
	User intrinsic information	Target intrinsic information
	Item intrinsic information	Compound intrinsic information
	User-item rating matrix	Target-compound activity matrix
	User-item transaction matrix	Target-compound selectivity matrix
Problems	Rating prediction	Activity prediction
	Top-N recommendation	Secondary screening library design
	Cold-start recommendation	Library design for novel target
Principles	"Match" between users' needs and items' characteristics	The "lock" and "key" model between protein binding sites and ligands
	"Similar" users purchase "similar"items	"Similar" proteins bind similar compounds

In particular, there is a significant analogy between the data used for recommender systems and chemical informatics. In recommender systems, the user-item rating/purchase information is typically formulated into a user-item matrix, whereas in chemical informatics, the target-compound activity/selectivity information is usually organized into a target-compound matrix. Therefore, the protein targets in chemical informatics are analogous to the users in recommender systems, the compounds are analogous to the items in recommender systems. In addition, predicting the activity value of a compound against a target is analogous to predicting the rating

value on an item given by a user, and predicting whether a compound is selective against a target is analogous to predicting whether to recommend an item to a user. Moreover, the cold start problem in recommender systems, which is to predict ratings or to generate recommendations for new users who are new to the system and do not have any historical information, is analogous to the problem in chemical informatics as to design a library of screening compounds for novel targets that do not have any experimental results. As to the underlying principles used in recommender systems and chemical informatics, the lock-and-key relation between protein targets and their ligands (i.e., the compounds that bind to the protein targets), which is formulated from the fact that the geometric and physicochemical match between the binding pocket of a protein target and its ligand is necessary for high binding affinities, is similar to the "match" relation between user preferences and item properties as discussed in Section 2.1. Due to the analogy between recommender systems and chemical informatics, many algorithms that were originally developed for recommender systems can be applied for SAR models, and SAR methods can also be adopted for recommender systems.

In general, *top-N* recommendation, rating prediction, SAR and SSR problems can all be considered as a form of matrix completion problem [1], that is, given a matrix with partially observed entries, it is to predict the values for the entries which are not observed (e.g., complete the matrix). In the case of *top-N* recommendation, the partially observed matrix is the user-item purchase matrix, in which the observed values are the purchase activities and the *top-N* recommendation algorithms try to complete the purchase matrix by recommending unpurchased items to the users. In the case of SAR problem, the partially observed matrix is the target-compound activity matrix in which the observed values are the activity values of compounds against the targets, and the SAR modeling algorithms try to complete the target-compound activity matrix by predicting the unseen activity values of compounds to the targets.

2.4 Learning Algorithms

In this thesis, various learning algorithms are used to solve the problems in recommender systems and chemical informatics. These learning algorithms include regressions such as regularization linear regression and Support Vector Regression (`SVR`), and classifications using Support Vector Machines (`SVM`) and Neural Networks (`NN`).

2.4.1 Bound-Constraint Least-Squares Problems

Many computational problems can be converted to bound-constraint regularized least-squares problems as in Equation 2.1,

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \frac{1}{2}\|A\mathbf{x}-\mathbf{b}\|_2^2 + \frac{\beta}{2}\|\mathbf{x}\|_2^2 + \boldsymbol{\lambda}^\mathsf{T}\mathbf{x} \\ \text{subject to} \quad & \boldsymbol{\ell} \le \mathbf{x} \le \mathbf{u}, \end{aligned} \tag{2.1}$$

where A is an $m \times n$ data matrix, $\mathbf{x}$ and $\boldsymbol{\lambda}^\mathsf{T}$ are size-n vectors, vector $\boldsymbol{\ell}$ and $\mathbf{u}$ are the lower and upper bounds for $\mathbf{x}$, respectively, and β is a non-negative regularization parameter that controls the ℓ_2 norm of $\mathbf{x}$. Thus, it is essential to have an efficient learning algorithm to solve such optimization problems particularly for large-scale real problems.

The software package BCLS [2, 3] implements an efficient method to solve the above optimization problem. The BCLS algorithm is based on a two-metric projection method [4]. A partitioning of the variables is maintained at all times. Variables that are well within the interior of the feasible set are labeled free, and variables that are at (or near) one of their bounds are labeled fixed. Correspondingly, the variables $\mathbf{x}$ and the data A and $\boldsymbol{\lambda}$ are correspondingly partitioned into their free (B) and fixed (N) components:

$$\mathbf{x} = [\mathbf{x}_B\ \mathbf{x}_N],\ \boldsymbol{\lambda} = [\boldsymbol{\lambda}_B\ \boldsymbol{\lambda}_N],\ \text{and}\ A = [A_B\ A_N]. \tag{2.2}$$

At each iteration, the two-metric projection method generates independent descent directions $\Delta\mathbf{x}_B$ and $\Delta\mathbf{x}_N$ for the free and fixed components of $\mathbf{x}$. They are generated from an approximate solution of the block-diagonal linear system

$$\begin{bmatrix} A_B^\mathsf{T}A_B + \beta^2 I & 0 \\ 0 & D \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_B \\ \Delta\mathbf{x}_N \end{bmatrix} = A^\mathsf{T}\mathbf{r} - \boldsymbol{\lambda} - \beta\mathbf{x}, \tag{2.3}$$

where $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ is the current residual, and D is a diagonal matrix with strictly positive entries. The right-hand side of Equation 2.3 is the negative of the gradient of the objective function in Equation 2.1. Then a Newton step is generated for the free variables $\mathbf{x}_B$, and a scaled steepest-descent step is generated for the fixed variables $\mathbf{x}_N$. The aggregate step $(\Delta\mathbf{x}_B, \Delta\mathbf{x}_N)$ is then projected in the feasible region and the minimizer is computed along the piecewise linear projected-search direction [5].

The linear system in Equation 2.3 is never formed explicitly. Instead, $\Delta \mathbf{x}_B$ is computed equivalently as a solution of the least-squares problem

$$\underset{\Delta \mathbf{x}_B}{\text{minimize}} \quad \frac{1}{2}\|A_B \Delta \mathbf{x}_B - \mathbf{r}\|_2^2 + \boldsymbol{\lambda}_B^{\mathsf{T}} \Delta \mathbf{x}_B + \frac{1}{2}\beta \|\mathbf{x}_B + \Delta \mathbf{x}_B\|_2^2. \tag{2.4}$$

An approximate solution to Equation 2.4 is found by applying the conjugate-gradient type solver LSQR [6] to the problem

$$\underset{\Delta \mathbf{x}_B}{\text{minimize}} \quad \frac{1}{2}\left\| \begin{bmatrix} A_B \\ \beta' I \end{bmatrix} \Delta \mathbf{x}_B - \begin{bmatrix} \mathbf{r} \\ \frac{1}{\beta'}\boldsymbol{\lambda}_B - \frac{\beta}{\beta'}\mathbf{x}_B \end{bmatrix} \right\|_2^2, \tag{2.5}$$

where $\beta' = \max\{\beta, \bar{\beta}\}$ and $\bar{\beta}$ is a small positive constant. If $\beta < \bar{\beta}$, then the resulting step is effectively a modified Newton step.

2.4.2 Support Vector Machines

Support Vector Machine (SVM) [7, 8] is a widely used supervised learning methodology based on large margin principles. In the past decade, have been widely accepted for solving problems in many fields including isolated hand-written digit recognition [7], text categorization [9], and face detection in multimedia [10].

Given training data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_\ell, y_\ell)\} \subset \mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y} = (-1, +1)$, a SVM tries to find a linear classification function

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \tag{2.6}$$

that separates the instances with $y_i = +1$ from the instances with $y_i = -1$ with a large margin, where $\mathbf{w} \in \mathcal{X}$, $b \in \mathbb{R}$, and $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathcal{X}$. This problem can be formulated as a convex optimization problem as follows

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1. \end{aligned} \tag{2.7}$$

In case the training instances are not separable by a hyperplane but classification errors are allowed, the notion of "soft margin" [11] is applied and slack variables ξ_i are introduced so as

to control error tolerance, and thus the problem is formulated as follows

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \end{aligned} \tag{2.8}$$

where the constant $C > 0$ determines the trade-off between the width of the hyperplane margin and the amount to which the errors are tolerated. The formulation in Equation 2.8 corresponds to having a hinge loss function $h(y_i, f(x_i))$

$$h(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i)). \tag{2.9}$$

The Lagrangian of Equation 2.8 is constructed as

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i + \sum_{i=1}^{\ell}\alpha_i[1 - \xi_i - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)] - \sum_{i=1}^{\ell}\mu_i\xi_i, \tag{2.10}$$

where α_i and μ_i are the Lagrange multipliers, and $\alpha_i \in [0, C]$. Requiring the partial derivatives of L with respect to $\mathbf{w}$ and b as 0 gives the conditions

$$\mathbf{w} = \sum_{i=1}^{\ell}\alpha_i y_i \mathbf{x}_i = \sum_{i,\alpha_i>0}\alpha_i y_i \mathbf{x}_i \tag{2.11}$$

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}\rangle + b = \sum_{i,\alpha_i>0}\alpha_i y_i \langle \mathbf{x}_i, x\rangle + b, \tag{2.12}$$

where the $\mathbf{x}_i$'s that have $\alpha_i > 0$ are called support vectors, and therefore, the solution is only the sum of the dot products with a (sparse) subset of training instances.

When the training instances are not linearly separable in $\mathcal{X}$, mapping the instances from $\mathcal{X}$ to another high-dimension (*Hilbert*) space $\mathcal{H}$ by a map function $\Phi : \mathcal{X} \to \mathcal{H}$ may make them linearly separable in $\mathcal{H}$, where the hyperplane becomes

$$f(\mathbf{x}) = \sum_{i,\alpha_i>0}\alpha_i y_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x})\rangle + b, \tag{2.13}$$

which only involves the dot product of $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x})$ in $\mathcal{H}$. Let

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle, \tag{2.14}$$

then Equation 2.13 becomes

$$f(\mathbf{x}) = \sum_{i,\alpha_i>0} \alpha_i y_i \mathcal{K}(\mathbf{x}_i, \mathbf{x}) + b, \tag{2.15}$$

that is, given $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$, it is not required to know explicitly Φ and $\mathcal{H}$, and $\mathbf{w}$ is not explicitly given. The function $\mathcal{K}$ in Equation 2.15 is the kernel function which satisfies the Mercer's condition [12].

2.4.3 Support Vector Regression

Given training data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_\ell, y_\ell)\} \subset \mathcal{X} \times \mathbb{R}$, ε-Support Vector regression (ε-SVR) [13, 14] tries to find a function $f(\mathbf{x})$ that fits the training data with at most ε deviation from their true targets y_i and is as flat as possible. In the case of linear functions

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b, \tag{2.16}$$

where $\mathbf{w} \in \mathcal{X}$, $b \in \mathbb{R}$, and $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathcal{X}$, flatness means that a small $\mathbf{w}$ is desired. One way to achieve this is to minimize the norm of $\mathbf{w}$ ($\|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w} \rangle$), and thus the problem can be formulated as a convex optimization problem as follows

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & |y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle| \leq \varepsilon. \end{aligned} \tag{2.17}$$

The optimization problem in Equation 2.17 may not always be feasible. In addition, errors may be allowed sometimes. Therefore, analogously to the "soft margin" loss function [11], slack variables ξ_i and ξ_i^* are introduced to handle infeasible constraints, and thus the problem is formulated as follows

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \\ \text{subject to} \quad & y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \varepsilon + \xi_i \\ & \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \end{aligned} \tag{2.18}$$

where the constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated. The formulation in Equation 2.18 corresponds

to having an ε-insensitive loss function $|\xi|_\varepsilon$

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{if } \xi \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \tag{2.19}$$

The Lagrangian of Equation 2.18 can be constructed as follows

$$\begin{aligned} L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) - \sum_{i=1}^{\ell}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ - \sum_{i=1}^{\ell}\alpha_i(\varepsilon + \xi_i - y_i + \langle \mathbf{w}, \mathbf{x}_i\rangle + b) \\ - \sum_{i=1}^{\ell}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle \mathbf{w}, \mathbf{x}_i\rangle - b), \end{aligned} \tag{2.20}$$

where $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ are the Lagrange multipliers, and $(\alpha_i, \alpha_i^*, \eta_i, \eta_i^*) \geq 0$. By substituting the partial derivatives of L with respect to $(\mathbf{w}, b, \xi_i, \xi_i^*)$ into Equation 2.20, the dual optimization problem of Equation 2.18 becomes

$$\begin{aligned} \underset{\alpha_i, \alpha_i^*}{\text{maximize}} \quad & \frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle \mathbf{x}_i, \mathbf{x}_j\rangle \\ & - \varepsilon\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \\ \text{subject to} \quad & \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \\ & \alpha_i, \alpha_i^* \in [0, C], \end{aligned} \tag{2.21}$$

and thus $\mathbf{w}$ and f can be described as a linear combination of the training instances $\mathbf{x}_i$ using α_i and α_i^*

$$\mathbf{w} = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\mathbf{x}_i, \tag{2.22}$$

$$f(\mathbf{x}) = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\langle \mathbf{x}_i, \mathbf{x}\rangle + b. \tag{2.23}$$

As it appears in Equation 2.23, the solution of ε-SVR only depends on the dot product of $\mathbf{x}_i$ in the linear case, which allows the notion of kernel taken into the picture. If a map $\Phi : \mathcal{X} \to \mathcal{F}$ maps the training instances $\mathbf{x}_i$ into a feature space $\mathcal{F}$ and let

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle, \tag{2.24}$$

then Equation 2.21 allows an $\varepsilon-$insensitive linear regression in the feature space $\mathcal{F}$

$$\mathbf{w} = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*)\Phi(\mathbf{x}_i), \tag{2.25}$$

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*)\mathcal{K}(\mathbf{x}_i, \mathbf{x}) + b. \tag{2.26}$$

In particular, in Equation 2.26, $f(\mathbf{x})$ does not require explicit knowing of the map Φ and $\mathbf{w}$ is not given explicitly.

2.4.4 Neural Networks

Neural Networks (NN) are biology-inspired computational models which are widely used to approximate a function from examples that are generated from the function. A typical example of NNs is shown in Figure 2.1.

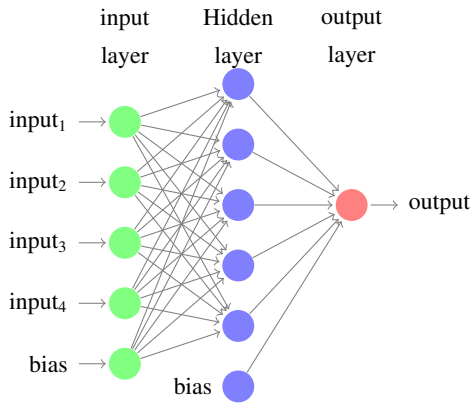


Figure 2.1: A neural network with one hidden layer

The NN in Figure 2.1 has 4 input nodes (i.e., neurons) plus a bias node on its input layer, one hidden layer of 5 hidden nodes and a bias node (with constant input to the bias nodes), and 2 output nodes on its output layer. Nodes between layers are fully connected except with the bias nodes, and no connections between nodes on the same layer. The connections between neurons are weighted. There is a predefined squash function associated with hidden neurons and output neurons. The information from input is constantly fed forward through the neuron connections

in a linear fashion, and then transformed by the squash function at each neuron in a nonlinear fashion until the NN generates an output at the output layer. The NN (weights on connections) is trained so as to minimize a loss function predefined at the output layer.

A popular squash function is sigmoid function, which is defined as follows

$$\sigma(y) = \frac{1}{1 + e^{-sy}} \tag{2.27}$$

where s is a steepness parameter which determines how aggressive the non-linear transform is. The larger the steepness is, the more aggressive the transform is. A good property of sigmoid function is that its derivative can be easily represented by its self, that is

$$\frac{\sigma y}{y} = s\sigma(y)(1 - \sigma(y)) \tag{2.28}$$

y_j is the output at a certain hidden/output neuron j, which is calculated as

$$y_j = \sum_{i=1}^{n} w_{ij}x_{ij} + \theta_k \tag{2.29}$$

where w_{ij} is the weight from neuron$_i$ to neuron$_j$, x_{ij} is the input from neuron i to neuron j (on different layers), and the θ_k is the bias on the layer as of neuron i.

At the output layer, a loss function is always defined so as to serve as the object as the neural network is trained. Sum of Mean Square Errors (MSE) is a popular loss function defined as

$$L(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{dk} - o_{dk})^2 \tag{2.30}$$

where D is the set of training data, t_{dk} is the target label of training instance d at output neuron k, o_{dk} is the output at output neuron k from the neural network for instance d, and $\vec{w}$ is the weights on the net.

Backpropagation algorithm is a widely used approach to train a neural network. The key idea of backpropagation is to feed backward the errors made at the output layer so as to update the network weights. Stochastic gradient decent method is always applied to do backpropagation.

Part I

Recommender Systems

Chapter 3

Preliminaries for Recommender Systems

3.1 Literature Reviews

3.1.1 *Top-N* Recommender Systems

Top-N recommender systems are used in E-commerce applications to recommend size-N ranked lists of items that users may like the most, and they have been intensively studied during the last few years. The methods for *top-N* recommendation can be broadly classified into two categories. The first category is the neighborhood-based Collaborative Filtering (`CF`) methods [15]. For a certain user, *user-based k-nearest-neighbor* (`userkNN`) collaborative filtering methods first identify a set of similar users, and then recommend *top-N* items based on what items those similar users have purchased. Similarly, *item-based k-nearest-neighbor* (`itemkNN`) collaborative filtering methods first identify a set of similar items for each of the items that the user has purchased, and then recommend *top-N* items based on those similar items. The user/item similarity is calculated from user-item purchase/rating matrix in a collaborative filtering fashion with some similarity measures (e.g., Pearson correlation, cosine similarity) applied. One advantage of the item-based methods is that they are efficient to generate recommendations due to the fact that the item neighborhood is sparse. However, they suffer from low accuracy since there is essentially no knowledge learned about item characteristics so as to produce accurate *top-N* recommendations.

The second category is model-based methods, particularly latent factor models as they have achieved the state-of-the-art performance on large-scale recommendation tasks. The key idea of latent factor models is to factorize the user-item matrix into (low-rank) user factors and item factors that represent user tastes and item characteristics in a common latent space, respectively. The prediction for a user on an item can be calculated as the dot product of the corresponding user factor and item factor. There are various Matrix Factorization (`MF`)-based methods proposed in recent years for building such latent factor models. Cremonesi *et al* [16] proposed a simple Pure Singular-Value-Decomposition-based (`PureSVD`) matrix factorization method, which describes users and items by the most principle singular vectors of the user-item matrix. Pan *et al* [17] and U *et al* [18] proposed a Weighted Regularized Matrix Factorization (`WRMF`) method formulated as a regularized Least-Squares (`LS`) problem, in which a weighting matrix is used to differentiate the contributions from observed purchase/rating activities and unobserved ones. Rennie [19] and Srebro [20] proposed a Max-Margin Matrix Factorization (`MMMF`) method, which requires a low-norm factorization of the user-item matrix and allows unbounded dimensionality for the latent space. This is implemented by minimizing the trace-norm of the reconstructed user-item matrix from the factors. Sindhwani *et al* [21] proposed a Weighted Non-Negative Matrix Factorization (`WNNMF`) method, in which they enforce non-negativity on the user and item factors so as to lend "part-based" interpretability to the model. Hofmann [22] applied Probabilistic Latent Semantic Analysis (`PLSA`) technique for collaborative filtering, which has been shown equivalent to non-negative matrix factorization. `PLSA` introduces a latent space such that the co-occurrence of users and items (i.e., a certain user has purchased a certain item) can be rendered conditionally independent. Koren [23] proposed an intersecting approach between neighborhood-based method and `MF`. In his approach, item similarity is learned simultaneously with a matrix factorization so as to take advantages of both methods.

Top-N recommendation has also been formulated as a ranking problem. Rendle *et al* [24] proposed a Bayesian Personalized Ranking (`BPR`) criterion, which is the maximum posterior estimator from a Bayesian analysis and measures the difference between the rankings of user-purchased items and the rest items. `BPR` can be well adopted for item *k*nn method (`BPRkNN`) and MF methods (`BPRMF`) as a general objective function.

3.1.2 *Top-N* Recommender Systems with Side Information

Various methods have been developed to incorporate side information in recommender systems. Most of these methods have been developed in the context of the rating prediction problem, whereas the *top-N* recommendation problem has received less attention. The follow review contains some of the best performing schemes for both the rating prediction and *top-N* recommendation problems.

The first category of these methods is based on latent factor models. In [25], Singh *et al* proposed the collective matrix factorization method for both rating prediction and *top-N* recommendation, which collectively factorizes user-item purchase profile matrix and item-feature content matrix into a common latent space such that the two types of information are leveraged via common the item factors. Agarwal *et al* [26] proposed regression-based latent factor models for rating prediction, which use features for factor estimation. In their method, the user and item latent factors are estimated through independent regression on user and item features, and the recommendation is calculated from a multiplicative function on user and item factors. Yang *et al* [27] developed a joint friendship and interest propagation model for *top-N* recommendation, in which the user-item interest network and the user-user friendship network (side information on users) are jointly modeled through latent user and item factors. User factors are shared by the interest network approximation component and the friendship network approximation component so as to enable information propagation. They demonstrated the their model is a generalization of Singh *et al* [25], Koren [23] and Agarwal *et al* [26].

Methods using tensor factorization (`TF`) have also gained popularity. Karatzoglou *et al* [28] considered the user-item-feature relation as a tensor, and they proposed to use regularized `TF` to model the relations between the three sets of entities for rating prediction. `TF` can be considered as a generalization of `MF`, in which the relations among all the modes (i.e., users, items and features) are jointly learned. Rendle *et al* [29] also treated user-item-feature as a tensor, and they factorized all pairwise interactions in the tensor (i.e, items vs users, items vs context features, users vs context features) rather than the entire tensor for rating prediction.

Another category of algorithms that utilize side information is based on networks. Gunawardana *et al* [30] proposed unified Boltzmann machines for *top-N* prediction, in which user-item profile information and side information are treated as homogeneous features, and interaction weights between such features and user actions are learned in a coherent manner so as to reflect how well such features can predict user actions. Campos *et al* [31] combined content-based and

collaborative-filtering based recommendations with Bayesian networks, which are composed of item nodes, user nodes and item feature nodes. During predictions, content information is propagated from purchased items to non-purchased items via feature nodes, and purchase information is propagated from other users to the user of concern via item nodes. Then the two parts are combined to make recommendations.

3.1.3 Rating Predictions and Multi-Task Learning

Some rating prediction methods have been briefly described in the previous section (i.e., Section 3.1.2). In addition to them, collaborative filtering (CF) has been one of the most successful approaches for rating prediction. A review on early works of traditional collaborative filtering is available in ([32]), whereas a recent review on up-to-date collaborative filtering methods can be found in ([33]). In recent years, Matrix Factorization (MF) has gained great popularity and it has achieved the state-of-the-art performance particularly on large-scale rating prediction problems. A review on MF for recommender systems is available in ([34]).

Multi-task learning [35, 36, 37, 38, 36] is a transfer learning mechanism designed to improve the generalization performance of a given model by leveraging the domain-specific information contained in the training signals of related tasks. In multi-task learning, multiple related tasks are represented by a common representation, and then they are learned in parallel, such that information from one task can be transferred to another task through their common representations or shared learning steps so as to boost that task's learning performance. A very intuitive multi-task model utilizes back-propagation Neural Networks (NN) [39]. Input to the back-propagation net is the common representations of all related tasks. For each task to be learned through the net, there is one output from the net. A hidden layer is shared across all the tasks such that by back-propagation all the tasks can learn task-related/target-specific signals from other tasks through the shared hidden layer. Within such a net, all the tasks can be learned simultaneously, and by leveraging knowledge from other related tasks, each task can be better learned than only from its own training instances. In recent years, many sophisticated multi-task learning methods have emerged, which include kernel methods [38], Gaussian processes [40], task clustering [35], Bayesian models [41], matrix regularization [42], *etc*. Various studies have reported promising results with the use of multi-task learning in diverse areas such as Cheminformatics [43, 44], face recognition [45], and text mining [46].

A number of different CF-based rating prediction methods have been developed that directly

or indirectly combine the prediction models of different users and as such can be considered as instances of multi-task learning. An example is Yu *et al*. ([47]), in which they assume that the ratings from each user are generated from a latent function that is derived from a common Gaussian process prior. They formulate the rating prediction problem for each user as an ordinal regression problem. They simultaneously learn the regression models for all the users iteratively by first obtaining a multivariate Gaussian on each latent function through expectation propagation, and then collaboratively updating the Gaussian process prior based on all latent functions. Similarly, in Yu ([48]), they formulate the collaborative filtering as a maximum-margin matrix approximation problem with regularization. They prove that under certain circumstances, such a formulation is equivalent to a kernel learning problem similar to ([47]), in which each user is modeled by a prediction function, and the common structure of all prediction functions is modeled by a kernel. By doing matrix factorization on the entire user-item rating matrix, all such prediction functions are essentially learned simultaneously. The close relationship between maximum-margin matrix factorization for collaborative filtering and multi-task learning has also been observed and studied in ([49]).

Another similar approach was developed by Abernethy *et al*. ([50]), in which they formulate the rating prediction problem as a linear map from user Hilbert space to item Hilbert space by an operator, and they solve the problem by looking for the best operator that can achieve such a map with minimum loss and penalty. They argue that if only users (or items) have attributes available, and a certain rank penalty is applied so as to enforce information sharing across prediction operators, the operator estimation formulation results in a multi-task learning method based on low-rank representation of the prediction functions.

A different methodology was developed by Phuong *et al*. ([51]), in which they formulate the rating prediction problem as binary classification (i.e., like or dislike). They build one classifier for each user, and all the classifiers are learned and boosted together. The classification algorithm fits a decision stump for each user on his/her weighted ratings, whereas the boosting algorithm selects a stump for each user that gives the lowest overall error for a subset of classifiers (corresponding to a certain class), and then updates weights on items correspondingly until best performance is achieved.

3.2 Definitions and Notations

The definitions and notations used recommender systems are described in Table 3.1. Note that in this thesis, all vectors (e.g., $\mathbf{m}_i^\mathsf{T}$ and $\mathbf{m}_j$) are represented by bold lower-case letters and all matrices (e.g., M) are represented by upper-case letters. Row vectors are represented by having the transpose supscript$^\mathsf{T}$, otherwise by default they are column vectors. A predicted/approximated value is denoted by having a $\sim$ head. The corresponding matrix/vector notations are used instead of user/item purchase/rating profiles if no ambiguity is raised.

Table 3.1: Definitions and Notations for Recommender Systems

Not	Definitions	Description
u (u_i)	a user (the i-th user)	
u_*	an active user	the user for whom a prediction needs to be computed
t (t_j)	an item (the j-th item)	
t_*	an active item	the item whose rating needs to be predicted
$\mathcal{U}$	the set of all the m users in the system	$\|\mathcal{U}\| = m$
$\mathcal{T}$	the set of all the n items in the system	$\|\mathcal{T}\| = n$
$M_{m\times n}$	user-item purchase matrix of size $m \times n$	binary matrix
$\mathbf{m}_i^\top$	the i-th row of M	the purchase history of u_i on $\mathcal{T}$
$\mathbf{m}_j$	the j-th column of M	the purchase profile of t_j from $\mathcal{U}$
m_{ij}	the (i, j) entry of M	$m_{ij} = 1/0$ if u_i has/has not purchased t_j
$R_{m\times n}$	user-item rating matrix of size $m \times n$	multivariate matrix
$r_{i,j}$	the (i, j) entry of R	$r_{i,j} = rating/0$ if u_i has rated t_j as *rating*/has not rated t_j
$\mathcal{U}_j$	the set of non-zero rows in the j-th column of R	the set of users that have rated t_j
$\mathcal{T}_i$	the set of non-zero columns in the i-th row of R	the set of items rated by u_i
$\bar{r}_{i,\cdot}$	the average of non-zero values in the i-th row of R	the average rating of u_i on $\mathcal{T}_i$
$\mathcal{N}_*$	a set of related users for u_*	described in Section 6.2
$\mathcal{N}$	a set of users	
$\mathcal{T}_\mathcal{N}$	the set of items rated by users $\mathcal{N}$	described in Section 6.2
$F_{\ell\times n}$	side information-item matrix of size $\ell \times n$	binary or multivariate matrix
$\mathbf{f}_j$	the j-th column of F	the side information vector of t_j
N	the number of items to be recommended	by default $N = 10$

3.3 Evaluation Methodology & Metrics

Table 3.2: Evaluation for Recommender Systems

Name	Definitions	Description
Hit Rate (HR)	$\text{HR} = \frac{\#\text{hits}}{\#\text{users}}, \quad (3.1)$	#users: the total number of users (i.e., m in Table 3.1), #hits: the number of users whose item in the testing set is correctly recommended (i.e., hit)
Average Reciprocal Hit-Rank (ARHR)	$\text{ARHR} = \frac{1}{\#\text{users}} \sum_{i=1}^{\#\text{hits}} \frac{1}{p_i}, \quad (3.2)$	p: the position of the hit item in the ranked recommendation list
per-rating Hit Rate (rHR)	$\text{rHR} = \frac{\#\text{hits}(r)}{\#\text{users}(r)}, \quad (3.3)$	The HR on items that have a certain rating value r.
cumulative Hit Rate (cHR)	$\text{cHR} = \frac{\#\text{hits}(\leq r)}{\#\text{users}(\leq r)}, \quad (3.4)$	The HR on items that have a rating value which is no lower than a certain rating threshold r.
Mean Absolute Error (MAE)	$MAE = \frac{\sum_{i,j} \lvert p_{i,j} - r_{i,j} \rvert}{\sum_{i,j} 1} \quad (3.5)$	$\sum_{i,j} 1$: the total number of predictions from $\mathcal{U}$ on $\mathcal{T}$, $p_{i,j}$: the predicted rating on t_j by i (if any), $r_{i,j}$: the corresponding true rating (Table 3.1)

For *top-N* recommendations, 5-time Leave-One-Out cross validation (LOOCV) is applied to evaluate the performance of different recommender systems. In each run, each of the datasets is split into a training set and a testing set by randomly selecting one of the non-zero entries of each user and placing it into the testing set. The training set is used to train a model, then for each user a size-N (by default $N = 10$) ranked list of recommended items is generated by the model. The evaluation is conducted by comparing the recommendation list of each user and the item of that user in the testing set.

The recommendation quality is measured by the Hit Rate (HR) and the Average Reciprocal

Hit-Rank (ARHR) [15]. HR and ARHR are defined as in Equation 3.1 and Equation 3.2 in Table 3.2, respectively. ARHR is a weighted version of HR and it measures how strongly an item is recommended, in which the weight is the reciprocal of the hit position in the recommendation list.

For the *top-N* recommendation experiments utilizing ratings, the performance of the methods is evaluated by looking at how well they can recommend the items that have a particular rating value. For this purpose, we define the per-rating Hit Rate (rHR) and cumulative Hit Rate (cHR). The metrics rHR and cHR are defined as in Equation 3.3 and Equation 3.4 in Table 3.2, respectively.

Note that in the *top-N* recommendation literature, there exist other metrics for evaluation. Such metrics include area under the ROC curve (AUC), which measures relative positions of true positives and false positives in an entire ranked list. Variances of AUC can measure the positions in top part of a ranked list. Another popular metric is recall. However, in *top-N* recommendation scenario, we believe HR and ARHR are the most direct and meaningful measures, since the users only care if a short recommendation list has the items of interest or not rather than a very long recommendation list. Due to this we use HR and ARHR in our evaluation.

For the rating prediction problem, the performance of the methods is evaluated using a five-fold cross validation framework. For each user, all of his/her ratings are randomly split into five equal-sized folds. For the active user u_*, 4 out of his/her 5 folds are used in training, and his/her remaining fold is used for evaluation. For other users, all of their 5 folds are used to compute user similarity with u_*'s 4 training folds and to select items from. This is done in order to maximize the opportunity for the active user to identify the best set of related users and items. Mean Absolute Error (MAE) is used as the statistical accuracy metric. MAE is a widely used evaluation metric in recommender system community, which is defined as in Equation 3.5 as in Table 3.2. MAE measures the average absolute deviation of recommendations from true user-specified ratings. Lower MAE indicates more accurate recommendations

Chapter 4

SLIM: Sparse Linear Methods for Top-N Recommender Systems

This chapter focuses on developing effective and efficient algorithms for *top-N* recommender systems. A novel Sparse LInear Method (SLIM) is proposed, which generates *top-N* recommendations by aggregating from user purchase/rating profiles. A sparse aggregation coefficient matrix S is learned from SLIM by solving an ℓ_1-norm and ℓ_2-norm regularized optimization problem. S is demonstrated to produce high-quality recommendations and its sparsity allows SLIM to generate recommendations very fast. A comprehensive set of experiments is conducted by comparing the SLIM method and other state-of-the-art *top-N* recommendation methods. The experiments show that SLIM achieves significant improvements both in run time performance and recommendation quality over the best existing methods.

4.1 Introduction

In recent years, various algorithms for *top-N* recommendation have been developed [52]. These algorithms can be categorized into two classes: neighborhood-based collaborative filtering methods and model-based methods. Among the neighborhood-based methods, those based on item neighborhoods can generate recommendations very fast, but they achieve this with a sacrifice on recommendation quality. On the other hand, model-based methods, particularly those based on latent factor models incur a higher cost while generating recommendations, but the quality of these recommendations is higher, and they have been shown to achieve the best

performance especially on large recommendation tasks.

In this chapter, we propose a novel Sparse LInear Method (SLIM) for *top-N* recommendation that is able to make *high-quality* recommendations *fast*. SLIM learns a sparse coefficient matrix for the items in the system solely from the user purchase/rating profiles by solving a regularized optimization problem. Sparsity is introduced into the coefficient matrix which allows it to generate recommendations efficiently. Feature selection methods allow SLIM to substantially reduce the amount of time required to learn the coefficient matrix. Furthermore, SLIM can be used to do *top-N* recommendations from ratings, which is a less exploited direction in recommender system research.

The SLIM method addresses the demands for high quality and efficiency in *top-N* recommender systems concurrently, so it is better suitable for real-time applications. We conduct a comprehensive set of experiments on various datasets from different real applications. The results show that SLIM produces better recommendations than the state-of-the-art methods at a very high speed. In addition, it achieves good performance in using ratings to do *top-N* recommendation.

4.2 Sparse Linear Methods for *Top-N* Recommendation

4.2.1 SLIM for *Top-N* Recommendation

In this chapter, we propose a Sparse LInear Method (SLIM) to do *top-N* recommendation. In the SLIM method, the recommendation score on an un-purchased/-rated item t_j of a user u_i is calculated as a *sparse* aggregation of items that have been purchased/rated by u_i, that is,

$$\tilde{m}_{ij} = \mathbf{m}_i^\top \mathbf{s}_j, \tag{4.1}$$

where $m_{ij} = 0$ and $\mathbf{s}_j$ is a sparse size-n column vector of aggregation coefficients. Thus, the model utilized by SLIM can be presented as

$$\tilde{M} = MS, \tag{4.2}$$

where M is the binary user-item purchase matrix or the user-item rating matrix, S is an $n \times n$ sparse matrix of aggregation coefficients, whose j-th column corresponds to $\mathbf{s}_j$ as in Equation 4.1, and each row $\tilde{\mathbf{m}}_i^\top$ of $\tilde{M}$ ($\tilde{\mathbf{m}}_i^\top = \mathbf{m}_i^\top S$) represents the recommendation scores on all items for user u_i. *Top-N* recommendation for u_i is done by sorting u_i's non-purchased/-rated items based on their recommendation scores in $\tilde{\mathbf{m}}_i^\top$ in decreasing order and recommending the *top-N* items.

4.2.2 Learning S for SLIM

We view the purchase/rating activity of user u_i on item t_j in M (i.e., m_{ij}) as the ground-truth item recommendation score. Given a user-item purchase/rating matrix M of size $m \times n$, we learn the sparse $n \times n$ matrix S in Equation 4.2 as the minimizer for the following regularized optimization problem:

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \frac{1}{2}\|M - MS\|_F^2 + \frac{\beta}{2}\|S\|_F^2 + \lambda\|S\|_1 \\ \text{subject to} \quad & S \geq 0 \\ & \text{diag}(S) = 0, \end{aligned} \tag{4.3}$$

where $\|S\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{n}|s_{ij}|$ is the entry-wise ℓ_1-norm of S, and $\|\cdot\|_F$ is the matrix Frobenius norm. In Equation 4.3, MS is the estimated matrix of recommendation scores (i.e., $\tilde{M}$) by the sparse linear model as in Equation 4.2. The first term $\frac{1}{2}\|M - MS\|_F^2$ (i.e., the residual sum of squares) measures how well the linear model fits the training data, and $\|S\|_F^2$ and $\|S\|_1^2$ are ℓ_F-norm and ℓ_1-norm regularization terms, respectively. The constants β and λ are regularization parameters. The larger the parameters are, the more severe the regularizations are. The non-negativity constraint is applied on S such that the learned S represents positive relations between items, if any. The constraint $\text{diag}(S) = 0$ is also applied so as to avoid trivial solutions (i.e., the optimal S is an identical matrix such that an item always recommends itself so as to minimize $\frac{1}{2}\|M - MS\|_F^2$). In addition, the constraint $\text{diag}(S) = 0$ ensures that m_{ij} is not used to compute $\tilde{m}_{ij}$.

ℓ_1-norm and ℓ_F-norm Regularizations for SLIM

In order to learn a sparse S, we introduce the ℓ_1-norm of S as a regularizer in Equation 4.3. It is well known that ℓ_1-norm regularization introduces sparsity into the solutions [53].

Besides the ℓ_1-norm, we have the ℓ_F-norm of S as another regularizer, which leads the optimization problem to an elastic net problem [54]. The ℓ_F-norm measures model complexity and prevents overfitting (as in ridge regression). Moreover, ℓ_1-norm and ℓ_F-norm regularization together implicitly group correlated items in the solutions [54].

Computing S

Since the columns of S are independent, the optimization problem in Equation 4.3 can be decoupled into a set of optimization problems of the form

$$\begin{aligned} \underset{\mathbf{s}_j}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{m}_j - M\mathbf{s}_j\|_2^2 + \frac{\beta}{2}\|\mathbf{s}_j\|_2^2 + \lambda\|\mathbf{s}_j\|_1 \\ \text{subject to} \quad & \mathbf{s}_j \geq \mathbf{0} \\ & s_{j,j} = 0, \end{aligned} \tag{4.4}$$

which allows each column of S to be solved independently. In Equation 4.4, $\mathbf{s}_j$ is the j-th column of S and $\mathbf{m}_j$ is the j-th column of M, $\|\cdot\|_2$ is ℓ_2-norm of vectors, and $\|\mathbf{s}_j\|_1 = \sum_{i=1}^{n} |s_{ij}|$ is the entry-wise ℓ_1-norm of vector $\mathbf{s}_j$. Due to the column-wise independence property of S, learning S can be easily parallelized. The optimization problem of Equation 4.4 can be converted to the BCLS problem as described in Section 2.4.1, given that with the non-negativity constraint $\mathbf{s}_j \geq \mathbf{0}$, $\lambda\|\mathbf{s}_j\|_1 = \lambda\mathbf{1}^\mathsf{T}\mathbf{s}_j$.

`SLIM` with Feature Selection

The estimation of $\mathbf{s}_j$ in Equation 4.4 can be considered as the solution to a regularized regression problem in which the j-th column of M is the dependent variable to be estimated as a function of the remaining $n - 1$ columns of M (independent variables). This view suggests that feature selection methods can potentially be used to reduce the number of independent variables prior to computing $\mathbf{s}_j$. The advantage of such feature selection methods is that they reduce the number of columns in M, which can substantially decrease the overall amount of time required for `SLIM` learning.

Motivated by these observations, we extended the `SLIM` method to incorporate feature selection. We will refer to these methods as `fsSLIM`. Even though many feature selection approaches can be used, in this chapter we only investigated one approach, inspired by the `itemkNN` *top-N* recommendation algorithms. Specifically, since the goal is to learn a linear model to estimate the j-th column of M (i.e., $\mathbf{m}_j$), then the columns of A that are the most similar to $\mathbf{m}_j$ can be used as the selected features. As our experiments will later show, using the cosine similarity and this feature selection approach, leads to a method that has considerably lower computational requirements with minimal quality degradation.

4.2.3 Efficient *Top-N* Recommendation from SLIM

The SLIM method in Equation 4.2 and the sparsity of S enable significantly faster *top-N* recommendation. In Equation 4.2, $\mathbf{m}_i^\intercal$ is always very sparse (i.e., the user usually purchased/rated only a small fraction of all items), and when S is also sparse, the calculation of $\tilde{\mathbf{m}}_i^\intercal$ can be very fast by exploiting S's sparse structure (i.e., applying a "gather" operation along S columns on its non-zero values in the rows corresponding to non-zero values in $\mathbf{m}_i^\intercal$). Thus, the computational complexity to do recommendations for user u_i is $O(n_{m_i} \times n_s + N \log(N))$, where n_{m_i} is the number of non-zero values in $\mathbf{m}_i^\intercal$, and n_s is the average number of non-zero values in the rows of S. The $N \log(N)$ term is for sorting the highest scoring N items, which can be selected from the potentially $n_{ai} \times n_w$ non-zeros entries in $\tilde{\mathbf{m}}_i^\intercal$ in linear time using linear selection.

4.2.4 SLIM vs Existing Linear Methods

Linear methods have already been used for *top-N* recommendation. For example, the itemkNN method in [15] has a linear model similar to that of SLIM. The model of itemkNN is a $k$nn item-item cosine similarity matrix S, that is, each row $\mathbf{s}_i^\intercal$ has exactly k nonzero values representing the cosine similarities between item t_j and its k most similar neighbors. The fundamental difference between itemkNN and SLIM's linear models is that the former is highly dependent on the pre-specified item-item similarity measure used to identify the neighbors, whereas the later generates S by solving the optimization problem of Equation 4.3. In this way, S can potentially encode rich and subtle relations across items that may not be easily captured by conventional item-item similarity metrics. This is validated by the experimental results in Section 4.4 that show the S substantially outperforms S.

Rendle *et al* [24] discussed an adaptive k-Nearest-Neighbor method, which used the same model as in itemkNN in [15] but adaptively learn the item-item similarity matrix. However, the item-item similarity matrix in [24] is fully dense, symmetric and has negative values. S is different from Rendle *et al*'s item-item similarity matrix in that, in addition to its sparsity which leads to fast recommendation and low requirement for storage, S is not necessarily symmetric due to the optimization process and thus allows more flexibility for recommendation.

Paterek [55] introduced a linear model for each item for rating prediction, in which the rating of a user u_i on an item t_j is calculated as the aggregation of the ratings of u_i on all other items. They learned the aggregation coefficients (equivalent to S) by solving an ℓ_2-norm

regularized least squares problem for each item. The learned coefficients are fully dense. The advantage of SLIM over Paterek's method is that ℓ_1-norm regularization is incorporated during learning which enforces S to be sparse, and thus, the most informative signals are captured in S while noises are discarded. In addition, SLIM learns S from all purchase/rating activities so as to better fuse information, compared to Paterek's method, which only uses a certain set of purchase/rating activities.

4.2.5 Relations between SLIM and MF Methods

MF methods for *top-N* recommendation have a model

$$\tilde{M} = UV^{\mathsf{T}}, \tag{4.5}$$

where U and V^{T} are the user and item factors, respectively. Comparing MF model in Equation 4.5 and the SLIM method in Equation 4.2, we can see that SLIM's model can be considered as a special case of MF model (i.e., M is equivalent to U and S is equivalent to V^{T})

U and V^{T} in Equation 4.5 are in a latent space, whose dimensionality is usually specified as a parameter. The "latent" space becomes exactly the item space in Equation 4.2, and therefore, in SLIM there is no need to learn user representation in the "latent" space and thus the learning process is simplified. On the other hand, U and V^{T} are typically of low dimensionality, and thus useful information may potentially get lost during the low-rank approximation of M from U and V^{T}. On the contrary, in SLIM, since information on users are fully preserved in M and the counterpart on items is optimized via the learning, SLIM can potentially give better recommendations than MF methods.

In addition, since both U and V^{T} in Equation 4.5 are typically dense, the computation of $\mathbf{m}_i^{\mathsf{T}}$ requires the calculation of each $\tilde{m}_{ij}$ from its corresponding dense vectors in U and V^{T}. This results in a high computational complexity for MF methods to do recommendations, that is, $O(k^2 \times n)$ for each user, where k is the number of latent factors, and n is the number of items. The computational complexity can be potentially reduced by utilizing the sparse matrix factorization algorithms developed in [56, 57, 58]. However, none of these sparse matrix factorization algorithms have been applied to solve *top-N* recommendation problem due to their high computational costs.

4.3 Materials

We evaluated the performance of SLIM methods on eight different real datasets whose characteristics are shown in Table 4.1.

Table 4.1: The Datasets Used in Evaluation

dataset	#users	#items	#trns	rsize	csize	density	ratings
ccard	42,067	18,004	308,420	7.33	17.13	0.04%	-
ctlg2	22,505	17,096	1,814,072	80.61	106.11	0.47%	-
ctlg3	58,565	37,841	453,219	7.74	11.98	0.02%	-
ecmrc	6,594	3,972	50,372	7.64	12.68	0.19%	-
BX	3,586	7,602	84,981	23.70	11.18	0.31%	1-10
ML10M	69,878	10,677	10,000,054	143.11	936.60	1.34%	1-10
Netflix	39,884	8,478	1,256,115	31.49	148.16	0.37%	1-5
Yahoo	85,325	55,371	3,973,104	46.56	71.75	0.08%	1-5

Columns corresponding to #users, #items and #trns show the number of users, number of items and number of transactions, respectively, in each dataset. Columns corresponding to rsize and csize show the average number of transactions for each user and on each item (i.e., row density and column density of the user-item matrix), respectively, in each dataset. Column corresponding to density shows the density of each dataset (i.e., density = #trns/(#users $\times$ #items)). Column corresponding to ratings shows the rating range of each dataset with granularity 1.

These datasets can be broadly classified into two categories.

The first category (containing ccard, ctlg2, ctlg3 and ecmrc [15]) was derived from customer purchasing transactions. Specifically, the ccard dataset corresponds to credit card purchasing transactions of a major department store, n which each card has at least 5 transactions. The ctlg2 and ctlg3 datasets correspond to the catalog purchasing transactions of two major mail-order catalog retailers. The ecmrc dataset corresponds to web-based purchasing transactions of an e-commerce site. These four datasets have only binary purchase information.

The second category (containing BX, ML10M, Netflix and Yahoo) contains multi-value ratings. All the ratings are converted to binary indications if needed. In particular, the BX dataset is a subset from the Book-Crossing dataset[1] , in which each user has rated at least

[1] http://www.informatik.uni-freiburg.de/~cziegler/BX/

20 items and each item has been rated by at least 5 users and at most 300 users. The ML10M dataset corresponds to movie ratings and was obtained from the MovieLens[2] research projects. The Netflix dataset is a subset extracted from the Netflix Prize dataset[3] such that each user has rated 20 – 250 movies, and each movie is rated by 20 – 50 users. Finally, the Yahoo dataset is a subset extracted from Yahoo! Music user ratings of songs, provided as part of the Yahoo! Research Alliance Webscope program[4] . In Yahoo dataset, each user has rated 20 – 200 songs, and each music has been rated by at least 10 users and at most 5000 users.

4.4 Experimental Results

In this section, we present the performance of SLIM methods and compare them with other popular *top-N* recommendation methods. We present the results from two sets of experiments. In the first set of experiments, all the *top-N* recommendation methods use binary user-item purchase information during learning, and thus all the methods are appended by -*b* to indicate binary data used (e.g., SLIM-*b*) if there is confusion. In the second set of experiments, all the *top-N* recommendation methods use user-item rating information during learning, and correspondingly they are appended by -r if there is confusion.

We optimized all the C implementations of the algorithms to make sure that any time difference in performance is due to the algorithms themselves, and not due to the implementation. For all the methods, we conducted an exhaustive grid search to identify the best parameters to use. We only report the performance corresponding to the parameters that lead to the best results in this section.

4.4.1 SLIM Performance on Binary data

Comparison Algorithms

We compare the performance of SLIM with another three categories of *top-N* recommendation algorithms. The first category of algorithms is the item/user neighborhood-based collaborative filtering methods itemkNN, itemprob and userkNN. Methods itemkNN and userkNN are as discussed in Section 3.1.1 and Section 4.2.4. Method itemprob is similarity to itemkNN except

[2] http://www.grouplens.org/node/12
[3] http://www.netflixprize.com/
[4] http://research.yahoo.com/Academic_Relations

that instead of item-item cosine similarity, it uses modified item-item transition probabilities. These methods are engineered with various heuristics for better performance[5] .

The second category of algorithms is the `MF` methods, including `PureSVD` and `WRMF` as discussed in Section 3.1.1. Note that both `PureSVD` and `WRMF` use 0 values in the user-item matrix during learning. `PureSVD` is demonstrated to outperform other `MF` methods in *top-N* recommendation using ratings [16], including the `MF` methods which treat 0s as missing data. `WRMF` represents the state-of-the-art matrix factorization methods for *top-N* recommendation using binary information.

The third category of algorithms is the methods which rely on ranking/retrieval criteria, including `BPRMF` and `BPRkNN` as discussed in Section 3.1.1 and Section 4.2.4. It is demonstrated in [24] that `BPRMF` outperforms other methods in *top-N* recommendation in terms of AUC measure using binary information.

Top-N Recommendation Performance

Table 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9 show the overall performance of different *top-N* recommendation algorithms.

Descriptions of the tables In Tables 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9, columns corresponding to params present the parameters for the corresponding method. For methods `itemkNN` and `userkNN`, the parameters are number of neighbors, respectively. For method `itemprob`, the parameters are the number of neighbors and transition parameter α. For method `PureSVD`, the parameters are the number of singular values and the number of iterations during SVD. For method `WRMF`, the parameters are the dimension of the latent space and the weight on purchases. For method `BPRMF`, the parameters are the dimension of the latent space and learning rate, respectively. For method `BPRkNN`, the parameters are the learning rate and regularization parameter λ. For method `SLIM`, the parameters are ℓ_2-norm regularization parameter β and ℓ_1-norm regularization parameter λ. For method `fsSLIM`, the parameters are the number of neighbors and ℓ_1-norm regularization parameter λ. Columns corresponding to HR and ARHR present the hit rate and average reciprocal hit-rank, respectively. Columns corresponding to mt and tt present the time used by model learning and recommendation, respectively. The mt/tt

[5] http://glaros.dtc.umn.edu/gkhome/suggest/overview

numbers with (s), (m) and (h) are time used in seconds, minutes and hours, respectively. **Bold** numbers are the best performance in terms of HR for each dataset.

Table 4.2: Comparison of *Top-N* Recommendation Algorithms on ccard

method	params		HR	ARHR	mt	tt
itemkNN	50	-	0.195	0.145	0.54(s)	1.34(s)
itemprob	50	0.2	0.226	0.154	0.97(s)	1.24(s)
userkNN	150	-	0.189	0.122	0.06(s)	14.84(s)
PureSVD	3500	10	0.101	0.058	42.89(m)	2.65(h)
WRMF	250	15	0.230	0.150	4.01(h)	9.14(m)
BPRMF	350	0.3	0.238	0.157	1.29(h)	6.64(m)
BPRkNN	1e-4	0.01	0.208	0.145	2.38(m)	8.15(m)
SLIM	5	0.5	**0.246**	0.170	17.24(m)	13.57(s)
fsSLIM	100	0.5	0.243	0.168	4.97(m)	4.45(s)
fsSLIM	50	0.5	0.244	0.169	2.40(m)	3.34(s)

The description of each row and column is available in Section 4.4.1.

Table 4.3: Comparison of *Top-N* Recommendation Algorithms on ctlg2

method	params		HR	ARHR	mt	tt
itemkNN	10	-	0.222	0.108	33.78(s)	1.19(s)
itemprob	10	0.5	0.222	0.105	47.86(s)	0.99(s)
userkNN	50	-	0.204	0.106	0.37(s)	48.45(s)
PureSVD	1300	10	0.196	0.099	3.95(m)	18.46(m)
WRMF	300	10	0.235	0.114	20.42(h)	5.49(s)
BPRMF	400	0.1	0.249	0.123	9.72(h)	3.14(m)
BPRkNN	0.001	0.001	0.224	0.104	1.28(h)	22.03(m)
SLIM	5	2.0	**0.272**	0.140	7.24(h)	26.98(s)
fsSLIM	100	1.0	**0.282**	0.149	10.92(m)	5.00(s)
fsSLIM	10	0.5	0.262	0.138	4.21(m)	2.01(s)

The description of each row and column is available in Section 4.4.1.

Table 4.4: Comparison of *Top-N* Recommendation Algorithms on ctlg3

method	params		HR	ARHR	mt	tt
itemkNN	300	-	0.544	0.313	0.55(s)	6.66(s)
itemprob	400	0.3	0.558	0.322	0.87(s)	7.62(s)
userkNN	350	-	0.492	0.285	0.11(s)	19.18(s)
PureSVD	3000	10	0.373	0.210	1.11(h)	4.28(h)
WRMF	420	20	0.543	0.308	14.42(h)	50.67(m)
BPRMF	300	0.5	0.541	0.283	1.49(h)	13.66(m)
BPRkNN	0.001	1e-4	0.542	0.304	6.20(m)	20.28(m)
SLIM	3	0.5	**0.579**	0.347	1.02(h)	16.23(s)
fsSLIM	100	0.0	0.546	0.292	12.57(m)	9.62(s)
fsSLIM	400	0.5	0.570	0.339	14.27(m)	12.52(s)

The description of each row and column is available in Section 4.4.1.

Table 4.5: Comparison of *Top-N* Recommendation Algorithms on ecmrc

method	params		HR	ARHR	mt	tt
itemkNN	300	-	0.218	0.125	0.06(s)	0.54(s)
itemprob	30	0.2	0.245	0.138	0.09(s)	0.12(s)
userkNN	400	-	0.212	0.119	0.01(s)	0.78(s)
PureSVD	1900	10	0.186	0.110	3.67(m)	3.22(m)
WRMF	270	15	0.242	0.133	3.22(h)	13.60(s)
BPRMF	350	0.1	0.249	0.128	4.00(m)	12.76(s)
BPRkNN	1e-5	0.010	0.242	0.130	1.02(m)	13.53(s)
SLIM	5	0.5	**0.255**	0.149	11.10(s)	0.51(s)
fsSLIM	100	0.5	0.252	0.147	16.89(s)	0.32(s)
fsSLIM	30	0.5	0.252	0.147	5.41(s)	0.16(s)

The description of each row and column is available in Section 4.4.1.

Table 4.6: Comparison of *Top-N* Recommendation Algorithms on BX

method	params		HR	ARHR	mt	tt
itemkNN	10	-	0.085	0.044	1.34(s)	0.08(s)
itemprob	30	0.3	0.103	0.050	2.11(s)	0.22(s)
userkNN	100	-	0.083	0.039	0.01(s)	1.49(s)
PureSVD	1500	10	0.072	0.037	1.91(m)	2.57(m)
WRMF	400	5	0.086	0.040	12.01(h)	29.77(s)
BPRMF	350	0.1	0.089	0.040	8.95(m)	12.44(s)
BPRkNN	1e-4	0.010	0.082	0.035	5.16(m)	42.23(s)
SLIM	3	0.5	**0.109**	0.055	5.51(m)	1.39(s)
fsSLIM	100	0.5	0.109	0.053	36.26(s)	0.63(s)
fsSLIM	30	1.0	0.105	0.055	16.07(s)	0.18(s)

The description of each row and column is available in Section 4.4.1.

Table 4.7: Comparison of *Top-N* Recommendation Algorithms on ML10M

method	params		HR	ARHR	mt	tt
itemkNN	20	-	0.238	0.106	1.97(m)	8.93(s)
itemprob	20	0.5	0.237	0.106	1.88(m)	7.49(s)
userkNN	50	-	0.303	0.146	2.26(s)	34.42(m)
PureSVD	170	10	0.294	0.139	1.68(m)	1.72(m)
WRMF	100	2	0.306	0.139	16.27(h)	1.59(m)
BPRMF	350	0.1	0.281	0.123	4.77(h)	5.20(m)
BPRkNN	0.001	1e-4	**0.327**	0.156	15.78(h)	1.08(h)
SLIM	1	2.0	0.311	0.153	50.98(h)	41.59(s)
fsSLIM	100	0.5	0.311	0.152	37.12(m)	17.97(s)
fsSLIM	20	1.0	0.298	0.145	14.26(m)	8.87(s)

The description of each row and column is available in Section 4.4.1.

Table 4.8: Comparison of *Top-N* Recommendation Algorithms on Netflix

method	params		HR	ARHR	mt	tt
itemkNN	150	-	0.178	0.088	24.53(s)	13.17(s)
itemprob	10	0.5	0.177	0.083	30.36(s)	1.01(s)
userkNN	200	-	0.154	0.077	0.33(s)	1.04(m)
PureSVD	3500	10	0.182	0.092	29.86(m)	21.29(m)
WRMF	350	10	0.184	0.085	22.47(h)	2.63(m)
BPRMF	400	0.1	0.156	0.071	43.55(m)	3.56(m)
BPRkNN	0.01	0.01	0.188	0.092	10.91(m)	6.12(m)
SLIM	5	1.0	**0.200**	0.102	7.85(h)	9.84(s)
fsSLIM	100	0.5	**0.202**	0.104	6.43(m)	5.73(s)
fsSLIM	150	0.5	**0.202**	0.104	9.09(m)	7.47(s)

The description of each row and column is available in Section 4.4.1.

Table 4.9: Comparison of *Top-N* Recommendation Algorithms on Yahoo

method	params		HR	ARHR	mt	tt
itemkNN	400	-	0.107	0.041	21.54(s)	2.25(m)
itemprob	350	0.5	0.107	0.041	34.23(s)	1.90(m)
userkNN	50	-	0.107	0.041	18.46(s)	3.26(m)
PureSVD	170	10	0.074	0.027	53.05(s)	11.18(m)
WRMF	200	8	0.090	0.032	16.23(h)	50.05(m)
BPRMF	400	0.1	0.093	0.033	10.36(h)	47.28(m)
BPRkNN	0.01	0.001	0.104	0.038	2.60(h)	4.11(h)
SLIM	5	0.5	**0.122**	0.047	21.30(h)	5.69(m)
fsSLIM	100	0.5	**0.124**	0.048	1.39(m)	41.24(s)
fsSLIM	400	0.5	**0.123**	0.048	2.41(m)	1.72(m)

The description of each row and column is available in Section 4.4.1.

These results show that SLIM produces recommendations that are consistently better (in terms of HR and ARHR) than other methods over all the datasets except ML10M (SLIM has HR 0.311 on ML10M, which is only worse than BPRkNN with HR 0.327). In term of HR, SLIM

is on average 19.67%, 12.91%, 22.41%, 50.80%, 13.42%, 14.32% and 12.95% better than itemkNN, itemprob, userkNN, PureSVD, WRMF, BPRMF and BPRkNN, respectively, over all the eight datasets. Similar performance gains can also be observed with respect to ARHR. Among the three MF-based models, WRMF and BPRMF have similar performance, which is substantially better than PureSVD on all datasets except ML10M and Netflix. BPRkNN has better performance than MF methods on large datasets (i.e., ML10M, Netflix and Yahoo) but worse than MF methods on small datasets.

In terms of recommendation efficiency, SLIM is comparable to itemkNN and itemprob (i.e., the required times range in seconds), but considerably faster than the other methods (i.e., the required times range in minutes). The somewhat worse efficiency of SLIM compared to itemkNN is due to the fact that the resulting best S matrix is denser than the best performing item-item cosine similarity matrix from itemkNN. PureSVD, WRMF and BPRMF have worse computational complexities (i.e., linear to the product of the number of items and the dimensionality of the latent space), which is validated by their high recommendation run time. BPRkNN produces a fully dense item-item similarity matrix, which is responsible for its high recommendation time.

In terms of the amount of time required to learn the models, we see that the time required by itemkNN/itemprob is substantially smaller than the rest of the methods. The amount of time required by SLIM to learn its model, relative to PureSVD, WRMF, BPRMF and BPRkNN, varies depending on the datasets. However, even though SLIM is slower on some datasets (e.g., ML10M and Yahoo), this situation can be easily remediated by the feature-selection-based fsSLIM as will be discussed later in Section 4.4.1.

One thing that is surprising with the results shown in Table 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9 is that the MF-based methods are sometimes even worse than the simple itemkNN, itemprob and userkNN in terms of HR. For example, BPRMF performs worse for BX, ML10M, Netflix and Yahoo. This may because that in the BPRMF paper [24], the authors evaluated the entire AUC curve to measure if the interested items are ranked higher than the rest. However, a good AUC value does not necessarily lead to good performance on *top-N* of the ranked list. In addition, in the case of PureSVD, the best performance is achieved when a rather larger number of singular values is used (e.g., ccard, ctlg3, BX and Netflix).

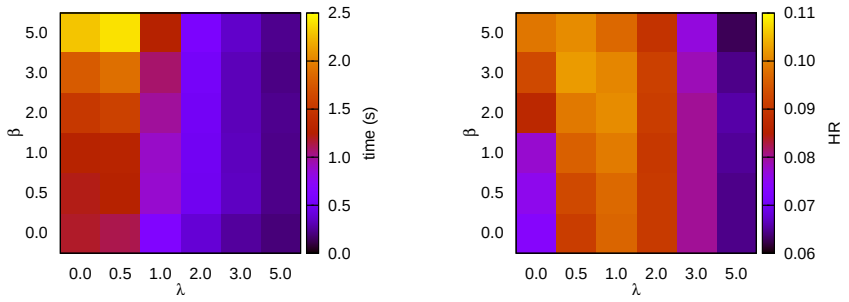
`fsSLIM` Performance

Table 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9 also present the results for the `SLIM` version that utilizes feature selection (rows labeled as `fsSLIM`). In the first set of experiments we used the item-item cosine similarity (as in `itemkNN`) and for each column of M selected its 100 other most similar columns and used the $m \times 100$ matrix to estimate the coefficient matrix in Equation 4.3. The results are shown in the first `fsSLIM` row in Table 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9. In the second set of experiments we selected the most similar columns of M based on item-item cosine similarity or item-item probability similarity (as in `itemprob`), whichever performs best, and corresponding number of columns. The results of these experiments are shown in the second `fsSLIM` row.

There are three important observations that can be made from the `fsSLIM` results. First, the performance of `fsSLIM` is comparable to that of `SLIM` for nearly all the datasets. Second, the amount of time required by `fsSLIM` to learn the model is much smaller than that required by `SLIM`. Third, using `fsSLIM` to estimate S, whose sparsity structure is constrained by the `itemkNN`/`itemprob` neighbors, leads to significantly better recommendation performance than `itemkNN`/`itemprob` itself. This suggests that we can utilize feature selection to improve the learning time without decreasing the performance.

Regularization Effects in `SLIM`

Figure 4.1 shows the effects of ℓ_1-norm and ℓ_2-norm regularizations in terms of recommendation time (which directly depends on how sparse S is.) and HR on the dataset BX(similar results are observed from all the other datasets). Figure 4.1 demonstrates that as greater ℓ_1-norm regularization (i.e., larger λ in Equation 4.3) is applied, lower recommendation time is achieved, indicating that the learned S is sparser. Figure 4.1 also shows the effects of ℓ_1-norm and ℓ_2-norm regularizations together for recommendation quality. The best recommendation quality is achieved when both of the regularization parameters β and λ are non-zero. In addition, the recommendation quality changes smoothly as the regularization parameters β and λ change.



(a) Recommendation time (b) HR

Figure 4.1: ℓ_1-Norm and ℓ_2-Norm Regularization Effects on BX

`SLIM` for the Long-Tail Distribution

The long-tail effect, which refers to the fact that a disproportionally large number of purchases/ratings are condensed in a small number of items (popular items), has been a concern for recommender systems. Popular items tend to dominate the recommendations, making it difficult to produce novel and diverse recommendations.

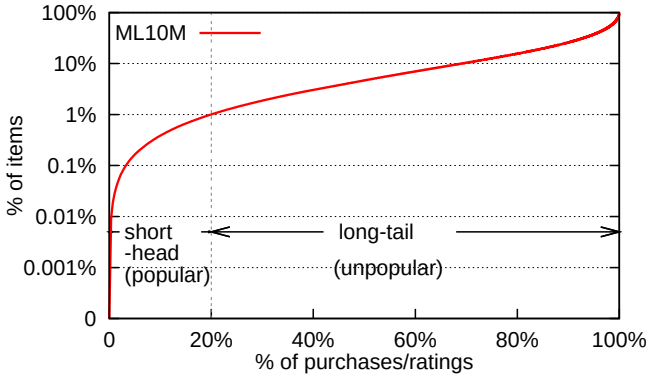


Figure 4.2: Purchase/Rating Distribution in ML10M

Table 4.10: Performance on the Long Tail of ML10M

method	**ML10M** long tail					
	params		HR	ARHR	mt	tt
itemkNN	10	-	0.130	0.052	1.59(m)	4.62(s)
itemprob	10	0.5	0.126	0.051	1.65(m)	4.04(s)
userkNN	50	-	0.162	0.069	2.10(s)	20.43(m)
PureSVD	350	70	0.224	0.096	2.98(m)	10.45(m)
WRMF	100	2	0.232	0.097	23.15(h)	1.74(m)
BPRMF	300	0.01	0.240	0.102	22.63(h)	8.56(m)
BPRkNN	0.001	1e-4	0.239	0.098	15.72(h)	36.42(m)
SLIM	1	5.0	**0.256**	0.106	57.55(h)	47.69(s)
fsSLIM	10	5.0	0.255	0.105	25.37(m)	9.57(s)
fsSLIM	100	4.0	0.255	0.105	58.32(m)	19.32(s)

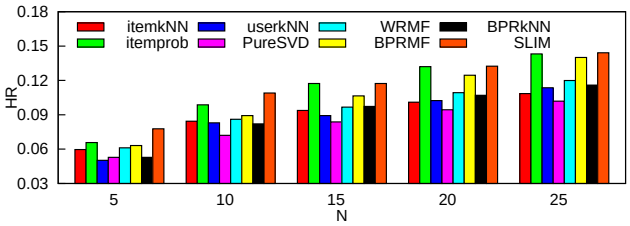
1% most popular items are eliminated from ML10M. Params have same meanings as those in Table 4.7.

Since while constructing the BX, Netflix and Yahoo datasets, we eliminated the items that are purchased/rated by many users, these datasets do not suffer from long-tail effects. The results presented in Table 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9 as they relate to these datasets demonstrate that SLIM is superior to other methods in producing non-trivial *top-N* recommendations when no significantly popular items are present.

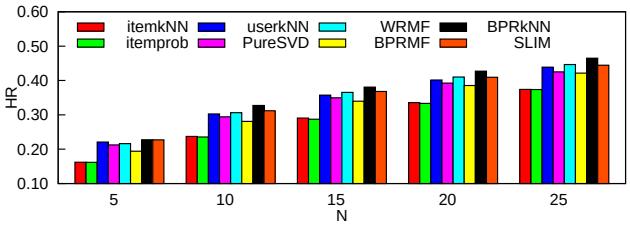
The plot in Figure 4.2 demonstrates the long-tail distribution of the items in ML10M dataset, in which only 1% of the items contribute 20% of the ratings. We eliminate these 1% most popular items and use the remaining ratings for all the *top-N* methods during learning. The results are presented in Table 4.10. These results show that the performance of all methods is notably worse than the corresponding performance in Table 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9 in which the "short head" (i.e., corresponding to the most popular items) is present. However, SLIM outperforms the rest methods. In particular, SLIM outperforms BPRkNN even though BPRkNN does better than SLIM as in Table 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9 when the popular items are presented in ML10M. This conforms to the observations based on BX, Netflix and Yahoo results, that SLIM is resistant to the long-tail effects.

Recommendation for Different *Top-N*

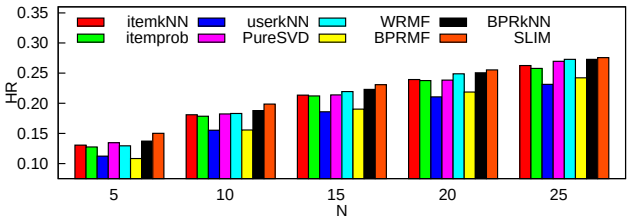
Figure 4.3 shows the performance of the methods for different values of N (i.e., 5, 10, 15, 20 and 25) for BX, ML10M, Netflix and Yahoo datasets. Table 4.11 presents the performance difference between SLIM and the best of the other methods in terms of HR on the four datasets. For example, 0.012 in Table 4.11 for BX when $N = 5$ is calculated as the difference between SLIM's HR and the best HR from all the other methods on BX when top-5 items are recommended. The performance difference between SLIM and the best of the other methods are higher for smaller values of N across the datasets BX, ML10M and Netflix. Figure 4.3 and Table 4.11 demonstrate that SLIM produces better than the other methods when smaller number of items are recommended. This indicates that SLIM tends to rank most relevant items higher than the other methods.



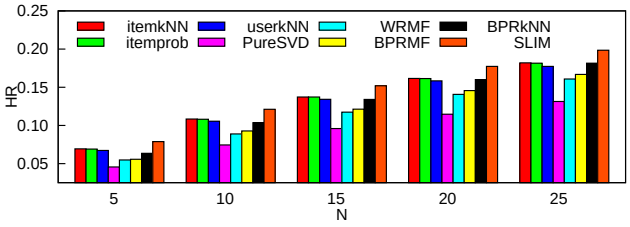
(a) BX



(b) ML10M



(c) Netflix



(d) Yahoo

Figure 4.3: Recommendations for Different Values of N

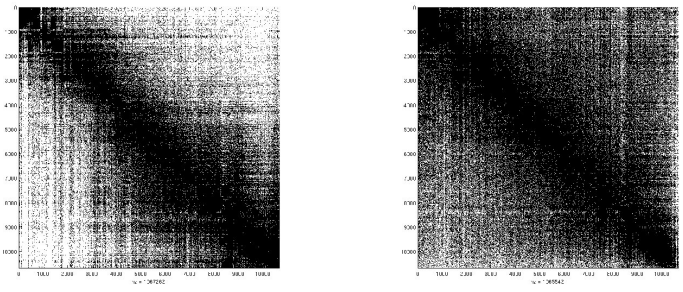
Table 4.11: Performance Difference on *Top-N* Recommendations

dataset	*N*				
	5	10	15	20	25
BX	0.012	0.006	0.000	0.000	0.001
ML10M	0.000	-0.016	-0.013	-0.018	-0.021
Netflix	0.013	0.012	0.008	0.005	0.003
Yahoo	0.009	0.015	0.015	0.016	0.017

Columns corresponding to *N* shows the performance (in terms of HR) difference between SLIM and the best of the rest methods on corresponding *top-N* recommendations.

Sparsity Pattern of S

We use ML10M as an example to illustrate what SLIM is learning. The item-item similarity matrix S constructed from itemkNN and the S from SLIM are shown in Figure 4.4. Note that in Figure 4.4, the S matrix is obtained using 100 nearest neighbors.



(a) S from itemkNN (b) S from SLIM

Figure 4.4: Sparsity Patterns for ML10M (dark for non-zeros)

The density of the matrices produced by itemkNN and SLIM is 0.936% and 0.935%, respectively. However, their sparsity patterns are different. First, the S matrix has non-zero item-item

similarity values that are clustered towards the diagonal, while S has non-zero values that are more evenly distributed. Second, during recommendation, on average 53.60 non-zero values in S contribute to the recommendation score calculation on one item for one user, whereas in case of S, on average 14.79 non-zero values make contributions, which is 1/3 as that in S.

S recovers 31.8% of the non-zero entries in S (those entries have larger values than average) and it also discovers new non-zero entries that are not in S. The newly discovered item-item similarities contribute to 37.1% of the hits from S. This suggests that S, even though it is also very sparse, recovers some subtle relations that are not captured by item-item cosine similarity measure, which brings performance improvement. In SLIM, item t_k that are co-purchased with item t_j also contributes to the similarity between item t_j and another item t_i, even t_k has never been co-purchased with t_i. Furthermore, treating missing values as 0s helps to generalize. Including all missing values as 0s in s_j vector in Equation 4.4 help smooth out item similarities and help incorporate the impacts from dissimilar/un-co-purchased items. The above can be shown theoretically by the coordinate descent updates (proof omitted here).

Matrix Reconstruction

We compare SLIM with MF methods by looking at how it reconstructs the user/item purchase matrix. We use BPRMF as an example of MF methods since it has the typical properties that most of the state-of-the-art MF methods have. We focused on ML10M, whose matrix M has a density of 1.3%. The reconstructed matrix $\tilde{M}_{\texttt{SLIM}} = MS$ from SLIM has a density 25.1%, whose non-zero values have a mean of 0.0593. For those 1.3% non-zero entries in M, $\tilde{M}_{\texttt{SLIM}}$ recovered 99.1% of them and their mean value is 0.4489 (i.e., 7.57 times of the non-zero mean). The reconstructed matrix $\tilde{M}_{\texttt{BPRMF}} = UV^{\mathsf{T}}$ is fully dense, with 13.1% of its values greater than 0 with mean of 1.8636, and 86.9% of its values less than 0 with a mean of -2.4718. For those 1.3% non-zero entries in M, $\tilde{M}_{\texttt{BPRMF}}$ has 97.3% of them as positive values with a mean of 4.7623 (i.e., 2.56 times of positive mean). This suggests that SLIM recovers M better than BPRMF since SLIM recovers more non-zero entries with relatively much larger values.

4.4.2 SLIM Performance on Ratings

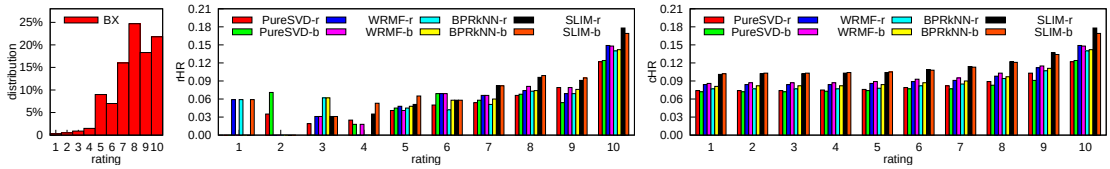
Comparison Algorithms

We compare the performance of SLIM with PureSVD, WRMF and BPRkNN. In SLIM, the S matrix is learned by using the user-item rating matrix M as in Equation 4.2. PureSVD also uses the user-item rating matrix for the SVD calculation. In WRMF, the ratings are used as weights following the approach suggested in [18]. We modified BPRkNN such that in addition to ranking rated items higher than non-rated items, they also rank high-rated items higher than low-rated items. We will use the suffix -*r* after each method to explicitly denote that a method utilizes rating information during the model construction. Similarly, we will use the suffix -*b* in this section to denote that a method utilizes binary information as in Section 4.4.1 for comparison purpose.

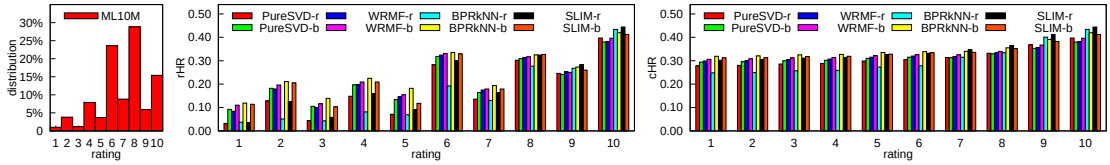
Top-N Recommendation Performance on Ratings

We compare SLIM-*r* with PureSVD-*r*, WRMF-*r* and BPRkNN-*r* on the BX, ML10M, Netflix and Yahoo datasets for which rating information is available. In addition, we also evaluated SLIM-*b*, PureSVD-*b*, WRMF-*b* and BPRkNN-*b* on the four datasets, for which the models are still learned from binary user-item purchase matrix but the recommendations are evaluated based on ratings.

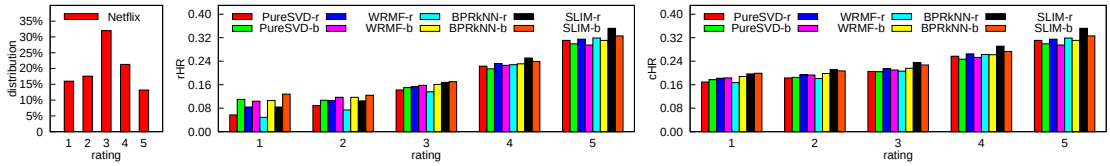
Figure 4.5 presents the results of these experiments. The first column of figures show the rating distribution of the four datasets. The second column of figures show the per-rating hit rates (rHR) for the four datasets. Finally, the third column of figures show the cumulative hit rates (cHR) for the four datasets. In Figure 4.5, the binary *top-N* models correspond to the best performing models of each method as in Table 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9. The results from *top-N* models using ratings are selected based on the cHR performance on rating 6, 6, 3 and 3 for the datasets, respectively.



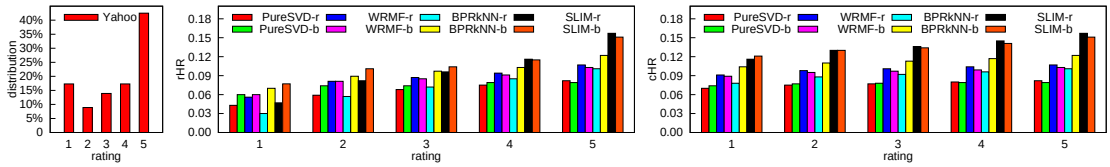
(a) BX distribution (b) BX rHR (c) BX cHR



(d) ML10M distribution (e) ML10M rHR (f) ML10M cHR



(g) Netflix distribution (h) Netflix rHR (i) Netflix cHR



(j) Yahoo distribution (k) Yahoo rHR (l) Yahoo cHR

Figure 4.5: *Top-N* Recommendation Performance on Ratings

The results in Figure 4.5 show that all the *-r* methods tend to produce higher hit rates on items with higher ratings. However, the per-rating hit rates of the *-b* methods have smaller dynamics across different ratings. This is because during learning, high-rated items and low-rated items are not differentiated in the *-b* methods. In addition, the *-r* methods outperform *-b* methods in terms of rHR on high-rated items. In particular, *-r* methods consistently outperform

-*b* methods on items with ratings above the average across all the datasets.

Figure 4.5 also shows that the SLIM-*r* consistently outperforms the other methods in terms of both rHR and cHR on items with higher ratings over all the four datasets. In particular, it outperforms PureSVD-*r* in terms of cHR, which is demonstrated in [16] to be the best performing methods for *top-N* recommendation using ratings. This indicates that incorporating rating information during learning allows the SLIM methods to identify more highly-rated items.

4.5 Discussion & Conclusions

4.5.1 Observed Data vs Missing Data

In the user-item purchase/rating matrix M, the non-zero entries represent purchase/rating activities. However, the entries with "0" value can be ambiguous. They may either represent that the users will never purchase the items, the users may purchase the items but have not done so, or we do not know if the users have purchased the items or not or if they will. This is the typical "missing data" setting and it has been well studied in recommender systems [17, 21].

In SLIM, we treated all missing data in $\mathbf{m}_j$ and M in Equation 4.4 as true negative(i.e., the users will never purchase the items). Differentiation of observed data and missing data in Equation 4.4 is under development.

4.5.2 Conclusions

In this chapter, we proposed a sparse linear method for *top-N* recommendation, which is able to generate high-quality *top-N* recommendations fast. SLIM employs a sparse linear model in which the recommendation score for a new item can be calculated as an aggregation of other items. A sparse aggregation coefficient matrix S is learned for SLIM to make the aggregation very fast. S is learned by solving an ℓ_1-norm and ℓ_2-norm regularized optimization problem such that sparsity is introduced into S.

We conducted a comprehensive set of experiments and compared SLIM with other state-of-the-art *top-N* recommendation algorithms. The results showed that SLIM achieves prediction quality better that the state-of-the-art MF-based methods. Moreover, SLIM generates recommendations very fast. The experimental results also demonstrated the good properties of SLIM compared to other methods. Such properties include that SLIM is able to have significant speedup

if feature selection is applied prior to learning. SLIM is also resistant to long-tail effects in *top-N* recommendation problems. In addition, when trained using ratings, SLIM tends to produce recommendations that are also potentially highly rated. Due to these properties, SLIM is very suitable for real-time *top-N* recommendation tasks.

Chapter 5

Sparse Linear Methods with Side Information for Top-N Recommendations

The increasing amount of side information associated with the items in E-commerce applications has provided a very rich source of information that, once properly exploited and incorporated, can significantly improve the performance of the conventional recommender systems. This chapter focuses on developing effective algorithms that utilize item side information for *top-N* recommender systems. A set of sparse linear methods with side information (SSLIM) is proposed, which involve a regularized optimization process to learn a sparse aggregation coefficient matrix based on both user-item purchase profiles and item side information. This aggregation coefficient matrix is used within an item-based recommendation framework to generate recommendations for the users. Our experimental results demonstrate that SSLIM outperforms other methods in effectively utilizing side information and achieving performance improvement.

5.1 Introduction

The conventional *top-N* recommendation algorithms as discussed in Chapter 4 primarily focus on utilizing user-item purchase profiles to generate recommendations. With the increased availability of additional information associated with the items (e.g., product reviews, movie

plots, etc), referred to as *side information*, there is a greater interest in taking advantage of such information to improve the quality of conventional *top-N* recommender systems. As a result, a number of approaches have been developed from Machine Learning (ML) and Information Retrieval (IR) communities for incorporating side information. Such approaches include hybrid methods [59], matrix/tensor factorization [25, 28], and other regression methods [26].

In this chapter, we propose a set of Sparse Linear Methods that utilize the item Side information (SSLIM) for *top-N* recommendation. These methods include collective SLIM (cSLIM), relaxed collective SLIM (rcSLIM), side information induced SLIM (fSLIM) and side information induced double SLIM (f2SLIM). The key idea behind these methods is to learn linear models that are constrained and/or informed by the relations between the item side information and the user-item purchase profiles so as to achieve better recommendation performance. We conduct a comprehensive set of experiments on various datasets from different real applications. The results show that SSLIM produces better recommendations than the state-of-the-art methods.

5.2 SLIM with Side Information

SLIM (Chapter 4) provides a general framework in which only the aggregation coefficient matrix S is needed for efficient *top-N* recommendations, and this matrix is learned from the user-item purchase profiles. In this chapter, we present four different extensions of SLIM that are designed to incorporate side information about the items in order to further improve the qualify of the recommendations.

5.2.1 Collective SLIM

The first approach assumes that there exist correlations between users' co-purchase behaviors on two items and the similarity of the two items' intrinsic properties encoded in their side information. In order to enforce such correlations, this approach imposes the additional requirement that both the user-item purchase profile matrix M and the item side information matrix F should be reproduced by the same sparse linear aggregation. That is, in addition to satisfying $M \sim MS$, the coefficient matrix S should also satisfy

$$F \sim FS. \tag{5.1}$$

This is achieved by learning the aggregation coefficient matrix S as the minimizer to the following optimization problem:

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \frac{1}{2}\|M - MS\|_F^2 + \frac{\alpha}{2}\|F - FS\|_F^2 \\ & + \frac{\beta}{2}\|S\|_F^2 + \lambda\|S\|_1 \\ \text{subject to} \quad & S \geq 0, \\ & \text{diag}(S) = 0, \end{aligned} \tag{5.2}$$

where $\|F - FS\|_F^2$ measures how well the aggregation coefficient matrix S fits the side information. The parameter α is used to control the relative importance of the user-item purchase information M and the item side information F when they are used to learn S. Note that in this method, the side information is actually used to regularize the original `SLIM` method (i.e., via adding the regularization term $\frac{\alpha}{2}\|F - FS\|_F^2$ into Equation 4.3 in Chapter 4). The recommendations are generated in exactly the same way as in `SLIM`. That is, the recommendation score for user u_i on item t_j is calculated as $\tilde{m}_{ij} = \mathbf{m}_i^\mathsf{T}\mathbf{s}_j$. Since the matrix S is learned from both M and F collectively by using F to regularize the original `SLIM` method, this approach is referred to as collective `SLIM` and denoted by `cSLIM`.

The solution to the optimization problem in Equation 5.2 is identical to the solution of an optimization problem in the same form as in Equation 4.3 with M in Equation 4.3 replaced by $M' = [M, \sqrt{\alpha}F]^\mathsf{T}$.

5.2.2 Relaxed cSLIM

The second approach also tries to reproduce the item side information using a sparse linear method as in `cSLIM`, but it uses an alternative approach for achieving this. Specifically, it uses an aggregation coefficient matrix Q to reproduce F as

$$F \sim FQ, \tag{5.3}$$

where Q is not necessarily identical to S as in Equation 4.2. Thus, this method is a relaxation from `cSLIM`. However, the two aggregation coefficient matrices S and Q are tied by requiring that Q should not be significantly different from S (i.e., $S \sim Q$). The matrix S and the matrix

Q in Equation 5.3 are learned as the minimizers of the following optimization problem:

$$\begin{aligned} \underset{S,Q}{\text{minimize}} \quad & \frac{1}{2}\|M - MS\|_F^2 + \frac{\alpha}{2}\|F - FQ\|_F^2 \\ & + \frac{\beta_1}{2}\|S - Q\|_F^2 + \frac{\beta_2}{2}(\|S\|_F^2 + \|Q\|_F^2) \\ & + \lambda(\|S\|_1 + \|Q\|_1) \\ \text{subject to} \quad & S \geq 0, \\ & Q \geq 0, \\ & \text{diag}(S) = 0, \\ & \text{diag}(Q) = 0, \end{aligned} \tag{5.4}$$

where the parameter β_1 controls how much S and Q are allowed to be different from each other. Similar to `cSLIM`, this method regularizes the original `SLIM` using item side information by adding the two regularization terms $\frac{\alpha}{2}\|F - FQ\|_F^2$ and $\frac{\beta_1}{2}\|S - Q\|_F^2$ and the recommendations are generated in the same way as in `SLIM`. Since this method is a relaxation from `cSLIM`, it is refereed to as relaxed collective `SLIM` and denoted by `rcSLIM`.

The optimization problem in Equation 5.4 can be solved via an approach alternating on solving S and Q. In each iteration, one variable is fixed and the problem becomes a regularized optimization problem with respect to the other variable, and it can be solved using a similar approach of stacking matrices as in Section 5.2.1. The solution of `cSLIM` is used as the initial value of S.

5.2.3 Side Information Induced SLIM

An alternative way to learn the aggregation coefficient matrix S of `SLIM` is to represent S as a function in the item feature space and thus it captures the feature-based relations of the items. One option of achieving this is to use the item-item similarity matrix calculated as FF^T, that is, the aggregation coefficient from one item to another is calculated as the dot-product of their feature vectors (i.e., item-item feature similarity). However, in this way, the aggregation coefficient matrix is not customized to the user-item purchase profiles M at all, and thus a `SLIM` with such aggregation coefficient matrix can fit M very poorly. Another way is to learn a weighting matrix W such that the aggregation coefficient value s_{ij} can be represented as a linear combination of item t_i's feature $\mathbf{f}_i$ weighted by item t_j's personalized weighting vector $\mathbf{w}_j$ over individual item

features, that is, $s_{ij} = \mathbf{f}_i^\mathsf{T}\mathbf{w}_j$ and $\mathbf{w}_j$ is W's j-th column. In this way, the coefficient matrix S can be represented as a weighted linear combination of the item features F using W, that is,

$$S = F^\mathsf{T}W. \tag{5.5}$$

Such weighting matrix W can be learned as the minimizer of the following optimization problem:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \frac{1}{2}\|M - M(F^\mathsf{T}W - D)\|_F^2 \\ & + \frac{\beta}{2}\|W\|_F^2 + \lambda\|F^\mathsf{T}W\|_1 \\ \text{subject to} \quad & W \geq 0 \\ & D = \text{diag}(\text{diag}(F^\mathsf{T}W)), \end{aligned} \tag{5.6}$$

where $D = \text{diag}(\text{diag}(F^\mathsf{T}W))$ is a diagonal matrix with the corresponding diagonal values from $F^\mathsf{T}W$. D is subtracted from $F^\mathsf{T}W$ so as to ensure that m_{ij} is not used to compute $\tilde{m}_{ij}$, and this is equivalent to the constraint $\text{diag}(S) = 0$ in Equation 4.3. In this method, the recommendation score for user u_i on item t_j is calculated as $\tilde{m}_{ij} = \mathbf{m}_i^\mathsf{T}(F^\mathsf{T}\mathbf{w}_j - \mathbf{d}_j)$, where $\mathbf{w}_j$ and $\mathbf{d}_j$ is the j-th column of W and D, respectively. Since this method explicitly specifies the aggregation coefficient matrix S as a function of the item side information F, it is referred to as side information induced `SLIM` and denoted by `fSLIM`.

The optimal solution to the optimization problem in Equation 5.6 is

$$W^* = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_j, \cdots, \mathbf{w}_n],$$

where $\mathbf{w}_j$ is the optimal solution to the following problem:

$$\begin{aligned} \underset{\mathbf{w}_j}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{m}_j - MF_{-j}^\mathsf{T}\mathbf{w}_j\|_F^2 + \frac{\beta}{2}\|\mathbf{w}_j\|_F^2 + \lambda\|\mathbf{c}\mathbf{w}_j\|_1 \\ \text{subject to} \quad & \mathbf{w}_j \geq 0, \end{aligned}$$

where $c_p = \sum_{k=1}^{n} f_{pk}$, and F_{-j} is a matrix with F's j-th column set to 0.

5.2.4 Side Information Induced Double SLIM

`SLIM` and `fSLIM` have their own advantages. `SLIM` learns the aggregation coefficient matrix S purely from purchase profiles such that it better fits the user-item purchase information. `fSLIM`

forces the aggregation coefficient matrix S to be expressed in the item feature space and therefore it captures useful information from the item features. `SLIM` and `fSLIM` can be coupled within one method so as to leverage both their advantages and better learn from purchase profiles and side information concurrently. One way to combine `SLIM` and `fSLIM` is to have the user-item purchase profile M reproduced by both `SLIM` and `fSLIM` as

$$M \sim MS + M(F^\top W - D), \tag{5.7}$$

where the S and W matrices can be learned as the minimizers of the following optimization problem:

$$\begin{aligned} \underset{S,W}{\text{minimize}} \quad & \frac{1}{2}\|M - MS - M(F^\top W - D)\|_F^2 \\ & + \frac{\beta}{2}(\|S\|_F^2 + \|W\|_F^2) + \lambda(\|S\|_1 + \|F^\top W\|_1) \\ \text{subject to} \quad & S \geq 0, \\ & W \geq 0, \\ & \text{diag}(S) = 0, \\ & D = \text{diag}(\text{diag}(FW)). \end{aligned} \tag{5.8}$$

In this method, the recommendation score for user u_i on item t_j is calculated as $\tilde{m}_{ij} = \mathbf{m}_i^\top \mathbf{s}_j + \mathbf{m}_i^\top (F^\top \mathbf{w}_j - \mathbf{d}_j)$, where $\mathbf{w}_j$ and $\mathbf{d}_j$ is the j-th column of W and D, respectively. This method is a combination of `SLIM` and `fSLIM` and thus it is refereed as side information reduced double `SLIM` and denoted by `f2SLIM`.

That the optimal solution of W in the problem in Equation 5.8 is identical to the first l rows of the optimal solution W' to the problem in Equation 5.6 with F replaced by $[F, I]^\top$ where I is an $n \times n$ identity matrix, and $D' = \text{diag}((F')^\top W')$, whereas the optimal S is the last n rows of W'.

5.3 Materials

5.3.1 Datasets

Table 5.1: The Datasets Used in Evaluation: Purchase Information

dataset	#users	#items	#nnzs	rsize	csize	density
ML100K	943	1,682	100,000	106.0	59.5	6.30%
NF	3,086	6,909	128,134	41.5	18.6	0.60%
CrossRef	84,260	23,458	466,068	5.5	19.9	0.02%
Lib	13,843	12,123	103,428	7.47	8.53	0.06%
BBY	127,285	7,330	162,451	1.3	22.2	0.02%
Yelp	13,574	6,896	89,608	6.6	13.0	0.10%

Columns corresponding to purchase information and side information show the dataset statistics for historical profile matrix M and side information matrix F, respectively. Under purchase information, column corresponding to #users, #items and #nnzs show the number of users, items and non-zero values in each dataset, respectively. Column corresponding to rsize, csize and density shows the average row density, the average column density and the matrix density, respectively. Under side information, column corresponding to desc shows the side information types. Column corresponding to #ftr and #nnz show the dimensionality of side information and the number of non-zero values in the side information, respectively. Column corresponding to srsize, scsize and sdensity show the average row density, the average column density and the density of the side information matrix, respectively.

Table 5.2: The Datasets Used in Evaluation: Side Information

dataset	desc	#ftr	#nnz	srsize	scsize	sdensity
ML100K	plots	2,327	46,915	27.9	20.2	1.20%
NF	plots	5,941	200,148	29.0	33.7	0.49%
CrossRef	titles	5,677	149,839	6.4	26.4	0.11%
Lib	titles	9,991	86,065	7.1	8.6	0.07%
BBY	reviews	9,686	1,912,444	260.9	197.4	2.7%
Yelp	reviews	10,305	330,865	48.0	32.1	0.47%

Column corresponding to desc shows the side information types. Column corresponding to #ftr and #nnz show the dimensionality of side information and the number of non-zero values in the side information, respectively. Column corresponding to srsize, scsize and sdensity show the average row density, the average column density and the density of the side information matrix, respectively.

We evaluated the performance of different methods on the following real datasets: ML100K, NF, CrossRef, Lib, BBY, and Yelp, whose characteristics are summarized in Table 5.1 and Table 5.2.

ML100K The ML100K dataset corresponds to movie ratings and was obtained from the MovieLens research project. The movie plots were fetched from the IMDb database and the words that appear in at least 5 plots are used as the movie side information.

NF The NF dataset is a subset extracted from the Netflix Prize dataset. The item side information was generated as in the ML100K dataset. Only the movies hat were rated by 10-30 users were selected.

CrossRef The CrossRef dataset was obtained from crossref.org, and contains scientific articles and lists of article citations. All the articles (i.e., references) that have DOI links and are cited by at least 50 other articles were first selected. Then the articles which cite more than 3 of such references were selected. In this way, an article-reference dataset is constructed, in which the articles (the references) are analogous to the users (the items). The words in the reference titles are used as side information. The *top-N* recommendation on CrossRef dataset becomes a task to recommend a reference for a certain article.

Lib The Lib dataset was obtained from the University of Minnesota libraries, and contains the library users and their viewed articles. From the entire library records, the users who viewed at least 5 different articles and the articles that were viewed by at least 10 users were collectively selected to construct an user-article matrix. The words in the article titles are used as the article side information. The *top-N* recommendation on Lib is to recommend an article to a user.

BBY The BBY dataset is a subset of the BestBuy user-product rating and review dataset from BestBuy website (https://developer.bestbuy.com/documentation/archives). The products that were reviewed by at least 5 users and the users who reviewed at least 2 such products were collectively selected so as to construct the dataset. The side information for each item was the text of all the reviews of that item.

Yelp The Yelp dataset is a subset of the academic version of Yelp user-business rating and review dataset downloaded from Yelp (http://www.yelp.com/academic_dataset). The users who reviewed at least 3 businesses and the corresponding businesses were selected to construct the dataset. The side information for each item was constructed from the reviews in a way similar to the BBY dataset.

For the original rating datasets (i.e., ML100K, NF, BBY, Yelp), the multivariate rating values

were converted to 1's.

5.3.2 Side Information Representation

Besides the learning capability of the SSLIM methods, the representation of the side information can impact the overall performance. Since the side information in our datasets is text (e.g., movie plots, product reviews, etc), we investigated different text representations. In all of these schemes, the text of the side information was preprocessed to eliminate stop words and each word was converted to its stem[1] .

Binary Representation (F_{b}) In this scheme, the text of the side information is represented using the bag-of-words model, and the frequency of each word is set to one. The reason for the binarization is that typically the text for an item is short, and there are not many informative words occurring multiple times, and thus a binarized vector is almost same as the original count vector. In addition, since the user-item profile M is binary, intuitively the item-item coefficient matrix Q learned from a binary feature matrix F should be comparable to the aggregation coefficient matrix S learned from M in terms of the values. In this case, the regularization using Q on S (i.e., the $\frac{\beta_1}{2}\|S - Q\|_F^2$ term in rcSLIM) can be more effective.

Normalized TFIDF Representation ($\mathsf{F}_{\mathsf{tfidf}}$) For the methods that directly learn from the item text (fSLIM and f2SLIM), it is essential that the text presentation encodes how important a word is in the text. For this purpose, a normalized TFIDF scheme is adopted. The TFIDF scheme [60] is widely used for weighting words in text mining. After the TFIDF scheme is applied on the feature vectors, the feature vectors are normalized to unit length.

Normalized TFIDF Representation with Feature Selection ($\mathsf{F}_{\mathsf{tfidf\_fs}}$) Another representation scheme is a modification of $\mathsf{F}_{\mathsf{tfidf}}$ by using feature selection. For each feature vector, the words were sorted in decreasing order according to their weights in the TFIDF representation. Then the highest weighted words were selected until cumulatively they contribute to 90% of the vector length.

Normalized TFIDF Binary Representation with Feature Selection ($\mathsf{F}_{\mathsf{tfidf\_fs\_b}}$) The last side information representation scheme is a compromise of F_{b} and $\mathsf{F}_{\mathsf{tfidf\_fs}}$, that is, it converts all the values that are calculated from $\mathsf{F}_{\mathsf{tfidf\_fs}}$ to binary. This scheme tries to use only the words that are considered as important by $\mathsf{F}_{\mathsf{tfidf\_fs}}$ and meanwhile still retain the advantages of the binary representations.

[1] http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/doc2mat-1.0.tar.gz

5.3.3 Comparison Methods

Singh *et al* [25] proposed the collective matrix factorization (`CMF`) method for relational learning as follow:

$$\begin{aligned}\underset{U,V,W}{\text{minimize}} \quad & \frac{1}{2}\|M - UV^{\mathsf{T}}\|_F^2 + \frac{\alpha}{2}\|F - WV^{\mathsf{T}}\|_F^2 \\ & + \frac{\beta}{2}(\|U\|_F^2 + \|V^{\mathsf{T}}\|_F^2 + \|W\|_F^2),\end{aligned} \tag{5.9}$$

where U is an $m \times k$ user factor from M, W is an $l \times k$ feature factor from F, and V^{T} is an $k \times n$ item factor which is collectively learned from both M and F. Particularly, $k \ll \min(m, n, l)$. `CMF` and `cSLIM` are similar in the sense that a common matrix is learned from both M and F concurrently. However, they are fundamentally different methods. The `cSLIM` method conforms to linear methods and it models the *top-N* recommendation process as an aggregation on items. On the contrary, `CMF` models the recommendation process in a low-dimension latent space.

Thu *et al* [18] proposed a weighted regularized matrix factorization (`WRMF`) method for *top-N* recommendation, which weights the purchase and nonpurchase activities in M differently using a weighting matrix C as follows:

$$\underset{U,V^{\mathsf{T}}}{\text{minimize}} \quad \frac{1}{2}\|C \circ (M - UV^{\mathsf{T}})\|_F^2 + \frac{\beta}{2}(\|U\|_F^2 + \|V^{\mathsf{T}}\|_F^2). \tag{5.10}$$

Inspired by this weighting method, we combined `WRMF` with `CMF` so as to have a collective weighted regularized matrix factorization method, denoted by `CWRMF`, as follows:

$$\begin{aligned}\underset{U,V^{\mathsf{T}},W}{\text{minimize}} \quad & \frac{1}{2}\|C \circ (M - UV^{\mathsf{T}})\|_F^2 + \frac{\alpha}{2}\|F - WV^{\mathsf{T}}\|_F^2 \\ & + \frac{\beta_1}{2}\|U\|_F^2 + \frac{\lambda}{2}\|V^{\mathsf{T}}\|_F^2 + \frac{\beta_2}{2}\|W\|_F^2,\end{aligned} \tag{5.11}$$

in which M and F are still collectively factorized but errors from M are weighted differently by C. We use `WRMF` and `CWRMF` as the comparison algorithms in the experiments. In addition, we use another two collaborative filtering methods for comparison purposes. The `itemkNN` method is a widely used item-based collaborative filtering method proposed in [15]. The `itemSI` method is a modification of `itemkNN`, in which the item similarities are calculated as a linear combination of the similarity values calculated from `itemkNN` and the cosine similarity values calculated from side information weighted by a parameter α.

5.4 Experimental Results

5.4.1 Overall Performance

Table 5.3: Methods without Side Information Performance on ML100K

feature	method	params				HR	ARHR
F_{no}	itemkNN	10	-	-	-	0.287	0.124
	WRMF	5	50	1	-	0.327	0.133
	SLIM	2	2	-	-	0.343	0.147

The description of each row and column is available in Section 5.4.1

Table 5.4: Methods with Side Information Performance on ML100K

feature	method	params				HR	ARHR	feature	method	params				HR	ARHR
F_b	itemSI	200	0.5	-	-	0.262	0.116	F_{tfidf}	itemSI	200	0.1	-	-	0.261	0.120
	CWRMF	1	0.5	50	1	0.327	0.139		CWRMF	1	0.5	50	1	0.327	0.139
	cSLIM	0.5	10	1	-	0.344	0.148		cSLIM	2	5	5	-	0.344	0.147
	rcSLIM	0.1	5	1	2	0.344	0.147		rcSLIM	1	2	2	2	0.345	0.147
	fSLIM	10	0.5	-	-	0.303	0.130		fSLIM	2	0.1	-	-	0.317	0.134
	f2SLIM	10	1	-	-	0.343	0.146		f2SLIM	5	5	-	-	0.341	0.147
F_{tfidf_fs}	itemSI	200	0.5	-	-	0.260	0.118	$F_{tfidf\_fs\_b}$	itemSI	200	0.1	-	-	0.262	0.121
	CWRMF	1	1	50	1	0.325	0.125		CWRMF	1	1	50	1	0.325	0.125
	cSLIM	2	5	5	-	0.344	0.147		cSLIM	0.5	5	2	-	0.346	0.151
	rcSLIM	1	2	2	2	0.345	0.147		rcSLIM	5	10	10	0.1	0.344	0.149
	fSLIM	2	0.1	-	-	0.317	0.133		fSLIM	20	1	-	-	0.312	0.134
	f2SLIM	5	5	-	-	0.343	0.148		f2SLIM	10	1	-	-	0.347	0.151

The description of each row and column is available in Section 5.4.1

Table 5.5: Methods without Side Information Performance on NF

feature	method	params				HR	ARHR
	itemkNN	300	-	-	-	0.043	0.015
F_{no}	WRMF	5	400	2	-	0.045	0.017
	SLIM	5	0.5	-	-	0.045	0.018

The description of each row and column is available in Section 5.4.1

Table 5.6: Methods with Side Information Performance on NF

feature	method	params				HR	ARHR	feature	method	params				HR	ARHR
	itemSI	200	0.5	-	-	0.042	0.016		itemSI	300	0.5	-	-	0.043	0.016
	CWRMF	1	0.5	200	2	0.028	0.009		CWRMF	1	0.5	200	2	0.028	0.009
F_b	cSLIM	0.1	10	1	-	0.047	0.017	F_{tfidf}	cSLIM	0.5	10	1	-	0.047	0.017
	rcSLIM	5	10	10	1	0.047	0.017		rcSLIM	0.5	10	2	1	0.047	0.018
	fSLIM	10	0.1	-	-	0.030	0.011		fSLIM	5	0.1	-	-	0.035	0.011
	f2SLIM	5	0.1	-	-	0.046	0.017		f2SLIM	10	0.5	-	-	0.046	0.017
	itemSI	50	0.5	-	-	0.042	0.015		itemSI	50	0.5	-	-	0.042	0.015
	CWRMF	1	2	200	1	0.031	0.010		CWRMF	1	2	200	1	0.031	0.010
F_{tfidf_fs}	cSLIM	1	10	1	-	0.047	0.017	$F_{tfidf\_fs\_b}$	cSLIM	0.5	5	0.5	-	0.047	0.018
	rcSLIM	1	10	10	0.1	0.047	0.017		rcSLIM	5	10	2	1	0.047	0.019
	fSLIM	5	0.1	-	-	0.034	0.012		fSLIM	10	0.1	-	-	0.029	0.011
	f2SLIM	10	5	-	-	0.046	0.017		f2SLIM	10	1	-	-	0.046	0.017

The description of each row and column is available in Section 5.4.1

Table 5.7: Methods without Side Information Performance on `CrossRef`

feature	method	params				HR	ARHR
F_{no}	`itemkNN`	300	-	-	-	0.404	0.212
	`WRMF`	1	400	10	-	0.352	0.169
	`SLIM`	5	0.5	-	-	0.408	0.211

The description of each row and column is available in Section 5.4.1

Table 5.8: Methods with Side Information Performance on `CrossRef`

feature	method	params				HR	ARHR	feature	method	params				HR	ARHR
F_b	`itemSI`	200	0.5	-	-	0.416	0.218	F_{tfidf}	`itemSI`	200	0.5	-	-	0.416	0.218
	`CWRMF`	1	2	200	2	0.168	0.080		`CWRMF`	1	2	200	2	0.168	0.080
	`cSLIM`	1	5	0.1	-	0.429	0.224		`cSLIM`	2	5	0.1	-	0.426	0.220
	`rcSLIM`	5	10	5	0.5	0.421	0.218		`rcSLIM`	5	10	5	0.1	0.422	0.217
	`fSLIM`	2	0.1	-	-	0.287	0.139		`fSLIM`	2	0.1	-	-	0.292	0.141
	`f2SLIM`	5	0.1	-	-	0.410	0.211		`f2SLIM`	5	0.1	-	-	0.413	0.212
F_{tfidf_fs}	`itemSI`	200	0.5	-	-	0.416	0.218	$F_{tfidf\_fs\_b}$	`itemSI`	200	0.5	-	-	0.416	0.218
	`CWRMF`	1	2	200	2	0.174	0.082		`CWRMF`	1	2	200	2	0.174	0.082
	`cSLIM`	2	5	0.1	-	0.425	0.220		`cSLIM`	1	5	0.1	-	0.429	0.223
	`rcSLIM`	5	10	5	0.1	0.422	0.217		`rcSLIM`	5	10	5	0.1	0.425	0.220
	`fSLIM`	2	0.1	-	-	0.292	0.141		`fSLIM`	2	0.1	-	-	0.287	0.139
	`f2SLIM`	5	0.1	-	-	0.413	0.212		`f2SLIM`	5	0.1	-	-	0.410	0.211

The description of each row and column is available in Section 5.4.1

Table 5.9: Methods without Side Information Performance on Lib

feature	method	params				HR	ARHR
	itemkNN	10	-	-	-	0.385	0.247
F_{no}	WRMF	0.5	200	10	-	0.361	0.221
	SLIM	5	0.5	-	-	0.407	0.266

The description of each row and column is available in Section 5.4.1

Table 5.10: Methods with Side Information Performance on Lib

feature	method	params				HR	ARHR	feature	method	params				HR	ARHR
F_b	itemSI	300	0.5	-	-	0.407	0.258	F_{tfidf}	itemSI	300	0.5	-	-	0.407	0.258
	CWRMF	1	2	200	2	0.278	0.162		CWRMF	1	2	200	2	0.278	0.162
	cSLIM	1	5	0.1	-	0.443	0.283		cSLIM	2	5	0.1	-	0.436	0.280
	rcSLIM	5	10	2	0.5	0.439	0.288		rcSLIM	5	10	2	0.1	0.434	0.286
	fSLIM	2	0.1	-	-	0.358	0.233		fSLIM	2	0.1	-	-	0.367	0.236
	f2SLIM	5	0.5	-	-	0.408	0.267		f2SLIM	5	0.5	-	-	0.410	0.266
F_{tfidf_fs}	itemSI	300	0.5	-	-	0.408	0.257	$F_{tfidf\_fs\_b}$	itemSI	300	0.5	-	-	0.407	0.257
	CWRMF	1	2	200	2	0.286	0.167		CWRMF	1	2	200	2	0.286	0.167
	cSLIM	2	5	0.1	-	0.436	0.280		cSLIM	1	5	0.1	-	0.446	0.284
	rcSLIM	5	10	1	0.5	0.435	0.286		rcSLIM	5	10	2	0.1	0.441	0.287
	fSLIM	2	0.1	-	-	0.367	0.236		fSLIM	2	0.1	-	-	0.358	0.234
	f2SLIM	5	0.5	-	-	0.410	0.267		f2SLIM	5	0.5	-	-	0.407	0.267

The description of each row and column is available in Section 5.4.1

Table 5.11: Methods without Side Information Performance on BBY

feature	method	params				HR	ARHR
	itemkNN	200	-	-	-	0.070	0.032
F_{no}	WRMF	0.5	100	10	-	0.124	0.062
	SLIM	10	0.1	-	-	0.119	0.064

The description of each row and column is available in Section 5.4.1

Table 5.12: Methods with Side Information Performance on BBY

feature	method	params				HR	ARHR	feature	method	params				HR	ARHR
F_b	itemSI	100	2	-	-	0.151	0.068	F_{tfidf}	itemSI	100	2	-	-	0.163	0.072
	CWRMF	1	0.5	50	2	0.095	0.041		CWRMF	1	0.5	100	1	0.095	0.041
	cSLIM	0.1	20	0.1	-	0.165	0.081		cSLIM	2	50	0.1	-	0.169	0.087
	rcSLIM	5	10	10	0.5	0.139	0.079		rcSLIM	5	10	10	0.1	0.152	0.081
	fSLIM	10	0.1	-	-	0.107	0.055		fSLIM	2	0.1	-	-	0.115	0.059
	f2SLIM	10	0.1	-	-	0.127	0.068		f2SLIM	20	0.1	-	-	0.132	0.069
F_{tfidf_fs}	itemSI	200	2	-	-	0.159	0.069	$F_{tfidf\_fs\_b}$	itemSI	100	2	-	-	0.153	0.068
	CWRMF	1	2	100	1	0.102	0.044		CWRMF	1	2	100	1	0.102	0.044
	cSLIM	2	50	0.1	-	0.169	0.087		cSLIM	0.1	50	0.1	-	0.180	0.090
	rcSLIM	5	10	10	0.1	0.152	0.081		rcSLIM	5	10	10	0.1	0.154	0.085
	fSLIM	2	0.1	-	-	0.115	0.059		fSLIM	20	0.1	-	-	0.109	0.057
	f2SLIM	20	0.1	-	-	0.132	0.069		f2SLIM	10	0.1	-	-	0.127	0.069

The description of each row and column is available in Section 5.4.1

Table 5.13: Methods without Side Information Performance on Yelp

feature	method	params				HR	ARHR
	itemkNN	100	-	-	-	0.257	0.092
F_{no}	WRMF	5	50	10	-	0.327	0.131
	SLIM	20	0.1	-	-	0.348	0.147

The description of each row and column is available in Section 5.4.1

Table 5.14: Methods with Side Information Performance on Yelp

feature	method	params				HR	ARHR	feature	method	params				HR	ARHR
F_b	itemSI	300	1	-	-	0.321	0.127	F_{tfidf}	itemSI	200	1	-	-	0.318	0.127
	CWRMF	1	0.5	50	2	0.244	0.100		CWRMF	1	0.5	50	2	0.244	0.100
	cSLIM	0.1	20	0.5	-	0.361	0.154		cSLIM	2	50	0.1	-	0.351	0.147
	rcSLIM	2	10	10	0.5	0.338	0.144		rcSLIM	5	10	10	0.1	0.338	0.143
	fSLIM	10	0.1	-	-	0.319	0.137		fSLIM	10	0.1	-	-	0.353	0.148
	f2SLIM	10	0.5	-	-	0.332	0.138		f2SLIM	20	0.1	-	-	0.354	0.148
F_{tfidf_fs}	itemSI	300	1	-	-	0.308	0.120	$F_{tfidf\_fs\_b}$	itemSI	300	1	-	-	0.311	0.120
	CWRMF	1	2	50	2	0.260	0.103		CWRMF	1	2	50	2	0.260	0.103
	cSLIM	2	50	0.1	-	0.351	0.147		cSLIM	0.5	20	0.1	-	0.360	0.156
	rcSLIM	5	10	10	0.1	0.338	0.143		rcSLIM	2	10	10	0.1	0.340	0.144
	fSLIM	10	0.1	-	-	0.353	0.148		fSLIM	20	0.1	-	-	0.338	0.141
	f2SLIM	20	0.1	-	-	0.354	0.148		f2SLIM	20	0.1	-	-	0.337	0.139

The description of each row and column is available in Section 5.4.1

Table 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13 and 5.14 present the detailed results of the SSLIM methods (cSLIM, rcSLIM, fSLIM and f2SLIM), the three methods without side information (itemkNN, WRMF and SLIM) and another two methods that utilize side information (itemSI and CWRMF), with respect to different side information representation schemes (F_b, F_{tfidf}, F_{tfidf_fs} and $F_{tfidf\_fs\_b}$). For the methods itemkNN, WRMF and SLIM, F_{no} is used in Table 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13 and 5.14 to denote that side information is not used.

Descriptions of the tables In Tables 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13 and 5.14, the parameters represented for each method are as follows, respectively: `itemkNN` and `itemSI`: the number of neighbors k, and weighting parameter α; `SLIM`: the ℓ_2-norm regularization parameter β and the ℓ_1-norm regularization parameter λ as in [61]; `cSLIM` and `rcSLIM`: the weighting parameter on side information regularizer α, the ℓ_2-norm regularization parameter β (β_1 and β_2) and the ℓ_1-norm regularization parameter λ as in Equation 5.2 and 5.4; `fSLIM` and `f2SLIM`: the ℓ_2-norm regularization parameter β and the ℓ_1-norm regularization parameter λ as in Equation 5.6 and 5.8; `WRMF`: the regularization parameter β, the number of latent dimensions k and the weight on non-zero values c as in [18]; `CWRMF`: the weighting parameter on side information α, the regularization parameter λ, the number of latent dimensions k and the weight on non-zero values c as in Equation 5.10; Columns corresponding to ZR and ARHR present the hit rate and average reciprocal hit-rank, respectively. N in this table is 10.

Table 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13 and 5.14 show that `SLIM` outperforms the other methods that do not utilize side information (i.e., `itemkNN` and `WRMF`) on all the datasets except `BBY`. For the `BBY` dataset, `WRMF` performs the best. This conforms to the conclusions as in [61], and thus we use `SLIM` as the baseline to further evaluate all the methods that utilize side information.

Table 5.15: Performance Improvement over `SLIM`

feature	`itemSI`	`CWRMF`	`cSLIM`	`rcSLIM`	`fSLIM`	`f2SLIM`
F_{b}	0.973	0.674	1.095	1.048	0.818	1.008
F_{tfidf}	0.988	0.674	1.090	1.062	0.877	1.026
F_{tfidf_fs}	0.974	0.707	1.090	1.063	0.873	1.027
$F_{tfidf\_fs\_b}$	0.970	0.707	1.113	1.069	0.828	1.012
avg	0.976	0.690	1.097	1.061	0.849	1.018

Each value in the first four rows is calculated as the geometric mean of HR ratios of the corresponding method over `SLIM` over all the datasets, given the corresponding feature representation scheme used. The values in the last row is calculated as the geometric mean of HR ratios of the corresponding method over `SLIM` over all the datasets and all the feature presentation schemes.

Table 5.15 summarizes the overall performance of the different methods that utilize side information, with respect to `SLIM`. Irrespective of the feature representation scheme, `cSLIM`, `rcSLIM` and `f2SLIM` perform better than `SLIM` with average improvement 9.7%, 6.1% and 1.8%, respectively (the last row in Table 5.15). This demonstrates that side information contains useful

information, and proper incorporation of side information into the recommender systems can bring significant performance improvement.

The methods itemSI, fSLIM and CWRMF perform worse than SLIM. The itemSI method is a trivial extension of itemkNN and it does not involve any learning. fSLIM is a method that learns directly from the side information. The performance of fSLIM indicates that this method may not be able to pick out and highly weight the individual features in the item side information that are most relevant to the recommendations. CWRMF is the worst one and even worse than itemSI. This may be related to the discussion on CMF as in Agarwal *et al* [62], that is, when the side information is sparse, CMF may not work well.

Comparing the gains that can be obtained by utilizing side information across the different datasets, we see that they are not uniform. For the two movie datasets (ML100K and NF), the side information provides minimal benefits, whereas the gains achieved from the other datasets is substantial. We believe that this is due to the fact that the side information used for the movie datasets was quite generic, and does not contain sufficient information.

5.4.2 Side Information Representation

Table 5.15 shows that the performance of the side information representation schemes depends on the recommendation methods. For cSLIM, which uses side information for regularization, the binary feature representations (F_b and $F_{tfidf\_fs\_b}$) lead to better performance than the multivariate feature representations (F_{tfidf} and F_{tfidf_fs}). This may be due to the fact that the binary features are treated homogeneously as the user-item purchase data and thus they can regularize the learning process effectively. However, for the methods fSLIM and f2SLIM, which involve direct learning from side information, F_{tfidf} and F_{tfidf_fs} result in better performance than the binary ones, since they differentiate the importance of features within the representations. In general, fSLIM and f2SLIM prefer the feature representations that encode word importance, so even the binary representation $F_{tfidf\_fs\_b}$, which has feature selection applied, also outperforms F_b, which does not differentiate features at all.

5.4.3 Recommendation on Different Top-N

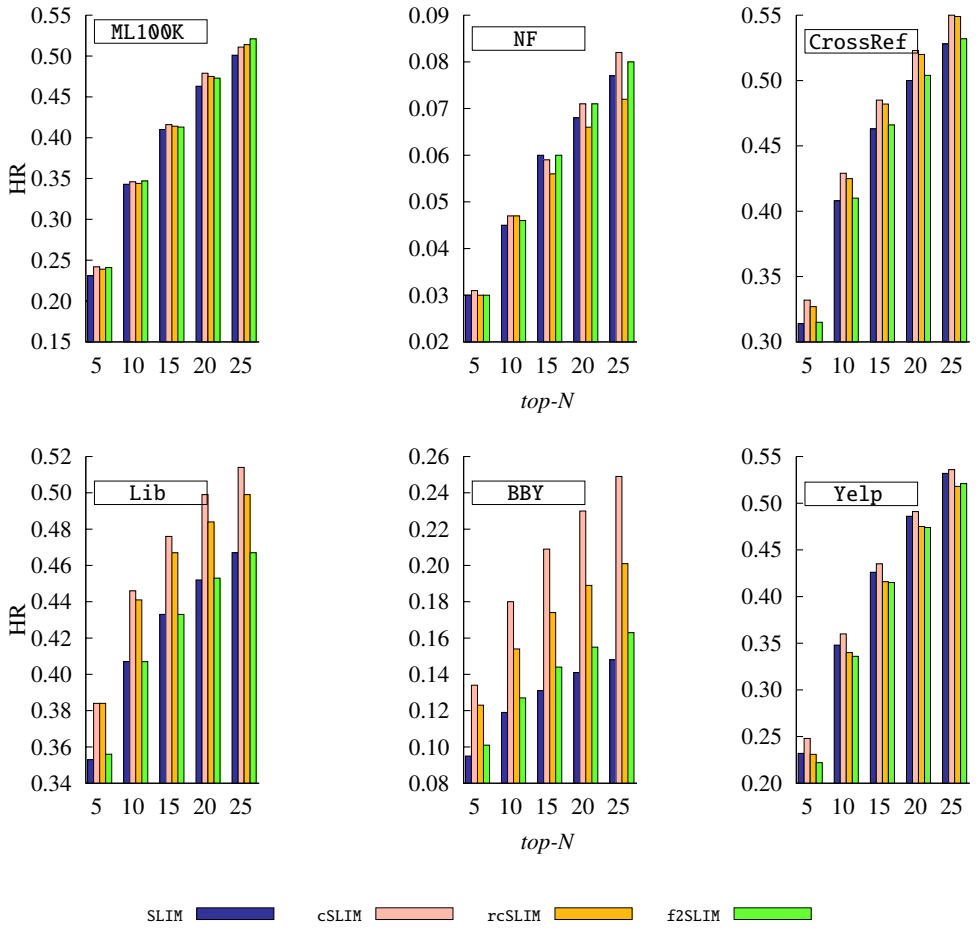


Figure 5.1: Recommendations for Different N Values

Figure 5.1 presents the performance of SLIM and SSLIM methods (except fSLIM since it performs poorly) for *top-N* recommendation with different values of N. The $\mathsf{F}_{\mathsf{tfidf\_fs\_b}}$ side information representation is used for all the methods. For all the datasets, cSLIM consistently outperforms SLIM and other SSLIM methods on all N values (except $N = 25$ for dataset ML100K and $N = 15$ for dataset NF). The rcSLIM method is the second competitive methods over all N values. The f2SLIM method shows performance that is comparable to SLIM and in some cases (i.e., $N = 25$ for ML100K, all the N values for BBY) it outperforms SLIM. It performs consistently

worse than SLIM only on Yelp.

5.4.4 Density Studies on the Purchase Data

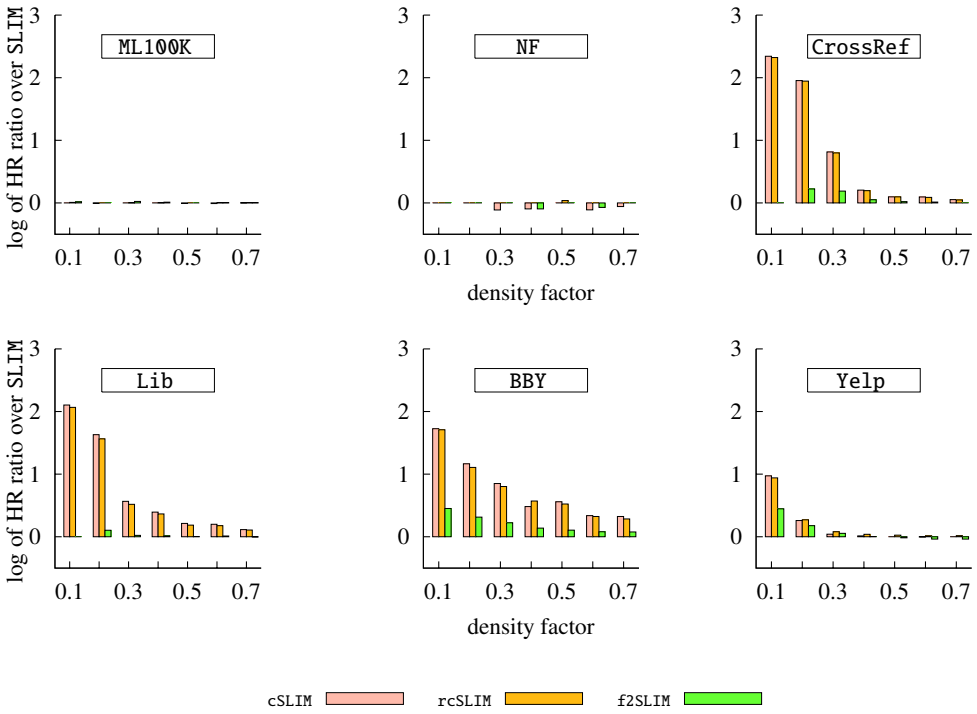


Figure 5.2: Density Studies

To understand how the density of the dataset impacts the gains that can be obtained by utilizing side information, we performed a series of experiments in which we removed some of the user-item purchase profile data as follows. For each dataset, we always keep the testing set and side information unchanged, but randomly select a certain percentage (defined as density factor) of non-zero values from each user so as to construct training sets of different information density. Figure 5.2 presents the results from different recommendation methods on the datasets. The results in these figures are relative to the performance achieved by SLIM at the same density level.

These results show that in general, cSLIM and rcSLIM lead to more significant performance gains when the user-item purchase profiles are sparser. This indicates that when the user-item

purchase data is sparse, cSLIM and rcSLIM are more effective in exploiting and incorporating side information and lead to more accurate *top-N* recommendations. Note that for the dataset ML100K and NF, performance improvement from incorporating side information is hardly observed. As discussed in Section 5.4.1, we believe that this is due to the low quality of the side information.

5.5 Conclusions

This chapter focused on incorporating side information into the sparse linear methods (SLIM) for *top-N* recommender systems. We developed four different approaches that incorporate side information during the estimation of SLIM's aggregation coefficient matrix. Our experiments showed that the developed methods lead to measurable improvements over the original SLIM methods that relied solely on user-item purchase profiles.

Chapter 6

Multi-task Learning for Recommender Systems

This chapter focuses on exploring personalized multi-task learning approaches for collaborative filtering towards the goal of improving the prediction performance of rating prediction systems. These methods first specifically identify a set of users that are closely related to the user under consideration (i.e., *active* user), and then learn multiple rating prediction models simultaneously, one for the active user and one for each of the related users. Such learning for multiple models (tasks) in parallel is implemented by representing all learning instances (users and items) using a coupled user-item representation, and within error-insensitive Support Vector Regression (ε-SVR) framework applying multi-task kernel tricks. A comprehensive set of experiments shows that multi-task learning approaches lead to significant performance improvement over conventional alternatives.

6.1 Introduction

In this chapter, we present multi-task learning methods for collaborative filtering. The methods first identify a set of related users for the active user based on his/her rating patterns, and then utilize rating information from all the related users together to build a multi-task regression model for predicting the active user's ratings on unseen items. In this fashion, the model for the active user is not only learned from his/her own preferences and tastes, but also from the knowledge that is transfered from other *related* users. The novelty of our methods is that by using the

rating prediction models of a set of related users as the parallel learning tasks, they construct multi-task learning models that are explicitly personalized to each individual user. This significantly improves the performance that can be achieved via transfer learning approaches, as the information that is being learned is now transferred across related learning tasks. The multi-task learning methods induce a high level of model interpretability and it is able to provide explicit information for further analysis on user behavior patterns.

6.2 Multi-Task Learning for Collaborative Filtering

A characteristic of existing multi-task learning formulations for collaborative filtering is that they treat the problem of learning the CF-based rating prediction models for the individual users as different learning tasks and they build a *single* multi-task prediction model that combines all these learning tasks. A direct consequence of that is that from the point of view of a specific user (u_i), the resulting multi-task model will be influenced both from the learning tasks that correspond to users that exhibit similar rating patterns with u_i (i.e., *related* tasks) and from the learning tasks that correspond to users that exhibit highly dissimilar and uncorrelated rating patterns with u_i (i.e., *unrelated* tasks). This goes against the transfer learning principle that underlies multi-task learning approaches ([35]), which are designed to leverage information that can be concurrently learned from a set of related tasks.

Motivated by this observation, the multi-task learning approach that we developed for solving the CF-based rating prediction problem does not build a single model for all the users, but instead builds a multi-task model that is explicitly designed for each user. Specifically, for each active user u_*, our approach first identifies m users, $\mathcal{N}_* = \{u_{*_1}, \ldots, u_{*_m}\}$, that are related to u_*, treats their corresponding rating prediction problems as the related learning tasks, and uses them to build a multi-task learning-based rating prediction model for u_* using ϵ-SVR. This model takes into account the ratings from the users in $\{u_*\} \cup \mathcal{N}_*$ and does not include any rating information from users that are unrelated to u_*. Moreover, the model is personalized for u_* since models built for different users will tend to have different sets of related users.

The multi-task model of user u_* is learned using error-insensitive Support Vector Regression (ε-SVR) ([63]). The input to the model are tuples of the form

$$((u_i, t_j), r_{i,j})$$

, where $u_i \in \{u_*\} \cup \mathcal{N}_*$, $t_j \in \mathcal{T}_i$, and $r_{i,j}$ is the rating that u_i has given to t_j. The user-item tuples (i.e., (u_i, t_j)) represent the instances on which the model is being learned and their corresponding ratings (i.e., $r_{i,j}$) represent the target values that are estimated by the learned regression function $f(\cdot)$. Once $f(\cdot)$ has been estimated, the prediction from the active user u_* on an unrated item t_* is determined as $f((u_*, t_*))$.

Following the previously developed kernel-based approaches for multi-task learning ([64]), the multi-task kernel function $\mathcal{K}_{mt}$ on training instances (i.e., user-item tuples) for support vector regression is defined as

$$\mathcal{K}_{mt}((u_i, t_j), (u_{i'}, t_{j'})) = \mathcal{K}_u(u_i, u_{i'}) \times \mathcal{K}_i(t_j, t_{j'}), \tag{6.1}$$

where $\mathcal{K}_u$ and $\mathcal{K}_i$ are kernel functions defined on the users and the items, respectively. When $\mathcal{K}_u$ and $\mathcal{K}_i$ are valid kernels (i.e., positive semi-definite), $\mathcal{K}_{mt}$ is a valid kernel as well ([8]).

Both the training instance representation and the multi-task kernel are essential in order to enable multiple task learning in parallel coupling with information transfer. Intuitively, the kernel function $\mathcal{K}_{mt}$ compares two tasks (users) $(u_i, t_j), \forall t_j \in \mathcal{T}_i$ and $(u_{i'}, t_{j'}), \forall t_{j'} \in I_{i'}$ by computing their similarity as the tensor of corresponding user similarity (i.e., $\mathcal{K}_u$) and item similarity (i.e., $\mathcal{K}_i$). In case $u_i = u_{i'}$, the kernel regression minimizes learning errors for only user u_i (i.e, $\mathcal{K}_u$ (u_i, u_i) is constant, and thus single-task learning). In case $t_j = t_{j'}$, the kernel regression transfers user taste on item t_j across user u_i and $u_{i'}$ through $\mathcal{K}_u$ (i.e, $\mathcal{K}_i$ (t_j, t_j) is constant). Note that the above two cases are integrated within one model learning process. In the rest of this section we discuss the methods that we developed for selecting the related users $\mathcal{N}_*$ for the active user u_*, and for computing the $\mathcal{K}_u$ and $\mathcal{K}_i$ functions.

6.2.1 Selection of Related Users

We developed two different approaches for selecting the users that will be used as the parallel tasks in our multi-task learning framework. Both of them take advantage of the "like-minded users" insight utilized by user-based collaborative filtering approaches, that is, to look for a set of users who rate items in a way that is consistent with the active user's rating behaviors.

Selection of Most Similar Users

The first approach, for a given user u_*, selects the m most related users $\mathcal{N}_* = \{u_{*_1}, u_{*_2}, \ldots, u_{*_m}\}$ as those users whose historical ratings on co-rated items are the most similar to user u_*. The

historical rating similarity between two users is computed using the modified version of Pearson correlation coefficient employed by user-based approaches ([65]), which is given by

$$\text{sim}_u(u_i, u_j) = p_u(u_i, u_j) \cdot \frac{\sum\limits_{t_k \in \mathcal{T}_c} (r_{i,k} - \bar{r}_{i,\cdot})(r_{j,k} - \bar{r}_{j,\cdot})}{\sqrt{\sum\limits_{t_k \in \mathcal{T}_c} (r_{i,k} - \bar{r}_{i,\cdot})^2} \sqrt{\sum\limits_{t_k \in \mathcal{T}_c} (r_{j,k} - \bar{r}_{j,\cdot})^2}}, \tag{6.2}$$

where $\mathcal{T}_c$ is the set of items that have been co-rated by users u_i and u_j (i.e, $\mathcal{T}_c = \mathcal{T}_i \cap \mathcal{T}_j$), $\bar{r}_{i,\cdot}$ and $\bar{r}_{j,\cdot}$ are the average ratings of users u_i and u_j over all their rated items $\mathcal{T}_i$ and $\mathcal{T}_j$, respectively. p_u is a penalty factor that linearly penalizes the similarity value when the number of co-rated items is smaller than a pre-defined small constant C, and p_u is defined as $p_u(u_i, u_j) = \min(|\mathcal{T}_c|, C)/C$. Note that since the Pearson correlation coefficient can be negative, we only select the users whose similarity to the active user u_* is positive. We refer to the above method as USM_{sim}.

Note that users with consistent negative similarities can also provide meaningful information from an opposite perspective. However, we did not include such users during user selection in order to avoid the complexity during model training (i.e., handling negative signs).

Selection of Most Covering Users

A limitation of USM_{sim} is that it does not take into account which items the selected users have provided ratings on. The items that are rated by the selected users but not by the active user are critical in order to generate accurate rating predictions for the active user on his/her unseen items. Note that the user similarity between users u_* and u_i is calculated on the items that are co-rated by u_* and u_i. Consider the following extreme scenario. If the active user u_* and another user u_i have rated a same set of items (i.e., $\mathcal{T}_* = \mathcal{T}_i$) and the corresponding ratings are identical, then their user similarity will be the highest, and thus by USM_{sim} user u_i is very likely to be selected. However, such a user u_i does not bring any additional information into model training, and therefore, he/she is not a good one to select.

The above example suggests that there are two aspects that we need to consider when select users. One is how similar the users are with the active user, and this aspect is the focus of USM_{sim}. The other is how informative the selected users are. We use the number of items that are rated by a certain user u_i but not by the active user to measure u_i's informativeness, and we propose the method which selects users that collectively maximize the coverage of the unrated items for u_*, in addition to the selection of users that rated in a consistent way with u_*. We

developed a rather simple greedy approach for combining these two heuristics, referred to as USM_{cvg}, as Program 2 shows. Along the user list identified by USM_{sim} (i.e., $(u_{*_1}, u_{*_2}, \ldots, u_{*_{2m}})$), this approach selects the user who have rated maximum number of items that are not rated by both of the active user and the users that are already selected by USM_{cvg} (line 9). In the user list (i.e., $(u_{*_1}, u_{*_2}, \ldots, u_{*_{2m}})$), the users are sorted in descending order based on their user similarity with the active user, In case that fewer users are selected by USM_{cvg} than required, USM_{cvg} chooses the most similar users from the rest of the user list (line 26). Note that by USM_{cvg}, we also increase the chances that the active item has already been rated by some selected users.

6.2.2 Kernel Functions for Users and Items

In order to construct a valid multi-task kernel $\mathcal{K}_{mt}$ as in Equation 6.1, a valid user kernel $\mathcal{K}_u$ and a valid item kernel $\mathcal{K}_i$ are required. For the kernel on users, we use the user-similarity function as defined in Equation 6.2 to construct $\mathcal{K}_u$. Similarly, we use an item-similarity function to construct the kernel $\mathcal{K}_i$ on items.

The item similarity between t_i and t_j is defined based on the adjusted cosine similarity that is used to determine the similarity between items in the context of item-based collaborative filtering algorithms for recommender systems ([66]). It is calculated as follows:

$$\text{sim}_i(t_i, t_j) = p_i(t_i, t_j) \cdot \frac{\sum\limits_{u_k \in \mathcal{U}_c} (r_{k,i} - \bar{r}_{k,\cdot})(r_{k,j} - \bar{r}_{k,\cdot})}{\sqrt{\sum\limits_{u_k \in \mathcal{U}_c} (r_{k,i} - \bar{r}_{k,\cdot})^2} \sqrt{\sum\limits_{u_k \in \mathcal{U}_c} (r_{k,j} - \bar{r}_{k,\cdot})^2}}, \tag{6.3}$$

where $\mathcal{U}_c$ is the set of users that have rated both items t_i and t_j, $\bar{r}_{k,\cdot}$ is the average ratings of user u_k over all the items that he/she rated (i.e, $\mathcal{T}_i$), and p_i is a penalty factor that linearly penalizes the similarity value when the number of co-rating users is smaller than a pre-defined small constant C and is given by $p_i(t_i, t_j) = \min(|\mathcal{U}_c|, C)/C$.

Note that both the modified Pearson correlation coefficient in Equation 6.2 and the adjusted cosine similarity in Equation 6.3 are computed only on the co-rated and co-rating items and users, respectively, and then penalized by a factor, they may not necessarily lead to valid kernels (i.e., positive semi-definite) to be used directly as $\mathcal{K}_u$ and $\mathcal{K}_i$, respectively. In such cases, the approach described in Saigo *et al* ([67]) is used to convert a symmetric matrix into positive semi-definite by subtracting from the diagonal of the matrix its smallest negative eigenvalue. After converting user similarity matrix and item similarity matrix into valid kernel matrix and used as $\mathcal{K}_u$ and $\mathcal{K}_i$, respectively, $\mathcal{K}_{mt}$ in Equation 6.1 is a valid kernel.

6.3 Materials

6.3.1 Datasets

Table 6.1: Dataset summary

	ML100K[1]	MR[2]	BX[3]	Wiki[4]	Music[5]
# users	943	500	3,305	150	5,000
# items	1,682	1,000	108,385	2,599	95,984
# ratings	100,000	43,865	217,736	11,559	675,279
min $\|\mathcal{T}_i\|$	20	11	20	20	20
rating range	1-5	1-5	1-10	0.5-5.0	1-5
avg rating	3.53	3.59	7.77	3.52	3.49
sparsity	93.7%	91.2%	99.9%	97.0%	99.9%

In this table, min $|\mathcal{T}_i|$ is the minimum number of ratings per user.

We evaluated the performance of our methods on multiple datasets. The first dataset is from MovieLens[1] , which contains 100,000 ratings from 943 users on 1,682 movies. Each user rates more than 20 movies. The rating scores fall in the range 1 to 5 as integers. We refer to this dataset as ML100K. The second dataset is from MovieRating[2] . It contains 43,865 ratings from 500 users on 1,000 movies. Each user rates more than 11 movies, and the rating scores fall in the range of 1 to 5 as integers. This dataset is referred to as MR. The third dataset is a subset extracted from BookCrossing[3] by selecting the users who have no less than 20 ratings. Thus, the extracted dataset contains 217,736 ratings from 3,305 users on 108,385 books. The rating scores fall in range 1 to 10 as integers. This dataset is referred to as BX. The next dataset is extracted from Wikilens[4] in which 150 users are selected with more than 20 ratings each, resulting in 2,599 items rated and 11,559 ratings in total. The rating scores range from 0.5 to 5.0 with granularity 0.5. The dataset is referred to as Wiki. The fifth dataset is extracted from Yahoo! Music user ratings of songs, provided as part of the Yahoo! Research Alliance

[1] http://www.grouplens.org/node/73
[2] http://www.cs.usyd.edu.au/~irena/movie_data.zip
[3] http://www.informatik.uni-freiburg.de/~cziegler/BX/
[4] http://www.grouplens.org/node/425

Webscope program [5] . Totally 5,000 users are sampled with each having more than 20 ratings, and there are 675,279 ratings on 95,984 items. The rating scores range from 1 to 5 as integers. This dataset is referred to as Music. For all the datasets above, higher rating represents higher preferences. Characteristics of the datasets is summarized in Table 6.1. We did not test the multi-task learning methods on very large datasets like Netflix dataset due to time constrains, but we discuss the scalability problem in Section 6.5.

6.3.2 Model Learning

We used the publicly and freely available support vector machine tool SVM^{light} ([68]) which implements an efficient soft margin optimization algorithm. We implemented support vector regression using SVM^{light} by specifying the "-z" option as "r" (i.e., support vector regression).

6.4 Experimental Results

In this section we present the experimental evaluation of the multi-task learning methods that we developed for building collaborative filtering-based recommender systems.

The experimental evaluation consists of two components in the following order. First, we evaluate the performance of the different methods for selecting related users. Second, we compare the performance achieved by our methods against those achieved by other widely used approaches for collaborative filtering-based recommender systems. The reported running times were obtained on workstations running Linux with Intel Xeon CPUs running at 2.66Ghz.

6.4.1 Evaluation of the Related User Selection Methods

Table 6.2 shows the MAE achieved by MTCF for the two related users selection schemes described in Section 6.2.1. These results show that the performance of the two related user selection schemes is very similar across the different datasets and number of related users, although USM_{cvg} is slightly better than USM_{sim}. This may because the information brought by extra users selected from USM_{cvg} has already been covered by users solely from USM_{sim}. Comparing the performance of the MTCF schemes as a function of the number of related users m, we see their performance initially improves as m increases, but it subsequently degrades as m

[5] http://research.yahoo.com/Academic_Relations

is further increased. However, for some datasets (e.g., ML100K) the performance differences are rather small. Table 6.8 shows corresponding computational complexities of different user selection methods.

Table 6.2: MAE of the methods for selecting related users.

method	ML100K			MR			BX		
	m = 10	m = 30	m = 50	m = 10	m = 30	m = 50	m = 5	m = 10	m = 30
USM_{sim}	0.714	0.712	0.714	0.715	0.709	0.710	1.177	1.175	1.175
USM_{cvg}	0.712	0.712	0.714	0.713	0.709	0.710	1.176	1.174	1.174

method	Wiki			Music		
	m = 5	m = 10	m = 30	m = 10	m = 30	m = 50
USM_{sim}	0.809	0.801	0.802	0.969	0.963	0.964
USM_{cvg}	0.808	0.802	0.802	0.967	0.960	0.961

In this table, *m* is the number of users selected.

6.4.2 Evaluation of the Training Instance Selection Methods

The recommendation performance achieved by the schemes that reduce the size of the training set by selecting a smaller subset of the ratings for each related user is shown in Tables 6.3, 6.4, 6.5, 6.6 and 6.7 for dataset ML100K, MR, BX, Wiki and Music, respectively.

Table 6.3: MAE for reducing the number of training instances: ML100K

k	USM_{sim}/ISM_{max}			USM_{sim}/ISM_{cvg}			USM_{cvg}/ISM_{max}			USM_{cvg}/ISM_{cvg}		
	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70
10	0.711	0.712	0.730	0.709	0.710	0.713	0.706	0.703	0.706	0.710	0.704	0.710
30	0.707	0.706	0.713	0.709	0.709	0.709	0.705	0.703	0.703	0.710	**0.700**	0.707
50	0.703	0.704	0.708	0.707	0.704	0.707	0.707	0.701	0.704	0.707	0.702	0.703

The description of each row and column is available in Section 6.4.2.

Table 6.4: MAE for reducing the number of training instances: MR

k	USM_{sim}/ISM_{max}			USM_{sim}/ISM_{cvg}			USM_{cvg}/ISM_{max}			USM_{cvg}/ISM_{cvg}		
	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70
5	0.713	0.723	0.728	0.712	0.714	0.715	0.712	0.724	0.728	0.712	0.711	0.715
10	0.713	0.727	0.732	0.711	0.718	0.719	0.712	0.728	0.730	**0.710**	0.719	0.718
30	0.719	0.725	0.724	0.718	0.721	0.720	0.716	0.721	0.726	0.714	0.719	0.720

The description of each row and column is available in Section 6.4.2.

Table 6.5: MAE for reducing the number of training instances: BX

k	USM_{sim}/ISM_{max}			USM_{sim}/ISM_{cvg}			USM_{cvg}/ISM_{max}			USM_{cvg}/ISM_{cvg}		
	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70
1	1.164	1.164	1.166	**1.161**	1.162	1.164	1.164	1.164	1.166	**1.161**	1.162	1.164
5	1.167	1.168	1.172	1.165	1.166	1.169	1.167	1.169	1.172	1.164	1.166	1.169
10	1.171	1.174	1.176	1.169	1.172	1.174	1.171	1.174	1.177	1.169	1.172	1.175

The description of each row and column is available in Section 6.4.2.

Table 6.6: MAE for reducing the number of training instances: Wiki

k	USM_{sim}/ISM_{max}			USM_{sim}/ISM_{cvg}			USM_{cvg}/ISM_{max}			USM_{cvg}/ISM_{cvg}		
	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70
1	0.806	0.804	0.812	0.806	0.805	0.812	0.790	**0.783**	0.785	0.788	**0.783**	0.785
5	0.802	0.804	0.809	0.801	0.803	0.809	0.790	0.785	0.791	0.786	0.784	0.790
10	0.800	0.803	0.806	0.802	0.804	0.805	0.791	0.787	0.794	0.789	0.786	0.793

The description of each row and column is available in Section 6.4.2.

Table 6.7: MAE for reducing the number of training instances: Music

k	USM_{sim}/ISM_{max}			USM_{sim}/ISM_{cvg}			USM_{cvg}/ISM_{max}			USM_{cvg}/ISM_{cvg}		
	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70	m=30	m=50	m=70
5	0.969	0.963	0.960	0.950	0.945	0.947	0.946	0.947	0.950	0.965	0.959	0.958
10	0.964	0.959	0.961	0.949	**0.943**	0.948	0.951	0.946	0.951	0.961	0.957	0.959
30	0.962	0.961	0.963	0.951	0.947	0.952	0.952	0.947	0.953	0.961	0.959	0.962

The description of each row and column is available in Section 6.4.2.

Descriptions of the tables In Tables 6.3, 6.4, 6.5, 6.6 and 6.7, m is the number of related users and k is the maximum number of ratings selected from each related user. USM_*/ISM_* is the paired user selection method and item selection method. **Bold** numbers are the best performance of each dataset.

Comparing the MAE achieved by these methods over those shown in Table 6.2, we see that by selectively reducing the size of the training set the accuracy of the resulting MTCF models actually improves (except MR).

We believe that these approaches lead to improved performance for two reasons. First, each of the related users may have provided ratings for some items that are not similar with the items rated by the active user. By removing such items and corresponding ratings from model learning, unrelated (i.e., noisy) information is eliminated, resulting in a more accurate model. Second, by restricting the set of ratings from each related user, we also reduce the total number of training instances that involve other users, increasing the importance of the active user's own ratings during the learning, and thus the model learned is more focused on the active user.

Comparing the performance of the two rating selection schemes, we see that ISM_{cvg} tends to perform better than ISM_{max} and that the best overall results are usually obtained when ISM_{cvg} is involved. Moreover, the results of Tables 6.3, 6.4, 6.5, 6.6 and 6.7 illustrate that unlike the earlier experiments in Section 6.4.1, USM_{cvg} tends to outperform USM_{sim} when the number of ratings is restricted, Also, the performance of the MTCF models can improve as the number of related users increases. This is especially true for the schemes using USM_{cvg} and ISM_{cvg}.

6.5, 6.6 and 6.7. This suggests that these methods are quite robust and

Table 6.8: Computational complexities of different related user selection methods.

metric	method	MK100K			MR			BX		
		m = 10	m = 30	m = 50	m = 10	m = 30	m = 50	m = 5	m = 10	m = 30
t	USM_{sim}	1.551	14.863	69.094	1.208	9.878	28.424	2.028	6.040	12.767
	USM_{cvg}	1.740	14.825	67.804	1.208	10.261	31.205	2.909	6.125	12.425
#sv	USM_{sim}	1188	3491	5887	1040	2984	4939	722	1133	2154
	USM_{cvg}	1247	3640	6024	1092	3067	4988	898	1204	2015

metric	method	Wiki			Music		
		m = 5	m = 10	m = 30	m = 10	m = 30	m = 50
t	USM_{sim}	0.351	1.213	8.608	5.423	42.47	125.32
	USM_{cvg}	0.372	1.322	8.799	5.679	44.47	129.54
#sv	USM_{sim}	616	1143	2823	1710	4720	7012
	USM_{cvg}	636	1193	2873	1739	4743	7026

In this table, m is number of related users. t is model learning time in second, and #sv is the number of support vectors in the regression model.

Table 6.9 shows the average learning time, as well as the average number of support vectors, for MTCF models with different number of selected ratings. Comparing these results with those of Table 6.8, we see that by reducing the number of training instances, we can achieve significant reductions in the amount of time required to train the models, and also the number of support vectors used in the resulting models.

Table 6.9: Computational complexities of different training set reduction schemes.

method	metric	ML100K			metric	MR		
		k = 10	k = 30	k = 50		k = 5	k = 10	k = 30
t	USM_{cvg}/ISM_{cvg}	0.466	1.874	5.394	USM_{cvg}/ISM_{cvg}	0.012	0.012	0.014
#sv	(m = 50)	473	979	1,335	(m = 1)	64	68	82

method	metric	BX			metric	Wiki		
		k = 1	k = 5	k = 10		k = 1	k = 5	k = 10
t	USM_{cvg}/ISM_{cvg}	0.008	0.009	0.009	USM_{cvg}/ISM_{cvg}	0.032	0.084	0.177
#sv	(m = 5)	47	56	68	(m = 10)	159	304	406

method	metric	Music		
		k = 5	k = 10	k = 30
t	USM_{sim}/ISM_{cvg}	0.038	0.052	0.155
#sv	(m = 10)	156	198	366

In this table, k is maximum number of ratings selected for each related user. m is the number of related users. t is model learning time in second. #sv is the number of support vectors.

6.4.3 Comparison with Other Methods

Table 6.10: Performance comparison with other methods

dataset	MTCF	ϵ-SVR	SVD	uKNN	iKNN
ML100K	0.700	0.703	0.746	**0.691**	0.705
MR	**0.710**	0.724	0.789	0.730	0.727
BX	**1.161**	1.164	-	1.218	1.199
Wiki	**0.783**	0.816	0.852	0.961	0.944
Music	**0.943**	0.956	-	1.053	0.979

In this table, **bold** numbers are the best performance for each dataset.

Table 6.10 compares the performance achieved by the multi-task learning schemes developed in this chapter against the performance achieved by some popular methods for building collaborative filtering-based recommender systems. Specifically, the performance of our methods is compared against four different schemes. The first, denoted by "ε-SVR", is a regression-based approach using ε-SVR, in which a model is learned for each user using *his/her own* ratings as the training instances. The second, denoted by "SVD", is the approach based on singular value decomposition ([66]), which has been shown to be closely related to multi-task learning. The third, denoted by "uKNN", is the standard user-based method described in ([65]). Finally, the fourth, denoted by "iKNN", is the standard item-based approach described in ([66]).

The results in this table show that MTCF achieves the best performance among the different schemes in four out of the five datasets. Its MAE is on the average lower by 1.9%, 9.1%, 11.2%, and 5.4% than that of the ε-SVR, SVD, uKNN, and iKNN schemes, respectively. Compared to the second best-performing scheme (ε-SVR), its actual performance gains varies across the different datasets. For some of them the gains are rather small (e.g., ML100K and BX), whereas for the rest the gains are substantial (e.g., MR, Wiki and Music).

6.5 Discussion & Conclusions

6.5.1 Comparison with Matrix Factorization

Since matrix factorization methods represent the state-of-the-art in recommender systems, due to their high accuracy and suitability to large-scale problems, we compare our methodologies and results with MF. We implemented the MF method BRISMF as described in [69], which gives a typical MF performance. Out of five datasets as described in Table 6.1, BRISMF outperforms MTCF on three datasets (ML100K: 0.682 vs 0.700, MR: 0.697 vs 0.710, BX: 1.146 vs 1.161, where the first number for each dataset is MAE by BRISMF and the second number is best MAE from MTCF as presented in Table 6.10). On the other two datasets, MTCF actually slightly outperforms BRISMF (Wiki: 0.789 vs 0.783, Music 1.002 vs 0.943). As it shows, our methods do not outperform MF significantly. However, we still compare these two categories of methods so as to illustrate their respective values.

The intuition behind MF methods is that users' tastes can be determined and expressed by some latent factors, and users make ratings on items mainly based on the item properties according to such factors. The main idea following this is to look for a factor space with low

dimensionality, which optimally describes all user in the system, and meanwhile all items can be mapped into the same space as feature vectors. Regularization is usually applied in order to avoid overfitting. One problem with such matrix factorization or low-rank approximation, as a compromise to its high accuracy, is that the model itself suffers from poor interpretability, due to the existence of a latent space with unknown structures. Therefore, in practical applications, to reason the predictions and furthermore enhance user experience within MF framework become not easy. This disadvantage of MF has been pointed out and well discussed by Koren ([70]).

Another problem of MF, which is related to the first one, is that MF is doing global optimization, and thus personalization is to a large extent ignored. Most of MF algorithms optimize a function of prediction errors over the entire system without consideration of the intrinsic difference that naturally exists between user communities. All the users within the system are treated as homogeneous through a relatively small yet same set of factors, and therefore community bias may not be effectively modeled in the user factor space, unless the latent space has a sufficient large dimensionality to necessarily cover user community variance, which in effect goes against the initial principle of MF methodology (i.e., low-rank factorization).

Compared to MF methods, neighborhood-based methods provide a higher level of tractability for user community analysis, as well as a stronger emphasis on user personalization, in compensation to its relatively low accuracy. Considering MTCF as a formal framework, it very naturally formulates user communities through the concept of "*related*" users. Another advantage of MTCF is that it is very handy to incorporate other source of information, once available, into model learning so as to improve recommendation performance. For example, item properties or product specifications can readily be used in neighborhood-based methods, but may not be easily incorporated into MF. The simple, tractable and flexible nature of neighborhood-based methods may still distinguish themselves as a good candidate for real applications of recommender systems.

6.5.2 Computational Considerations

Computational scalability is a major consideration in recommender systems. In the context of model-based approaches for recommender systems, there are two relevant aspects in terms of computational scalability. The first has to do with the amount of time required to build the models and the second with the amount of time required to compute the predictions.

Since model building is an off-line process, this aspect of the system can tolerate methods

that have relatively high computational requirements given that they exhibit reasonable scaling properties as the numbers of users and/or their ratings increase over time. The multi-task learning methods described in Section 6.2 need to learn a separate model for each user in the system through multiple steps. In our experience, the ε-SVR model learning step takes by far the largest amount of time. The amount of time required for updating the ε-SVR models will be lower because for many of them, the sets of users involved will remain the same. When new ratings are added, the regression function estimated in the earlier learning cycle can be used as the initial function to be further optimized so as to take into account the updates.

The prediction aspect of recommender systems is often a real- or a near real-time component and as such it is important to be able to compute the predictions fast. The time required for this step depends on the number of user-item tuples that make up the model's support vectors and whether or not the active item was part of the items used in training the model. If the active item was not used in the training step, then the item-to-item similarities between the active item and the items in support vectors need to be computed on the fly. Otherwise, the item-to-item similarities can be directly retrieved from the saved model.

The above discussion suggests two optimizations that can save computational costs for the MTCF system. The first is to reduce the number of models to be learned, which will lower the expense of off-line computations and space. The second is to reduce the number of items used for training.

Reducing the Models Learned

Our approach to reduce the number of models being learned is motivated by the following observation. If two users are among the m related users for each other and their sets of related users overlap significantly, then their multi-task models should be quite similar, due to overlapping training sets, and thus they can be used interchangeably to some extent.

We formalize this idea by using a maximal independent set (MIS) in the user-to-user graph formed during related user selection. Specifically, we construct an undirected user-to-user graph G in which there is an edge between u_i and u_j if

(i) $u_i \in \mathcal{N}_j$, and
(ii) $u_j \in \mathcal{N}_i$, and
(iii) $|\mathcal{N}_i \cap \mathcal{N}_j| / |\mathcal{N}_i \cup \mathcal{N}_j| \geq \theta$,

where θ is a threshold that measures the minimum overlap between the two sets of related users. Then, using a greedy algorithm, we find a maximal independent set MIS of G, and we learn multi-task models only for the users in MIS.

During prediction, if the active user u_* is part of MIS, its prediction is determined by its own multi-task model. Otherwise, its prediction is determined by other models through two ways. The first simply selects the user $u_i \in$ MIS such that $u_i \in \mathcal{N}_*$ and $|\mathcal{N}_i \cap \mathcal{N}_*|/|\mathcal{N}_i \cup \mathcal{N}_*|$ is the highest among $\mathcal{N}_*$ in MIS, and then computes the prediction using u_i's model. Note that because MIS is maximal, there has to exist such a user u_i that satisfies the above constraint. The second computes the predictions using the models of $\mathcal{N}_*$ within MIS and determines the final rating as the weighted average of these predictions. For the predictions obtained from u_i's model ($u_i \in \mathcal{N}_*$), the weight corresponds to $|\mathcal{N}_i \cap \mathcal{N}_*|/|\mathcal{N}_i \cup \mathcal{N}_*|$ (i.e., the degree of overlap among their sets of related users).

Selection of Item Ratings

We developed two methods to select ratings for the related users. By doing this, we can reduce the number of training instances, and meanwhile further exclude inconsistent rating information. In the first method, for each related user, we calculate the average item similarity between each of his/her rated items and all the rated items of the active user. Based on the average similarity, we select the top k most similar items and corresponding ratings from that related user. Note that the item similarity is calculated from the entire user space (Equation 6.3), and thus the average item similarity can be dominated by two different subsets of users. The first subset is the selected users. If an item has high average item similarity that mainly comes from the selected users (i.e., other users seldom rated the item), then this item is very specific to the selected users and thus to the multi-task model. Inclusion of such items can provide information on user-specific rating behaviors. The second subset is the other users rather than the selected ones. If an item is similar because many non-selected users rated it in a consistent way, then such items provide background preferences that are not specific to a certain user. By incorporating such items into model training, we can regularize the rating predictions so as to avoid extreme rating behaviors.

The second method, in addition to using the ratings identified as above for each related user, we also use ratings on the items that are not rated by the active user but have been rated by that related user. The motivation of this method is that it is possible the unseen items of the

active user are diverse from his/her already rated items, and thus accurate predictions on such unseen items cannot be produced simply from his/her own rated items and their most similar items from his/her related users. By incorporating related users' rated items that are not rated by the active user, we may include those items that are similar to the active user's unrated items into model learning so as to improve prediction accuracy. Note that the similarity between two items is measured using the item-similarity function defined in Equation 6.3.

Due to space constraints, we do not present the results of MIS-based algorithms. However, the conclusion is the MIS-based methods do not degrade recommendation performance but significantly lower down the running time.

6.5.3 Conclusions

In this chapter, we presented a model for collaborative filtering based on multi-task learning. This model leverages information from a set of related users and leads to more accurate predictions. The experimental results show that the multi-task model consistently gives better predictions than other existing methods.

Part II

Chemical Informatics

Chapter 7

Preliminaries for Chemical Informatics

7.1 Literature Reviews

7.1.1 Structure-Activity-Relationship (SAR) Models

Over the years, a number of methods have been developed for improving the accuracy of the SAR models that utilize additional information beyond the known ligands of the targets under consideration. One of the early methods utilizes approaches based on active learning and iteratively expands the set of training compounds used for learning the SAR models [71]. In this approach, the target's experimentally determined ligands are used to build an initial SVM-based SAR model. Compounds that are close to the decision boundary of the SVM model are then selected and treated as additional positive training examples for learning a new SVM model. This process is repeated multiple times until the performance of the learned model cannot be further improved.

Probably the most widely used approaches for improving the quality of the SAR models are those based on chemogenomics [72, 73, 74]. The key idea behind these approaches is to synergistically use information across a set of proteins that belong to the same family (e.g., GPCRs, Kinases, etc). The rationale of these approaches is that proteins belonging to the same protein family tend to bind to compounds that share certain common characteristics. Thus, by taking

into account the known ligands of all or a subset of the family members, better models can be expected. In these approaches, a model is trained using instances consisting of target-ligand pairs from protein members of the same family and their ligands. This model can then determine the SAR score for a specific target and a specific compound by using it to predict that particular target-compound pair. The different chemogenomics-based approaches that have been developed differ on the features of the targets, compounds, and their complexes that they utilize (e.g., physicochemical properties [75, 76], protein structure [77], amino acid sequence [78], binding site descriptors [79, 80], topological descriptors [75], protein-ligand fingerprints [81], etc.), how they represent target-compound pairs (e.g., concatenation of descriptor vectors [75], tensor products [78], kernel fusion [78], etc.), and the machine-learning methods that they use for learning the models (e.g., support vector machines [78, 82], neural networks [64], partial least-squares [80, 76, 77], random forests [81], multi-task learning [82], etc.).

Support Vector Machines (SVM) [8] are widely applied learning algorithms to do classification and regression (SVR) [83] in chemical informatics research. In Bock *et al*. [75], they proposed a support vector regression method to predict compound activity against orphan GPCRs. They represented targets by their physicochemical properties of their primary structures (i.e., surface tension, isoelectric point, accessible surface area, etc.). To describe compounds, they used a two-dimensional molecular connectivity matrix supplemented by chemical properties, and then the matrix is factorized using singular value decomposition (SVD) and singular values are used to represent the compounds. Target-ligand complexes are represented by concatenating target representation and compound representation and then the complexes are trained within support vector regression. Similarly in Strömbergsson *et al*. [79], they used 3D structures of proteins around binding sites to describe proteins and then also used support vector regression to train a model and predict enzyme-ligand interactions.

Partial Least-Squares (PLS) is also a popular technique for SAR model learning in the context of Chemogenomics. In Lapinsh *et al*. [76], a partial least-squares is learned to predict organic compound-amine G protein-coupled receptor interactions. In their study, compounds are represented by grid independent descriptors of their generated 3D structures. Receptor descriptions are derived from differences in the physicochemical properties of the seven cell membrane-spanning alpha-helical regions in the receptor series. Then ligand-receptor cross-terms are calculated. A partial least-squares is learned on compound descriptions, receptor descriptions and ligand-receptor cross-terms and this PLS is used to do predictions. In Deng

et al. [80], they developed a kernel partial least squares (kernel PLS) model to predict protein-ligand binding affinities, in which protein-ligand interactions are characterized as a pattern comprising occurrence counts of atom pairs within a specified cutoff range (6Å) from ligand atoms.

7.1.2 Structure-Selectivity-Relationship (SSR) Models

Developing computational methods to aid in the identification of selective compounds has recently been recognized as an important step in lead optimization and several studies have shown the promise of utilizing machine-learning approaches towards this goal. Vogt *et al*.[84] investigated approaches for identifying selective compounds based on how similar they are to known selective compounds (similarity search-based approach). They tested five widely used 2D fingerprints for compound representation and their results demonstrated that 2D fingerprints are capable of identifying compounds which have different selectivity properties against closely related target proteins. Stumpfe *et al*.[85] developed two approaches that they referred to as *single-step* and *dual-step* approaches. The single-step approach builds the SSR model by utilizing only the selective compounds (one class classification). The two-step approach uses a pair of classifiers that are applied in sequence. The first is a binary classifier trained on selective compounds (positive class) and non-selective active compounds (negative class), whereas the second classifier is the one-class classifier as used in the single-step approach. A compound is considered to be selective if both classifiers predicted it as such. For both approaches, they used both *k*-nearest-neighbor (similarity search) and Bayesian methods in building the models and represented the compounds using MACCS and Molprint2D descriptors. Their experimental results demonstrated that both of these approaches are able to identify selective compounds. Wassermann *et al*.[86, 87] built on this work and investigated the use of Support Vector Machines (`SVM`) [8] as the underlying machine learning framework for learning SSR models. Specifically, they investigated four types of SSR models. The first is a binary classifier that uses selective compounds as positive instances and inactive compounds as negative instances. The second is a set of three one-vs-rest binary classifiers whose positive classes correspond to the selective, non-selective active, and inactive compounds, respectively, and whose negative class correspond to the compounds that did not belong to the positive class. The third is a two-step approach in which the model of the first step uses active compounds as positive instances and inactive compounds as negative instances (i.e., a standard SAR model) and the model of the second step uses selective compounds as positive instances and non-selective active compounds as negative

instances. Finally, the fourth is a preference ranking model that incorporates pairwise constraints that rank the selective compounds higher than the inactive compounds and the inactive compounds higher than the non-selectives (i.e., selectives > inactives > non-selectives). Their results showed that SVM-based methods outperformed conventional similarity search methods and that the ranking and one-versus-rest methods performed similarly to each other and outperformed the other SVM-based methods.

7.1.3 Multi-Task Learning (MTL) for Chemical Informatics

Recently, multi-task learning has quickly gained attention of people from chemical informatics. In Erhan *et al*. [64], they predicted target-ligand interactions inside a family of targets by employing Collaborative Filtering (CF) [88] technology, which is a specific form of multi-task learning, by designing Neural Network (NN) [89] and a kernel-based ordinal regression method named JRank in which a set of new kernels on targets and compounds (i.e., identity kernel, Gaussian kernel, correlation kernel and quadratic kernel) are also developed. In a very recent paper [81] Weill *et al*. proposed a novel protein-ligand fingerprint and then applies Ranomd Forest(RF) [90] and Naïve Bayesian (NB) [91] to do perform interactions between GPCRs and their ligands, which in effect is also doing multi-task learning in the framework of Chemogenomics. The fingerprint is generated by concatenating the GPCR cavity descriptor and GPCR ligand descriptor. In Geppert *et al*. [82], they designed six methods to predict compound activity, including linearly combining multiple linear SVM models and multi-task SVM models. In Jacob *et al*. [78], they proposed multi-task learning methodology to predict protein-ligand interactions, and they developed multiple kernels on targets, termed as Dirac kernel, multitask kernel and hierarchy kernel. They applied tensor product on protein kernel and compound kernel as the kernel on compound-ligand complexes. In Nigsch *et al*. [92], they applied Laplacian-modified Naïve Bayesian classifier and a perceptron-like learning algorithm called Winnow [93] to perform multi-class classification to predict ligand-target interactions. They augmented the features of compounds by creating addition so-called Orthogonal Sparse Bigrams (OSBs) from original features. In Strömbergsson *et al*. [94], they used a learning method called Rough Set (RS) [95] to generate rules from GPCR-ligand interactions. GPCR-ligand complexes are represented in a table in which a set of attributes are generated to indicate if a certain GPCR-ligand complex along a row in the table satisfies certain properties. Then RS derives a set of decision rules from the table to predict if unseen compound is active against a GPCR. In Lindström *et al*. [77],

they used protein structures, ligand physical chemical properties and protein-ligand interaction descriptors such as steric and electrostatic complementarity to describe proteins and ligands, and they employed hierarchical partial-least-squares regression to latent structures to predict protein-ligand interactions.

7.2 Definitions and Notations

The notations and definitions that are used for chemical informatics in the thesis are given in Table 7.1.

Table 7.1: Definitions and Notations for Chemical Informatics

Not	Definitions	Description
p (p_i)	a target (the i-th target)	
P_i	the *challenge set* of p_i	defined as in Equation 7.3
$\mathcal{P}$	the entire set of targets under consideration	
c	a compound	
C	a set of compounds	
C_i^+	a set of experimentally determined active compounds for p_i	
C_i^-	a set of experimentally determined inactive compounds for p_i	
C_i	a set of experimentally determined compounds for p_i	$C_i = C_i^+ \bigcup C_i^-$
$\mathcal{C}$	the union of active compounds over $\mathcal{P}$	$\mathcal{C} = \bigcup C_i$
$\text{nbrs}_k(c, C)$	the k most similar compounds of c in C	referred to as c's k nearest-neighbor in C
$\mathcal{N}_k(C_x, C_y)$	the union of the k nearest-neighbors of each compound $c \in C_x$ in C_y	$\mathcal{N}_k(C_x, C_y) = \bigcup_{c \in \boldsymbol{C}_x} \text{nbrs}_k(c, C_y).$ (7.1)
$S_i^+(P_i)$	p_i's selective compounds against P_i	S_i^+ if only a single challenge set
$S_i^-(P_i)$	the nonselective active compounds of p_i against P_i	S_i^- if only a single challenge set
H_i^+	the newly labeled positive compounds for p_i	defined in Section 8.2.4

For reasons discussed later in Section 8.2.1, the set of compounds in C_i^- will be obtained by randomly sampling the compounds that do not belong in C_i.

Each compound will be represented by a topological descriptor-based representation in which each compound is modeled as a frequency vector of certain subgraphs (descriptors) present in its molecular graph [96]. The similarity between two compounds c_x and c_y will

be denoted by $\text{sim}_c(c_x, c_y)$ and will be computed as the Tanimoto coefficient of their descriptor-based representation [97]. The Tanimoto coefficient is given by:

$$\text{sim}_c(c_x, c_y) = \frac{\sum_k c_{x,k} c_{y,k}}{\sum_k c_{x,k}^2 + \sum_k c_{y,k}^2 - \sum_k c_{x,k} c_{y,k}}, \tag{7.2}$$

where k goes over all the dimensions of the descriptor space and $c_{x,k}$ is the number of times descriptor k occurs in compound c_x.

The activity of a compound will be determined by its IC_{50} value (i.e., the concentration of the compound that is required for 50% inhibition of the target under consideration, and lower IC_{50} values indicate higher activity[1]). A compound will be considered to be *active* for a given target if its IC_{50} value for that target is less than 1000 nM.

A compound c will be *selective* for p_i against a set of targets P_i if the following two conditions are satisfied:

$$\begin{aligned} &\text{(i)} \quad c \text{ is active for } p_i, \text{ and} \\ &\text{(ii)} \quad \min_{\forall t_j \in P_i} \frac{\text{IC}_{50}(c, p_j)}{\text{IC}_{50}(c, p_i)} \geq 50. \end{aligned} \tag{7.3}$$

This definition follows the common practice of using the ratio of binding affinities in determining the selectivity of compounds [98]. Note that c can be either active or inactive for some or all of the targets in P_i while being selective for p_i.

An important aspect of the selectivity definition is that it is done by taking into account both the target under consideration (p_i) and also another set of targets (P_i) against which a compound's selectivity for p_i is defined. We will refer to P_i as the *challenge set*. Depending on the problem at hand, each target may have multiple challenge sets and they will be denoted using subscripts like $P_{i,1}, P_{i,2}, \cdots, P_{i,n}$. In such cases, a compound's selectivity properties for a target can be different against different challenge sets.

Given a target p_i, the goal of the *Structure-Activity-Relationship model* (SAR) is to predict if a compound is active for p_i. Given a target p_i and a challenge set P_i, the goal of the *Structure-Selectivity-Relationship model* (SSR) is to predict if a compound is selective for p_i against all the targets in P_i. We will refer to target p_i as the *target of interest*.

[1] http://www.ncgc.nih.gov/guidance/section3.html

7.3 Chemical Compound Descriptor

The chemical compounds were represented using the topological descriptors based on graph fragments (GF) [99]. The GF descriptors correspond to all connected subgraphs up to a user-specified length that exist in a compound library. Comparisons against other popular topological descriptors (extended connectivity fingerprints, Maccs keys (MDL Information Systems Inc.), and frequent subgraph descriptors) have shown that the GF descriptors lead to a chemical compound representation that captures its structural characteristics effectively. As a result, its performance is either better than or comparable to that achieved by currently used descriptors for the tasks of building `SVM`-based SAR models and similarity searching. The GF descriptors were generated using the AFGEN [100] program and contained all the graph fragments of size four to seven bonds.

7.4 Evaluation Methodology & Metrics

The evaluation metrics used in the chemical informatics are listed in Table 7.2.

The performance of the different SAR methods was evaluated using a five-fold cross validation framework. For each target p_i, its set of positive C_i^+ and negative C_i^- compounds were split into five equal-size parts (folds). The compounds in each subset of four folds were used to train a model, which was then used to predict the compounds of the left-out fold.

Note that in the case of the approaches based on semi-supervised learning, some of the newly labeled positive compounds (i.e., H_i^+) may already be in the test set. In such cases, we remove from H_i^+ all these compounds. This allows us to use exactly the same test sets for all the different schemes. This step tends to remove around 2.5% of the newly labeled compounds are removed from H_i^+.

The quality of the SAR models was measured using the ROC score [101] as defined in Table 7.2. Since a five-fold cross validation was used, the computed ROC scores correspond to the average of the five folds.

The performance of the different SSR methods are evaluated via a five-fold cross validation, in which the corresponding active compounds and inactive compounds of each target are randomly split into 5 folds, four folds for model learning and the rest fold for testing, and each of these folds is enforced to have about the same number of selectively active compounds.

The quality of the SSR models is measured using F_1 as defined in Table 7.2.

Table 7.2: Evaluation Metrics for Chemical Informatics

Name	Definitions	Description
ROC	The normalized area under the curve that plots the true positives against the false positives for different thresholds for classification (receiver operating characteristic curve).	ROC $\in [0, 1]$, ROC=0.5 corresponds to random performance. Higher ROC corresponds to better performance.
precision	$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$ (7.4)	The fraction of correctly classified positive instances (i.e., true positive) over all testing instances that are classified as positive (i.e, true positive and false positive).
recall	$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$ (7.5)	the fraction of correctly classified positive instances (i.e., true positive) over all positive testing instances (i.e., true positive and false negative).
F_1	$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$ (7.6)	the harmonic mean of precision and recall. Intuitively, if precision and recall are both high, or one of the two is very high, F_1 measure will be high, and thus higher F_1 values indicate better performance.
F_1^b	best F_1 values over a set of threshold values α as defined in Section 7.4.	

Conventionally, in `NN` settings, if prediction scores are above 0.5 (in case of 0/1 binary labels), it is considered as positive prediction, and thus 0.5 by default serves as a threshold (referred to as α) to determine if a prediction is positive or not, and precision and recall values are calculated based on the threshold. However, in some cases a different threshold α may be preferred so as to favor or disfavor predictions above or below a certain value. In our study, we evaluated threshold values and calculated precision and recall, and F_1 measure corresponding to each of the thresholds α, and search for the best parameter α which gives best F_1 measure. We refer to this best F_1 values as F_1^b. During the experimental evaluation, we report the average

F_1^b values across 5 folds.

Chapter 8

Multi-Assay-Based Structure-Activity-Relationship Models

Structure-activity-relationship (SAR) models are used to inform and guide the iterative optimization of chemical leads, and play a fundamental role in modern drug discovery. In this chapter we present a new class of methods for building SAR models, referred to as *multi-assay-based*, that utilize activity information from different targets. These methods first identify a set of targets that are *related* to the target under consideration and then they employ various machine-learning techniques that utilize activity information from these targets in order to build the desired SAR model. We developed different methods for identifying the set of related targets, which take into account the primary sequence of the targets or the structure of their ligands, and we also developed different machine learning techniques that were derived by using principles of semi-supervised learning, multi-task learning, and classifier ensembles. The comprehensive evaluation of these methods shows that they lead to considerable improvements over the standard SAR models that are based only on the ligands of the target under consideration. On a set of 117 protein targets obtained from PubChem, these multi-assay-based methods achieve an ROC score that is on the average 7.0%–7.2% higher than that achieved by the standard SAR models. Moreover, on a set of targets belonging to six protein families, the multi-assay-based methods outperform chemogenomics-based approaches by 4.33%.

8.1 Introduction

The pioneering work of Hansch *et al*. [102, 103], which demonstrated that the biological activity of a chemical compound can be mathematically expressed as a function of its physicochemical properties, led to the development of quantitative methods for modeling structure-activity relationships (SAR). Since that work, many different approaches have been developed for building such structure-activity-relationship (SAR) models [96, 104]. These *in silico* models have become an essential tool for predicting the biological activity of a compound from its molecular structure and played a critical role in drug and chemical probe discovery by informing the initial screens, design, and optimization of chemical compounds with the desired biological properties.

In this chapter we present a different approach for improving the quality of SAR models that also utilizes activity information from other protein targets. This approach, referred to as *multi-assay based*, identifies a set of targets that are *related* to the target under consideration and then utilizes only activity information from these targets while learning the desired SAR model. Even though this approach shares some characteristics with those based on chemogenomics, its advantage is that, by using appropriate target-to-target similarity functions to identify the related targets, it can adapt to the characteristics of the protein target under consideration and lead to higher-quality SAR models. In addition, its adaptive nature allows it to select a smaller number of targets than those present in the entire family, or to select targets from different families if their use will lead to better quality models.

We developed and investigated different methods to identify the set of related targets and to incorporate their activity information into the multi-assay-based SAR model. Specifically, we developed different target-to-target similarity measures for identifying the set of related targets that take into account the primary structure of the targets themselves or the structure of their ligands. In addition, we developed three different machine-learning approaches for building the SAR models that were derived from the principles of semi-supervised learning [105], multi-task learning [106, 37, 38, 36], and classifier ensembles [107, 108, 109]. The experimental evaluation of these methods on a set of 117 targets extracted from PubChem show that for nearly all of them, the incorporation of activity information from other targets leads to quality improvements in the resulting multi-assay-based SAR models. The best results are obtained for the ligand-based target-to-target similarity methods and the multi-task learning and classifier ensembles schemes, which achieve an average of 7.0%–7.2% ROC improvement. In addition,

on a set of six protein families, the multi-assay-based methods achieve a 4.3% improvement over chemogenomics-based approaches.

8.2 Methods

8.2.1 Baseline SAR Models

For each target p_i, we used support vector machines (SVM) [8] to build the baseline SAR model that relies only on its own activity information. SVM has been introduced in Section 2.4.2. Alternatively, SVM learning can be also interpreted as follows. Given a set of positive training instances $\mathcal{I}^+$ and a set of negative training instances $\mathcal{I}^-$, it learns a classification function $f(x)$ of the form

$$f(x) = \sum_{x_i \in \mathcal{I}^+} \lambda_i^+ \mathcal{K}(x, x_i) - \sum_{x_i \in \mathcal{I}^-} \lambda_i^- \mathcal{K}(x, x_i) \tag{8.1}$$

where λ_i^+ and λ_i^- are non-negative weights that are computed during training by maximizing a quadratic objective function, and $\mathcal{K}(.,.)$ is a *kernel* function that measures the similarity between the compounds. Given a new instance x, the sign of the prediction score $f(x)$ is used to determine the class of x. In addition, a set of compounds can be ranked based on their likelihood of being positive by sorting their prediction scores in non-increasing order.

In the context of our problems, the set of positive instances for p_i corresponds to its own set of experimentally determined ligands C_i^+. However, determining the set of compounds that will form the negative class is problematic for two reasons. First, in many target-ligand activity databases, only information about actual ligands of a particular target is available and information about non-binding compounds is not provided. Second, even when the activity information is obtained from screening assays, the negative information may not be very reliable as compounds can fail to bind to a particular target due to assay-related artifacts. Thus, the actual learning problem associated with building a SAR model is that of learning a classifier from only positive and unlabeled instances [110, 111, 112, 113] (an instance is considered to be unlabeled if it is not positively labeled). An approach that has been successfully used in the past to address this problem is to select as negative instances a random subset of the unlabeled compounds [104]. Recent work has shown that under the assumption that the labeled instances are *selected completely at random*, the model learned from such randomly selected negative instances produces rankings that are equivalent to the real model [113].

In this work, motivated by these empirical and theoretical results, the set of negative instances (C_i^-) for the baseline SAR model is obtained by selecting $|C_i^+|$ random compounds from $C \setminus C_i^+$. This allows for the creation of equal-size positive and negative training sets. Moreover, by using $C \setminus C_i^+$ as the pool of compounds from which to select the negative instances, it allows for the definition of a more realistic (and harder) negative class as it contains compounds that are known to bind to other protein targets. Note that the same C_i^- set is also used for defining the negative instances for all the multi-assay-based methods that are described in the subsequent sections.

8.2.2 Multi-Assay-based SAR Models

In recent years, chemogenomics-based approaches have illustrated that the quality of the SAR models can be improved by taking into account the activity information of proteins in the same family. However, the fundamental step in these approaches, which is building a SAR model based on all or a subset of the proteins in the family, has a number of shortcomings. First, it can only be applied to protein families for which activity information is available for multiple members. Second, for a specific target p_i, the chemogenomics-based model may contain activity information from protein targets that may not be helpful for it (e.g., targets that bind to substantially different sets of ligands). This can easily happen for protein families that contain a diverse set of proteins. The inclusion in the model of these less-relevant proteins can negatively impact the quality of the model learned for p_i. For example, in the case of the `SVM`-based approaches, the decision hyperplane may be unnecessarily minimizing the errors that are associated with the targets that are not relevant for p_i, whereas at the same time increasing the errors associated with p_i itself or other relevant targets. Third, for the cases in which a specific target p_i shares key characteristics related to ligand binding and recognition with proteins of other families, the intra-family focus of the chemogenomics-based approaches fails to take advantage of the relevant activity information provided by proteins in other families, leading to lower quality SAR models.

The multi-assay-based approaches that are developed in this chapter are designed to overcome all three of the above shortcomings. For each specific target p_i, these approaches identify a set of protein targets that are related to p_i and then utilize only the activity information from these targets while learning p_i's SAR model. In addition, by using appropriate target-to-target similarity functions, these approaches can adapt to the characteristics of the individual protein

targets and allow them to potentially select a subset of the proteins in p_i's family or proteins across different families. Finally, since these approaches do not rely on protein family membership, they can be used for proteins for which there is no activity information for any other family member.

The subsequent sections describe the different target-to-target similarity measures that we developed for identifying the set of related proteins and the different machine-learning methods that we developed for improving the quality of the target-specific SAR model by utilizing activity information from its related targets.

8.2.3 Identifying Related Targets

We developed two classes of target-to-target similarity functions that capture the similarity between the targets by taking into account two different types of information. The first takes into account the amino acid sequence of the targets whereas the second takes into account the similarity between their ligands.

Sequence-based Methods

Protein targets that have similar ligand binding sites in terms of their amino acid composition and their 3D structure show similar binding affinity towards a similar set of compounds [114]. Thus, a natural way of comparing two targets is to compare the sequences and structures of their binding sites. However, in many case the 3D structure of the proteins under consideration is not known (e.g., GPCRs), making it hard to accurately and reliably compare the ligand binding sites for all proteins. For this reason, we developed a target-to-target similarity function, referred to as $\mathcal{K}_p^{Seq}$, that measures the similarity between two protein targets by taking into account their entire amino acid sequences. Specifically, $\mathcal{K}_p^{Seq}(p_i, p_j)$ is computed as the optimal local alignment score [115] between p_i's and p_j's PSI-BLAST derived sequence profiles [116] and the PICASSO [117] profile-based scoring scheme. This profile-based alignment method combined with the PICASSO scoring scheme has been shown to better capture the evolutionary conserved sequence conservation signals between the proteins [118, 117].

Ligand-based Methods

The similarity between two targets can also be indirectly determined by considering their ligands. If two targets p_i and p_j have similar sets of ligands, then most likely their corresponding ligand binding sites share certain common characteristics. As a result, the similarity between their sets of ligands can be an implicit measure of the similarity of their binding sites. Motivated by this, we developed two approaches for determining the target-to-target similarity that take into account the similarity between their ligands. The first, referred to as $\mathcal{K}_p^{Aligs}$, measures the pairwise similarity of two targets p_i and p_j as the average pairwise similarity between their ligands. That is,

$$\mathcal{K}_p^{Aligs}(p_i, p_j) = \frac{\sum_{c_x \in C_i^+} \sum_{c_y \in C_j} \text{sim}_c(c_x, c_y)}{|C_i^+||C_j|}. \tag{8.2}$$

The second, referred to as $\mathcal{K}_p^{kligs}$, measures the pairwise similarity of two targets p_i and p_j by considering only the average pairwise similarity of the k-nearest neighbors of each ligand to the other target's ligands. Specifically, $\mathcal{K}_p^{kligs}(p_i, p_j)$ is given by

$$\begin{aligned} \mathcal{K}_p^{kligs}(p_i, p_j) = \; & \frac{1}{k|C_i^+|} \sum_{c_x \in C_i^+} \sum_{c_y \in \text{nbrs}_k(c_x, C_j)} \text{sim}_c(c_x, c_y) \\ & + \\ & \frac{1}{k|C_j|} \sum_{c_x \in C_j} \sum_{c_y \in \text{nbrs}_k(c_x, C_i^+)} \text{sim}_c(c_x, c_y). \end{aligned} \tag{8.3}$$

The design of $\mathcal{K}_p^{kligs}$ was motivated by the fact that targets may contain ligands that come from multiple (and potentially different) scaffolds. As a result, the $\mathcal{K}_p^{Aligs}$ function, will unnecessarily penalize a pair of protein targets, each containing ligands derived from different scaffolds, even when the sets of scaffolds in each target are similar.

8.2.4 Affinity-based SAR Models using Semi-Supervised Learning

The first method that we developed for building a multi-assay-based SAR model for a specific target utilizes approaches based on semi-supervised learning [105]. The main idea of semi-supervised learning methods is to take advantage of the unlabeled instances during training in order to modify or re-prioritize the hypotheses obtained from the labeled instances [119]. This is usually done using a two-step process. In the first step, labels are assigned to the unlabeled

instances. In the second step, a model is learned using both the original and the newly labeled instances.

Within the context of learning a multi-assay-based SAR model for a specific target p_i, the semi-supervised learning approach that we developed considers as unlabeled only those compounds that are ligands to at least one of the related proteins and are neither positive nor negative instances for p_i. Specifically, if $R_i = \{p_{i_1}, p_{i_2}, \ldots, p_{i_m}\}$ are the m most similar target of p_i in $\mathcal{P}$ in non-increasing similarity order (i.e., the m related targets of p_i), then the set of compounds that are considered to be unlabeled is

$$U_i = \left(\bigcup_{1 \leq j \leq m} C_{i_j}^+ \right) \setminus (C_i^+ \cup C_i^-).$$

The motivation behind this definition is that the compounds in U_i corresponds to a biologically relevant subset of the chemical space as it contains compounds that have been experimentally determined to bind to a set of protein targets that are similar to the specific target p_i.

Details on how the labels are assigned to the compounds in U_i and how they are used to build better SAR models are provided in the next two sections.

Methods for Labeling Unlabeled Compounds

We developed two methods for labeling the compounds in U_i. The first method is based on a simple k-nearest neighbor scheme, whereas the second method employs an approach based on label propagation [120] that is used extensively for labeling unlabeled instances in semi-supervised learning.

In the k-nearest-neighbor-based method, referred to as $\text{LS}_{k\text{nn}}$, the compounds in U_i that belong in the k-nearest-neighbor list of at least one compound in C_i^+ (i.e., $\mathcal{N}_k(C_i^+, U_i)$) are labeled as positives and the remaining compounds are labeled as negatives. This is motivated by the fact that compounds that are structurally similar tend to share the same biological activity [121]. As a result, those compounds in U_i that are similar to p_i's own ligands have a high probability of being active for p_i (i.e., be positive), whereas compounds that are dissimilar to p_i's ligands have a high probability of being inactive (i.e., be negative). Note that $\text{LS}_{k\text{nn}}$, is similar in spirit to the cluster kernel [122], which assumes that unlabeled data within the neighborhood of the labeled data should be used with the same labels.

In the label propagation-based method, referred to as LS_{LP}, the labels of the compounds in U_i are determined by first constructing a weighted k-nearest neighbor compound-to-compound

similarity graph involving both labeled and unlabeled compounds and then using an iterative procedure to propagate the labels from the labeled to the unlabeled nodes in this graph. Specifically, the graph contains a positively labeled node for each compound in C_i^+, a negatively labeled node for each compound in C_i^-, and an unlabeled node for each compound in U_i.

The pseudo-code for the label propagation algorithm is shown in Algorithm 1. With $n = |C_i^+ \cup C_i^- \cup U_i|$, T is a $n \times n$ transition matrix, L is a $n \times 2$ label matrix, and $w_{p,q}$ is the weight assigned to the edge (p, q) that corresponds to the similarity between compounds p and q. The algorithm initially starts by computing the transition matrix (lines 1–6), initializing the labels of the nodes corresponding to the compounds in C_i^+ and C_i^- (lines 7–15), and assigning a weight of 0.5 to the labels for the rest of the nodes (lines 16–18). Then it proceeds to iteratively propagate the labels (lines 19–22) until convergence [120]. Finally, the labels of the nodes in U_i are determined as the maximum weight label (lines 23–31).

Building SAR Models Using the Newly Labeled Compounds

The labeling methods described in the previous section will assign a label (either positive or negative) to all the compounds in U_i. However, since the nature of the models that we learn rely only on positively labeled instances (the negative instances are obtained by randomly sampling the unlabeled instances), we use only the positive subset of the newly labeled instances, denoted by H_i^+, as additional labeled instances to learn a SAR model for target p_i.

Specifically, we developed two different approaches for incorporating the newly labeled compounds into p_i's SAR model. The first approach, treats the original (C_i^+) and the newly labeled (H_i^+) positive instances equally, whereas the second approach, controls the influence that H_i^+'s compounds can have on the final model, by assigning a different misclassification cost to the compounds in C_i^+ and H_i^+. This differential weighting is done by using a variable α ($0 \leq \alpha \leq 1$) that controls the relative importance of C_i^+'s compounds over those in H_i^+ and then assigning a weight w_k to each compound c_k such that

$$w_k = \begin{cases} \alpha\left(1 + \frac{|H_i^+|}{|C_i^+|}\right) & \text{if } c_k \in C_i^+, \\ (1-\alpha)\left(1 + \frac{|C_i^+|}{|H_i^+|}\right) & \text{if } c_k \in H_i^+. \end{cases} \tag{8.4}$$

As a result, the compounds in C_i^+ will account for $\alpha\%$ of the overall weight of the positive instances, whereas the compounds in H_i^+ will account for the rest. Note that when $\alpha =$

$|C_i^+|/|C_i^+ \cup H_i^+|$, this approach assigns a weight of one to all the compounds in C_i^+ and H_i^+, at which point it becomes identical to the first approach. We will denote these two approaches as CWS_{none} and CWS_α, respectively.

In addition, we also extended the CWS_α weighting scheme to take into account the similarity between p_i and its m related targets. The motivation behind this weighting scheme is to increase the importance of the compounds obtained from the most similar targets over the targets that are less similar. We used two methods for providing such weighting. The first, referred to as $\text{CWS}_\alpha^{\text{sim}}$, assigns a weight to compound $c_{l,j} \in H_i^+$, which was originally active against target p_{i_j} (i.e., $c_{l,j} \in C_{i_j}^+$), that is linearly proportional to the similarity between targets p_i and p_{i_j}. The second, referred to as $\text{CWS}_\alpha^{\text{exp}}$, assigns a weight to $c_{l,j}$ that decays exponentially with j (i.e., the rank of its target p_{i_j} in the list of m most similar targets of p_i). Note that when a compound in H_i^+ is active against more than one of the m most similar targets, it is only considered for its most similar target.

The precise weights assigned to the different compounds in conjunction with the differential weighting scheme of Equation 8.4 are as follows. For the $\text{CWS}_\alpha^{\text{sim}}$, the weight $w_{l,j}$ assigned to compound $c_{l,j}$ is given by

$$w_{l,j} = \frac{(1-\alpha)(|C_i^+| + |H_i^+|)}{\sum\limits_{c_{r,q} \in H_i^+} \text{sim}_t(p_i, p_{i_q})} \text{sim}_t(p_i, p_{i_j}), \tag{8.5}$$

where $\text{sim}_t(p_i, p_{i_j})$ is the target-to-target similarity calculated from $\mathcal{K}_p^{Seq}$, $\mathcal{K}_p^{Aligs}$ or $\mathcal{K}_p^{kligs}$. For the $\text{CWS}_\alpha^{\text{exp}}$, the weight is given by

$$w_{l,j} = \frac{(1-\alpha)(|C_i^+| + |H_i^+|)}{\sum\limits_{c_{r,q} \in H_i^+} 2^{-q}} 2^{-j}. \tag{8.6}$$

8.2.5 Multi-Assay-based SAR Models using Multi-Task Learning

The second class of methods that we developed for building multi-assay-based SAR models for a specific target is based on multi-task learning [106, 37, 38, 36]. Multi-task learning is a transfer learning mechanism designed to improve the generalization performance of a given model by leveraging the domain-specific information contained in the training signals of related tasks. In multi-task learning the model for a task (i.e., class) is learned in parallel with that of other related tasks using a shared representation so as to exploit dependencies between the tasks

during learning. In recent years, various studies have reported promising results with the use of multi-task learning for various problems in cheminformatics [123, 124, 64, 125, 78].

Motivated by the success of these methods, we developed a multi-task learning-based approach that leverages the activity information of the related targets. In this approach, the model for the specific target (p_i) is learned simultaneously with the models of its m related targets ($R_i = \{p_{i_1}, p_{i_2}, \ldots, p_{i_m}\}$) and the dependencies between the different targets and their ligands are captured via the use of target- and compound-specific kernel functions during SVM learning.

The input to this approach is a set of target-compound tuples (p_q, c_j) for each $p_q \in \{p_i\} \cup R_i$. For each target in $\{p_i\} \cup R_i$, tuples corresponding to target-ligand pairs (i.e., $c_j \in C_q^+$) are considered to be positive instances, whereas tuples corresponding to the non-binding compounds (i.e., $c_j \in C_q^-$) are considered to be negative instances. These tuples are used to train an SVM model $f()$ that learns how to separate the positive from the negative tuples. A SAR model for target p_i is then derived from $f()$ by computing $f((p_i, c))$ for each compound c whose activity against target p_i needs to be predicted.

Following the approach used by previously developed SVM-based approaches for learning multi-task models [38, 78], the dependencies between the different targets and compounds are coupled using a fusion-kernel based approach [126]. In this approach, the kernel function $\mathcal{K}_{mt}$ defined on the input target-compound tuples is given by

$$\mathcal{K}_{mt}((p_i, c_j), (p_{i'}, c_{j'})) = \beta \mathcal{K}_p(p_i, p_{i'}) + (1 - \beta)\mathcal{K}_c(c_i, c_{i'}),$$

where $\mathcal{K}_p$ and $\mathcal{K}_c$ are kernel functions defined on the targets and the compounds, respectively, and β ($0 \leq \beta \leq 1$) is a parameter that weights the relative importance of the two components during training. The optimal value of β can be determined either during the learning phase [126, 127, 128] or empirically by performing a grid search over a set of values for these two parameters [129]. Note that the above formulation, by using a kernel function that combines both a target- and a compound-based component, allows SVM to capture relations between similar targets and their compounds and as such transfer knowledge across the different tasks during learning.

In order to use the above formulation, suitable target- and compound-based kernel functions need to be defined. For the target-based kernel function, we used the target-to-target similarity function (Section 8.2.3) that was used to identify the set of related proteins R_i. For example, if the set of related targets were identified using the $\mathcal{K}_p^{kligs}$ similarity function, then the same

function was used as the target-based kernel. For the compound-based kernel function, we used the Tanimoto coefficient (Equation 7.2) as the kernel function for the compounds ($\mathcal{K}_c$) as it has been shown to produce good results for building SVM-based SAR models.

Note that a problem with the definitions of the $\mathcal{K}_p^{Seq}$ and $\mathcal{K}_p^{kligs}$ target-to-target similarity functions is that they lead to Gram-matrices that are symmetric but not necessarily positive semi-definite. For this reason they do not represent valid kernels and as such cannot be used directly for learning SVM models. To overcome this problem we use the approach described in Saigo *et al* [67] that converts a symmetric matrix into positive semi-definite by subtracting from the diagonal of the matrix its smallest negative eigenvalue. For the rest of the discussion, we will assume that this transformation has been applied to the $\mathcal{K}_p^{Seq}$ and $\mathcal{K}_p^{kligs}$ functions.

8.2.6 Multi-Assay-based SAR Models using Multi-Ranking

Finally, motivated by classification approaches that determine the class of an unknown instance by combining predictions of a set of different classifiers, known as classification ensembles [107, 108, 109], we developed an alternate method to improve the quality of a specific target's SAR model by taking advantage of the activity information of its m related targets $R_i = \{p_{i_1}, p_{i_2}, \ldots, p_{i_m}\}$. The main idea of this approach, referred to as *multi-ranking*, is to learn $m + 1$ different SAR models, one for p_i and one for each target in R_i, use each of these models to compute a prediction score for an unknown compound c_i, and then determine the overall prediction score for c_i with respect to target p_i, by combining the $m + 1$ individual predictions. The rationale behind this approach is that the SAR models of p_i's most similar targets should be able to detect the compounds that are active against p_i, and as such they can be used to re-enforce the predictions obtained from p_i's own SAR model.

In the multi-ranking methods, each of the $m + 1$ SAR models are learned using the SVM-based framework described in Section 8.2.1. Specifically, for a target $p_j \in \{p_i\} \cup R_i$, its SAR model is learning using its active compounds C_j as the positive instances and C_j^- as the negative instances. The $m + 1$ individual prediction scores are combined by taking into account two factors. First, the relative importance of p_i's own prediction over that of its m related targets, and second, the relative importance of the m most similar targets among themselves. These two factors are similar to those discussed earlier in the context of semi-supervised learning for assigning different weights to the newly labeled compounds (Section 8.2.4).

We developed three different schemes for accounting for these two factors. Let s'_i be the

prediction score for compound c_l obtained by p_i's SAR model, and let s'_{i_j} be the prediction scores obtained from the SAR models of p_i's m most similar targets. Then the overall prediction score s_i for the three schemes is given by

$$s_i = \alpha s'_i + \frac{1-\alpha}{m} \sum_{1 \leq j \leq m} s'_{i_j}, \tag{8.7}$$

$$s_i = \alpha s'_i + \frac{1-\alpha}{\sum_{1 \leq j \leq m} \text{sim}_t(p_i, p_{i_j})} \sum_{1 \leq j \leq m} \text{sim}_t(p_i, p_{i_j}) s'_{i_j}, \tag{8.8}$$

$$s_i = \alpha s'_i + \frac{1-\alpha}{\sum_{1 \leq j \leq m} 2^{-j}} \sum_{1 \leq j \leq m} 2^{-j} s'_{i_j}, \tag{8.9}$$

where α ($0 \leq \alpha \leq 1$) is a parameter that controls the relative importance of p_i's own prediction over the predictions obtained from the other targets to the overall prediction score. The predictions from the other targets are incorporated by the second term of the above equations. Equation 8.7 simply uses the average prediction score, whereas Equations 8.8 and 8.9 are based on the $\text{CWS}_\alpha^{\text{sim}}$ and $\text{CWS}_\alpha^{\text{exp}}$ schemes (Section 8.2.4), respectively. We refer to these three prediction combination schemes as $\text{MWS}_\alpha^{\text{eql}}$, $\text{MWS}_\alpha^{\text{sim}}$ and $\text{MWS}_\alpha^{\text{exp}}$, respectively.

8.3 Materials

8.3.1 Datasets

We evaluated the performance of the multi-assay-based SAR models using a set of 146 protein targets and their ligands that were derived from various target-specific dose-response confirmatory screening assays. These screens were performed by NIH's Molecular Libraries Probe Production Centers Network (MLPCN) and are available in PubChem [130]. For each protein target its set of active compounds was determined using the activity assessment provided by the screening centers. Compounds that showed different activity signals in different assays against the same target were filtered out. For each of the protein targets, a baseline SAR model was learned and its performance was assessed using a five-fold cross validation. Since the goal of this work is to improve the performance of SAR models, we eliminated the set of targets for which their baseline SAR models achieved an ROC score greater of 0.80 (i.e., targets for which good models can be built by existing methods). This filtering resulted in 117 targets, 15,833 ligands, 16,088 target-ligand activity pairs (compounds can show activity against multiple protein

targets), and an average of 138 active compounds per target. The distribution of the 117 protein targets in terms of their biological activity is shown in Figure 8.1.

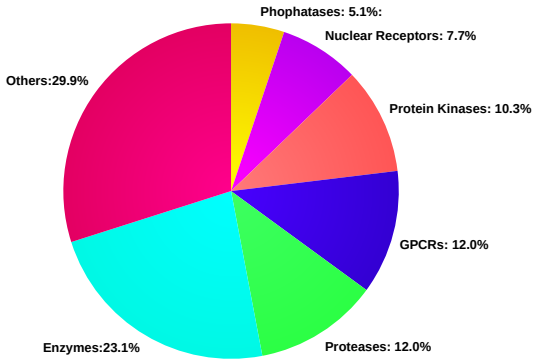


Figure 8.1: Distribution of protein targets

8.3.2 Support Vector Machines

We used the publicly available support vector machine tool SVM *light* [131] that implements an efficient soft margin optimization algorithm. In all of our experiments, we used the default parameters for solving the quadratic programming problem and the default regularization parameter C that controls the margin width.

8.4 Experimental Results

We performed a comprehensive study of the various parameters of the multi-assay-based methods described in Section 8.2 in order to assess the extent to which they lead to SAR model improvements. In the rest of this section we present and summarize the key results from this study. All comparisons are done against the performance achieved by the baseline SAR models (Section 8.2.1). The results being presented correspond to some of the best performing combinations of the various parameters for the different schemes. The complete set of results are available as part of the supplementary material[1] .

[1] http://www-users.cs.umn.edu/~xning/supplementary/

During the experimental evaluation we primarily report the average ROC Table 7.2 improvements achieved by a method over the baseline models across the 117 protein targets, which was computed as the ratio of the ROC score of our schemes over that obtained by the baseline method. We used the geometric mean to compute these average improvements as it is better suited for averaging ratios.

8.4.1 Performance of the Methods Based on Semi-Supervised Learning

Table 8.1 and Table 8.2 show the average improvements achieved by the multi-assay-based semi-supervised learning methods over the baseline methods over the entire set of targets in our dataset. These results show that for certain parameter combinations the multi-assay-based semi-supervised learning approaches can achieve consistent improvements over the baseline model. The best performance achieved by the multi-assay-based semi-supervised learning approach is an average improvement of 1.8% ($\mathcal{K}_p^{Aligs}$ with $\mathrm{LS_{LP}}$).

Comparing the performance achieved by the three target-to-target similarity measures we see that $\mathcal{K}_p^{Aligs}$ achieves consistently better results, whereas the results achieved by $\mathcal{K}_p^{Seq}$ are consistently the worst. The performance of $\mathcal{K}_p^{kligs}$ is between these two. These results suggest that the ligand-based similarity measures can better identify the proteins whose binding sites have similar characteristics than those based on sequence alignment scores. This is not surprising as the ligand-based similarity measures allow for the indirect comparison of the proteins binding sites, whereas the alignment-based methods may fail to compare the actual binding sites. One reason for the performance difference between $\mathcal{K}_p^{Aligs}$ and $\mathcal{K}_p^{kligs}$ is due to the differences in the number of unlabeled instances that exist in the sets of related targets identified by these two methods. The set of related targets identified by $\mathcal{K}_p^{kligs}$ results in a larger number of unlabeled instances (second column of Table 8.1 and Table 8.2) than the corresponding set for $\mathcal{K}_p^{Aligs}$. As a result, the number of positive labeled instances is larger for $\mathcal{K}_p^{kligs}$ than $\mathcal{K}_p^{Aligs}$ (columns labeled as $|H_i|$), which creates more diverse training sets that do not lead to good models. This difference between $\mathcal{K}_p^{Aligs}$ and $\mathcal{K}_p^{kligs}$ occurs because the former selects the related targets by taking into account all the ligands of the selected targets, whereas the latter looks at only the union of the k most similar ligands. As a result, ceteris paribus, targets with more ligands will be selected since they have a higher probability of containing a subset of compounds that are similar to the ligands of the target under consideration.

Table 8.1: Performance improvements of multi-assay-based semi-supervised learning methods with labelling scheme $\mathrm{LS}_{k\mathrm{nn}}$.

				$\mathrm{LS}_{k\mathrm{nn}}$				
	m	$\lvert U_i\rvert$	$\lvert H_i^+\rvert$	$\mathrm{CWS}_{\mathrm{none}}$	$\mathrm{CWS}_\alpha^{\mathrm{exp}}$			
					0.2	0.5	0.8	0.9
$\mathcal{K}_p^{Seq}$	1	86	49	-1.7%	-3.8%	-2.0%	-1.8%	-0.9%
	3	274	113	-2.1%	-3.0%	-2.2%	-1.3%	-1.0%
	5	449	146	-2.8%	-3.7%	-2.9%	-2.1%	-1.3%
	7	594	167	-2.9%	-2.6%	-1.7%	-2.2%	-1.6%
	9	752	182	-3.8%	-3.0%	-2.5%	-2.3%	-1.9%
$\mathcal{K}_p^{Aligs}$	1	41	26	1.1%	-0.4%	0.9%	0.8%	0.9%
	3	122	70	0.7%	-1.0%	0.5%	**1.2%**	0.9%
	5	216	106	-0.5%	-1.5%	-0.2%	0.6%	0.8%
	7	317	134	-0.7%	-1.2%	-0.5%	0.4%	0.5%
	9	432	157	-1.1%	-1.2%	-0.4%	-0.5%	-0.1%
$\mathcal{K}_p^{kligs}$	1	114	61	0.7%	-0.4%	0.8%	0.5%	0.8%
	3	364	135	-0.4%	-0.9%	-0.5%	-0.2%	0.2%
	5	625	179	-1.0%	-1.0%	-0.8%	-0.5%	0.6%
	7	894	208	-1.4%	-1.5%	-1.4%	-0.6%	-0.4%
	9	1181	229	-1.8%	-1.7%	-1.3%	-1.2%	-1.2%

In this table, m is the number of related targets, $|U_i|$ is the total number of unlabeled compounds, $|H_i^+|$ is the number of unlabeled compounds that were labeled as positive by the labeling scheme $\mathrm{LS}_{k\mathrm{nn}}$. The columns labeled 0.2, 0.4, 0.8, and 0.9 correspond to the value of the α parameter for $\mathrm{CWS}_\alpha^{\mathrm{exp}}$. The $\mathrm{LS}_{\mathrm{LP}}$ was applied on the 5-nearest-neighbor graph of the labeled and unlabeled compounds. The $\mathcal{K}_p^{kligs}$ target-to-target similarity used $k = 5$. Bold-faced numbers indicate the best performing scheme under a certain combination of target-to-target similarity function and labeling scheme. Underlined numbers represent schemes with positive improvements.

Table 8.2: Performance improvements of multi-assay-based semi-supervised learning methods with labelling scheme $\mathrm{LS_{LP}}$.

				$\mathrm{LS_{LP}}$				
	m	$\lvert U_i\rvert$	$\lvert H_i^+\rvert$	$\mathrm{CWS_{none}}$	$\mathrm{CWS_\alpha^{exp}}$			
					0.2	0.5	0.8	0.9
$\mathcal{K}_p^{Seq}$	1	86	68	-2.1%	-3.5%	-2.3%	-2.5%	-2.0%
	3	274	203	-5.8%	-6.5%	-5.9%	-5.8%	-4.1%
	5	449	367	-7.5%	-7.8%	-7.6%	-7.0%	-6.2%
	7	594	512	-7.3%	-7.3%	-7.1%	-7.3%	-6.6%
	9	752	621	-8.3%	-8.3%	-8.0%	-7.5%	-6.8%
$\mathcal{K}_p^{Aligs}$	1	41	28	**1.8%**	0.7%	1.7%	1.3%	0.8%
	3	122	78	1.3%	0.2%	1.5%	1.5%	**1.8%**
	5	216	122	0.8%	0.4%	0.8%	0.9%	1.5%
	7	317	243	-0.5%	-0.5%	-0.4%	0.6%	0.9%
	9	432	324	-1.3%	-1.0%	-0.9%	0.0%	0.8%
$\mathcal{K}_p^{kligs}$	1	114	89	-0.3%	-0.6%	0.0%	-0.9%	-0.8%
	3	364	302	-0.5%	-0.5%	-0.5%	-0.5%	-0.4%
	5	625	543	-1.5%	-1.5%	-1.3%	-1.2%	-1.1%
	7	894	703	-1.7%	-1.7%	-1.6%	-1.9%	-1.9%
	9	1181	945	-2.6%	-2.3%	-2.2%	-2.0%	-2.2%

In this table, m is the number of related targets, $|U_i|$ is the total number of unlabeled compounds, $|H_i^+|$ is the number of unlabeled compounds that were labeled as positive by the labeling scheme $\mathrm{LS_{LP}}$. The columns labeled 0.2, 0.4, 0.8, and 0.9 correspond to the value of the α parameter for $\mathrm{CWS_\alpha^{exp}}$. The $\mathrm{LS_{LP}}$ was applied on the 5-nearest-neighbor graph of the labeled and unlabeled compounds. The $\mathcal{K}_p^{kligs}$ target-to-target similarity used $k = 5$. Bold-faced numbers indicate the best performing scheme under a certain combination of target-to-target similarity function and labeling scheme. Underlined numbers represent schemes with positive improvements.

Comparing the performance of the two labeling schemes ($\mathrm{LS}_k\mathrm{nn}$ and $\mathrm{LS_{LP}}$) we see that $\mathrm{LS}_k\mathrm{nn}$ tends to label as positive a smaller fraction of the unlabeled compound than $\mathrm{LS_{LP}}$. Depending on the target-to-target similarity method being used, this can either lead to better or worse results. In the case of $\mathcal{K}_p^{kligs}$, for which the total number of unlabeled instances is large, the performance achieved by $\mathrm{LS_{LP}}$ is worse than that of $\mathrm{LS}_k\mathrm{nn}$, as it ends up labeling too many

instances as positive. On the other hand, when $\mathcal{K}_p^{Aligs}$ is used, for which the total number of unlabeled instances is small, LS_{LP} performs better than $\text{LS}_{k\text{nn}}$. However, when the number of compounds that are being labeled by both schemes is approximately the same (e.g., $\mathcal{K}_p^{Aligs}$ and $m = 1, 3, 5$), the LS_{LP} achieves better results, suggesting that it does a better job in labeling the unlabeled compounds.

Comparing the performance of the different compound weighting schemes (CWS_{none} and $\text{CWS}_{\alpha}^{\text{exp}}$) we see that as the number of unlabeled compounds that is labeled as positive increases, $\text{CWS}_{\alpha}^{\text{exp}}$ does better than CWS_{none}. This is because under these conditions $\text{CWS}_{\alpha}^{\text{exp}}$, by decreasing the mis-classification weight of each newly labeled compound, reduces the overall influence that these compounds in the learned model. Also, not surprisingly, $\text{CWS}_{\alpha}^{\text{exp}}$'s performance improves when more weight is given to the original set of positive instances (i.e., the known ligands of each target) than the positive instances obtained as a result of the semi-supervised learning method (i.e., putative ligands).

Finally, comparing the performance of the schemes as the number m of related targets changes we see that, in general, their performance tends to initially improve as m increases and then it starts to degrade. Depending on the specific set of parameters, the best performance is usually achieved when 3–5 related targets are used. However, the methods based on $\mathcal{K}_p^{Seq}$ exhibit different performance characteristics as their performance consistently decreases as m increases.

8.4.2 Performance of the Methods Based on Multi-Task Learning

The average improvements achieved by the multi-assay-based multi-task learning methods over the baseline models are shown in Table 8.3. These results show that the ROC scores achieved by these models are usually higher than those achieved by the baseline model. Both the $\mathcal{K}_p^{Aligs}$ and $\mathcal{K}_p^{kligs}$ kernel functions achieve substantial improvements that range between 2.9% and 7.2%. Moreover, even the $\mathcal{K}_p^{Seq}$ kernel function, which in the context of semi-supervised learning (Table 8.1 and Table 8.2) always resulted in lower ROC scores than the baseline model, is able to achieve an improvement of 2.2% for $m = 1$ and $\beta = 0.1$.

Table 8.3: Performance improvements of the multi-assay-based multi-task learning methods.

	m	β			
		0.1	0.2	0.5	0.8
$\mathcal{K}_p^{Seq}$	1	2.2%	1.8%	0.8%	0.5%
	3	1.1%	0.8%	-0.8%	-1.8%
	5	-0.4%	-0.7%	-1.8%	-2.8%
	7	-0.5%	-1.0%	-1.8%	-3.1%
	9	-0.8%	-1.0%	-2.5%	-4.2%
$\mathcal{K}_p^{Aligs}$	1	3.3%	3.4%	3.2%	3.2%
	3	5.9%	5.9%	5.7%	5.8%
	5	5.4%	5.4%	5.3%	5.1%
	7	4.9%	5.0%	4.8%	4.4%
	9	4.9%	5.0%	4.8%	4.4%
$\mathcal{K}_p^{kligs}$	1	4.3%	3.7%	3.1%	2.9%
	3	7.0%	7.1%	4.9%	4.0%
	5	7.0%	**7.2%**	5.5%	4.1%
	7	6.4%	6.8%	5.3%	3.5%
	9	6.6%	6.9%	5.2%	3.4%

In this table, m is the number of related targets. The columns labeled 0.1, 0.2, 0.5 and 0.8 correspond to the value of the β parameter (i.e., weight on the target-based kernel). The $\mathcal{K}_p^{kligs}$ target-to-target similarity used $k = 5$. Bold-faced numbers indicates the best performance of multi-assay-based multi-task learning. Underlined numbers represent schemes with positive improvements.

Comparing the three kernel functions, we see that out of the 20 cases shown in Table 8.3, $\mathcal{K}_p^{kligs}$ achieves the best performance in 14 cases, $\mathcal{K}_p^{Aligs}$ in 6, whereas $\mathcal{K}_p^{Seq}$ never outperforms the other methods. The best overall performance is achieved by $\mathcal{K}_p^{kligs}$, which is a 7.2% improvement over the baseline model. The relatively poor performance of $\mathcal{K}_p^{Seq}$ over the ligand-based kernel functions is consistent with the earlier results involving semi-supervised learning and further re-enforces the fact that it is not well-suited for identifying appropriate targets for improving the accuracy of SAR models. However, in light of the results obtained by semi-supervised learning, the relative performance advantage of $\mathcal{K}_p^{kligs}$ over $\mathcal{K}_p^{Aligs}$ is somewhat surprising. This is due to the higher diversity among the targets identified by $\mathcal{K}_p^{kligs}$ and is further discussed

later in Section 8.5. Comparing the performance of the ligand-based kernel functions as the number m of related targets increases, we observe that for $\mathcal{K}_p^{Aligs}$ and $\mathcal{K}_p^{kligs}$, the performance first improves and then degrades. The best performance is usually achieved when 3–5 related targets are used. However, for $\mathcal{K}_p^{Seq}$, as was the case with semi-supervised learning, the performance consistently decreases as m increases. Finally, comparing the performance of the two best-performing kernel functions as the value of β changes (Equation 8.2.5), we see that they exhibit distinctly different trends. The performance of $\mathcal{K}_p^{Aligs}$ remains largely unchanged as β ranges from 0.1 to 0.8, whereas the performance of $\mathcal{K}_p^{kligs}$ tends to markedly decrease for higher values of β. Thus, these results indicate that for $\mathcal{K}_p^{kligs}$, the best way to combine the target- and compound-based kernels in the fusion kernel formulation is by giving less weight to the target kernel and a higher-weight to the compound component.

8.4.3 Performance of Multi-Ranking

The average improvements achieved by the multi-ranking-based models over the baseline models are shown in Table 8.4. These results show that for a wide-range of parameter combinations, multi-ranking can achieve considerable improvements over the baseline models. The relative advantages of the three target-to-target similarity measures are consistent with the results obtained using the multi-assay-based multi-task learning method. $\mathcal{K}_p^{kligs}$ tends to perform the best, with some schemes achieving an average improvement of 7.0%, whereas $\mathcal{K}_p^{Seq}$ does relatively the worst, with its best performing parameter combination achieving only a 3.1% improvement. However, this 3.1% improvement is still substantially higher than the best performance achieved by any of the previous methods using this target-to-target similarity function or kernel function. Comparing the three prediction combination schemes $\text{MWS}_\alpha^{\text{eql}}$, $\text{MWS}_\alpha^{\text{sim}}$ and $\text{MWS}_\alpha^{\text{exp}}$, we see that on average $\text{MWS}_\alpha^{\text{exp}}$ performs the best, followed by $\text{MWS}_\alpha^{\text{sim}}$, and $\text{MWS}_\alpha^{\text{eql}}$ is the worst. This suggests that models from different targets do show different characteristics and function differently. Also, not surprising, the best performance is usually achieved when the original models contribute more to the overall prediction (i.e., $\alpha = 0.8$).

Comparing the performance of the multi-ranking approach as the number m of related targets increases, we observe that in general the performance initially improves and then it starts to degrade. The $\text{MWS}_\alpha^{\text{exp}}$ scheme is an exception, as in many cases its performance does not degrade. This is due to the exponential weighting on less similar targets which brings little impact on the combination of predictions. The best performance usually happens when 5–7

related targets are used. The degradation of performance associated with large m is because less similar models make less reliable predictions, and thus combining them will not introduce any benefits.

Table 8.4: Performance improvements of multi-assay-based multi-ranking methods.

	m	$\text{MWS}_\alpha^{\text{eql}}$				$\text{MWS}_\alpha^{\text{sim}}$				$\text{MWS}_\alpha^{\text{exp}}$			
		0.2	0.5	0.8	0.9	0.2	0.5	0.8	0.9	0.2	0.5	0.8	0.9
$\mathcal{K}_p^{Seq}$	1	-2.7%	1.7%	1.8%	1.3%	-2.7%	1.7%	1.8%	1.3%	-3.0%	1.7%	2.0%	1.3%
	3	-13.5%	0.7%	2.3%	1.5%	-11.8%	1.6%	2.5%	1.4%	-12.0%	0.2%	3.1%	1.9%
	5	-19.8%	-5.0%	2.1%	1.5%	-18.6%	-2.8%	2.1%	2.0%	-17.0%	-1.5%	3.0%	2.1%
	7	-20.2%	-9.8%	1.7%	1.2%	-19.4%	-6.3%	2.1%	1.7%	-18.0%	-2.1%	2.9%	2.1%
	9	-23.9%	-17.8%	2.1%	1.3%	-23.8%	-14.6%	2.7%	1.6%	-22.1%	-2.8%	2.8%	2.1%
$\mathcal{K}_p^{Aligs}$	1	1.3%	2.9%	2.4%	1.7%	1.3%	2.9%	2.4%	1.7%	1.0%	2.9%	2.2%	1.7%
	3	-6.6%	3.8%	2.5%	1.8%	-5.7%	4.2%	2.5%	1.9%	-4.9%	4.2%	3.1%	2.6%
	5	-12.1%	-0.4%	2.0%	1.8%	-11.7%	0.5%	2.1%	1.9%	-10.2%	2.7%	4.0%	2.9%
	7	-12.9%	-4.6%	3.0%	1.9%	-12.7%	-3.4%	3.1%	1.8%	-11.0%	2.1%	4.0%	3.0%
	9	-13.1%	-7.8%	4.6%	2.0%	-13.2%	-7.1%	4.4%	2.0%	-11.3%	2.4%	3.9%	3.0%
$\mathcal{K}_p^{kligs}$	1	0.8%	4.0%	4.2%	3.1%	0.8%	4.0%	4.2%	3.1%	1.0%	4.0%	4.0%	3.1%
	3	-5.1%	5.1%	5.3%	3.2%	-4.5%	5.8%	5.5%	3.2%	-3.3%	6.4%	6.0%	4.8%
	5	-10.2%	1.4%	6.3%	4.0%	-9.7%	2.4%	**6.4%**	4.1%	-6.2%	6.6%	**7.0%**	5.4%
	7	-13.6%	-5.4%	**6.5%**	4.4%	-13.4%	-4.6%	6.1%	4.4%	-11.0%	6.0%	**7.0%**	5.5%
	9	-16.2%	-10.6%	5.6%	3.4%	-16.1%	-9.8%	5.9%	3.6%	-14.2%	5.2%	**7.0%**	5.5%

In this table, m is the number of related targets. The columns labeled 0.2, 0.5, 0.8 and 0.9 correspond to the value of the α parameter for $\text{MWS}_\alpha^{\text{eql}}$, $\text{MWS}_\alpha^{\text{sim}}$ and $\text{MWS}_\alpha^{\text{exp}}$, respectively. The $\mathcal{K}_p^{kligs}$ target-to-target similarity used $k = 5$. Bold-faced numbers indicate the best performing scheme under a certain combination of target-to-target similarity function and prediction combination scheme. Underlined numbers represent schemes with positive improvements.

8.5 Discussion & Conclusions

8.5.1 Overall Performance

Table 8.5: Summary of the performance improvements of the different multi-assay-based methods.

methods	target-to-target similarity	compound labeling	weighting scheme	weight	m	best average improvement	ROC	% improved targets	p-value
semi-supervised learning	$\mathcal{K}_p^{Aligs}$	LS_{LP}	CWS_α^{exp}	$\alpha = 0.9$	3	<u>1.8%</u>	0.66	54%	5.69e-02
multi-task learning	$\mathcal{K}_p^{kligs}$	-	-	$\beta = 0.2$	5	**<u>7.2%</u>**	0.70	63%	6.39e-05
multi-ranking	$\mathcal{K}_p^{kligs}$	-	MWS_α^{exp}	$\alpha = 0.8$	5	<u>7.0%</u>	0.70	85%	7.06e-11

In this table, m is the number of related targets, *best average improvement* is the geometric mean achieved for each of the multi-assay-based methods under the parameter combination shown in the corresponding row, *ROC* is the average area under the ROC curved achieved by each scheme, *% improved targets* is the percentage of the 117 targets for which the multi-assay-based method resulted in better performance. The p value of the statistical significance test using the paired students t test of the results achieved by each scheme over the baseline approach is shown in the column labeled *p-value*. The average ROC for baseline model was 0.65. Bold-faced numbers indicate the best performance over all multi-assay-based methods. Underlined numbers represent the schemes with positive improvements.

Table 8.6: ROC scores for baseline, semi-supervised learning, multi-task learning and multi-ranking

target -ID	base-line	semi-supervised learning	multi -task learning	multi -ranking	target -ID	base-line	semi-supervised learning	multi -task learning	multi -ranking
1	0.300	0.600	0.750	0.500	60	0.664	0.674	0.708	0.705
2	0.311	0.200	0.578	0.467	61	0.666	0.684	0.739	0.718
3	0.311	0.422	0.778	0.511	62	0.669	0.641	0.651	0.686
4	0.392	0.404	0.531	0.445	63	0.669	0.661	0.649	0.669

Continued on next page

Table 8.6 – *Continued from previous page*

target -ID	base-line	semi-supervised learning	multi -task learning	multi -ranking	target -ID	base-line	semi-supervised learning	multi -task learning	multi -ranking
5	0.400	0.400	0.800	0.600	64	0.672	0.671	0.548	0.659
6	0.400	0.511	0.556	0.511	65	0.675	0.669	0.711	0.691
7	0.412	0.500	0.650	0.650	66	0.675	0.704	0.708	0.740
8	0.475	0.500	0.713	0.637	67	0.676	0.677	0.713	0.701
9	0.511	0.528	0.507	0.519	68	0.676	0.726	0.622	0.730
10	0.525	0.575	0.694	0.653	69	0.685	0.677	0.689	0.704
11	0.533	0.600	0.778	0.644	70	0.686	0.671	0.684	0.693
12	0.536	0.558	0.582	0.573	71	0.686	0.739	0.624	0.706
13	0.541	0.572	0.465	0.570	72	0.687	0.696	0.716	0.705
14	0.544	0.584	0.704	0.600	73	0.688	0.640	0.766	0.744
15	0.547	0.539	0.530	0.541	74	0.695	0.689	0.641	0.700
16	0.551	0.600	0.608	0.653	75	0.696	0.664	0.680	0.736
17	0.555	0.562	0.587	0.630	76	0.699	0.698	0.663	0.700
18	0.561	0.579	0.667	0.602	77	0.700	0.700	0.800	0.750
19	0.561	0.589	0.450	0.613	78	0.702	0.682	0.767	0.757
20	0.567	0.582	0.572	0.579	79	0.702	0.725	0.740	0.742
21	0.576	0.565	0.633	0.609	80	0.707	0.731	0.759	0.742
22	0.576	0.592	0.664	0.672	81	0.710	0.688	0.748	0.742
23	0.586	0.583	0.563	0.592	82	0.711	0.711	0.822	0.844
24	0.596	0.604	0.684	0.654	83	0.712	0.744	0.784	0.806
25	0.600	0.562	0.675	0.662	84	0.720	0.792	0.736	0.776
26	0.600	0.578	0.511	0.644	85	0.728	0.768	0.872	0.756
27	0.601	0.601	0.636	0.619	86	0.730	0.732	0.723	0.727
28	0.606	0.600	0.567	0.604	87	0.731	0.719	0.756	0.746
29	0.609	0.609	0.687	0.638	88	0.732	0.719	0.673	0.721
30	0.610	0.641	0.707	0.698	89	0.733	0.711	0.822	0.800
31	0.611	0.604	0.641	0.631	90	0.736	0.759	0.819	0.777

Continued on next page

Table 8.6 – *Continued from previous page*

target -ID	base-line	semi-supervised learning	multi -task learning	multi -ranking	target -ID	base-line	semi-supervised learning	multi -task learning	multi -ranking
32	0.613	0.713	0.837	0.737	91	0.737	0.730	0.644	0.704
33	0.616	0.584	0.691	0.653	92	0.738	0.863	0.850	0.888
34	0.619	0.623	0.592	0.601	93	0.743	0.762	0.722	0.749
35	0.621	0.617	0.607	0.635	94	0.747	0.788	0.762	0.772
36	0.624	0.596	0.627	0.661	95	0.752	0.736	0.816	0.760
37	0.626	0.607	0.528	0.649	96	0.755	0.784	0.649	0.718
38	0.626	0.626	0.625	0.626	97	0.756	0.747	0.715	0.766
39	0.631	0.637	0.651	0.640	98	0.756	0.772	0.827	0.799
40	0.631	0.650	0.635	0.656	99	0.756	0.774	0.832	0.788
41	0.635	0.643	0.610	0.663	100	0.758	0.791	0.827	0.788
42	0.636	0.644	0.700	0.662	101	0.761	0.761	0.771	0.761
43	0.637	0.641	0.672	0.660	102	0.762	0.762	0.737	0.788
44	0.637	0.650	0.597	0.681	103	0.768	0.772	0.744	0.732
45	0.637	0.663	0.812	0.737	104	0.769	0.773	0.764	0.786
46	0.638	0.638	0.700	0.700	105	0.770	0.766	0.784	0.786
47	0.641	0.555	0.608	0.633	106	0.772	0.759	0.788	0.812
48	0.641	0.630	0.680	0.659	107	0.772	0.764	0.758	0.785
49	0.643	0.635	0.682	0.667	108	0.778	0.756	0.711	0.756
50	0.645	0.641	0.677	0.673	109	0.785	0.793	0.782	0.793
51	0.645	0.648	0.656	0.655	110	0.795	0.806	0.829	0.817
52	0.647	0.647	0.666	0.639	111	0.797	0.791	0.859	0.837
53	0.650	0.800	0.750	0.800	112	0.797	0.809	0.843	0.816
54	0.651	0.666	0.651	0.671	113	0.799	0.807	0.770	0.811
55	0.654	0.595	0.611	0.650	114	0.800	0.700	0.750	0.750
56	0.659	0.671	0.674	0.675	115	0.800	0.700	0.800	0.850
57	0.660	0.681	0.714	0.693	116	0.800	0.760	0.760	0.808
58	0.660	0.682	0.612	0.664	117	0.800	0.800	0.800	0.800

Continued on next page

Table 8.6 – *Continued from previous page*

target -ID	base- line	semi- supervised learning	multi -task learning	multi -ranking	target -ID	base- line	semi- supervised learning	multi -task learning	multi -ranking
59	0.662	0.703	0.767	0.697					

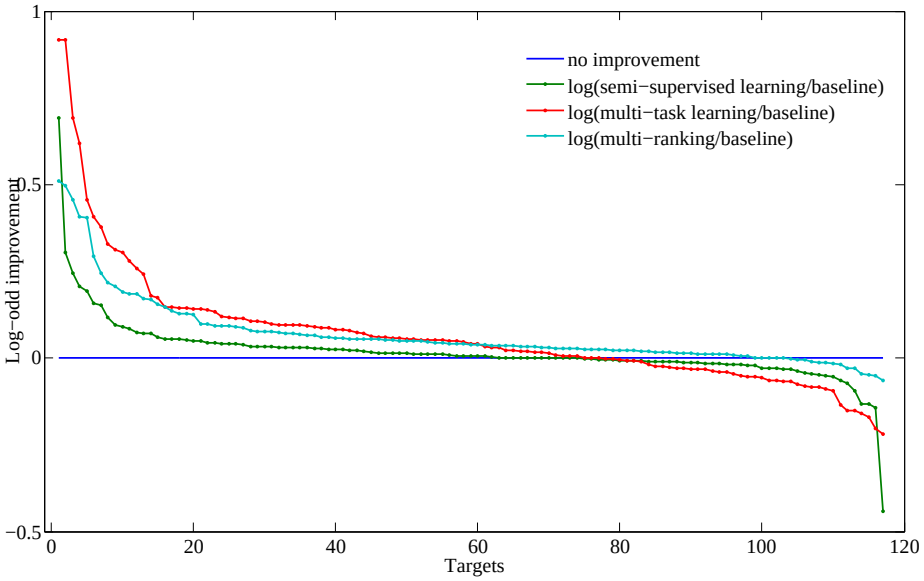


Figure 8.2: Improvement log-ratio distribution

Table 8.5 summarizes the best results achieved by the three multi-assay-based schemes developed in this work, whereas Figure 8.2 presents a finner grain view of these results by plotting the log-ratio of the ROC score achieved by each of them over the ROC score achieved by the baseline model for the 117 targets in our dataset. Note that for each multi-assay-based

scheme, the results in Figure 8.2 are presented in a non-increasing order according to these log-ratios. The actual ROC scores for the 117 target and the four schemes (baseline and multi-assay-based methods) are shown in Table 8.6.

These results provide strong evidence that the multi-assay-based approaches can improve the quality of target-specific SAR models by utilizing activity information from related targets. When viewed together, these results point to the following trends. First, in terms of average ROC, multi-task learning and multi-ranking perform comparably and achieve the best overall results, whereas in terms of performance consistency over the baseline approach, the multi-ranking method performs the best leading to better ROC scores for 99 out of the 117 targets (85% of the targets). Moreover, the performance gains achieved by multi-task learning and multi-ranking are statistically significant using a paired t test with p values of 1.3e-05 and 8.6e-14, respectively. Second, the target-to-target similarity function that takes into account the entire sequence of the protein targets, does not perform as well as the ligand-based functions. This is due to the fact that the latter approaches indirectly account for the characteristics of the ligand binding sites, whereas the former does not. Third, the three multi-assay-based methods behave differently for the two ligand-based target-to-target similarity functions. Semi-supervised learning performs best for $\mathcal{K}_p^{Aligs}$, whereas the other two perform better for $\mathcal{K}_p^{kligs}$. As discussed in Section 8.4.1, $\mathcal{K}_p^{kligs}$ tends to select targets that have a large number of ligands. In the context of semi-supervised learning methods, this leads to a large number of unlabeled instances, which is the reason behind the lower performance of $\mathcal{K}_p^{kligs}$ over $\mathcal{K}_p^{Aligs}$. However, in the case of the methods based on multi-task learning and multi-ranking, this property of $\mathcal{K}_p^{kligs}$ actually leads to improved performance. This is because the targets selected by $\mathcal{K}_p^{kligs}$ tend to contain more diverse sets of compounds than those selected by $\mathcal{K}_p^{Aligs}$ (the average pairwise compound similarity of $\mathcal{K}_p^{Aligs}$'s five most similar targets was is 0.0138, whereas the corresponding similarity for $\mathcal{K}_p^{kligs}$ was only 0.0071) and consequently, there is a higher degree of diversity among the set of targets that are being selected by $\mathcal{K}_p^{kligs}$. This increase in diversity enables multi-task learning to exploit different areas of the chemical space during learning and enables multi-ranking to compute more robust predictions by averaging over less homogeneous models. Such increases in prediction heterogeneity are known to lead to performance improvements for ensemble-based methods [107, 108, 109].

8.5.2 Performance on Actual Inactives

One of the decisions that were made in building the various SAR models for each target p_i, was to ignore the inactive compounds that may be available for p_i (see the discussion in Section 8.2.1 for the reasons underlying this decision) and to use instead a random subset of the ligands of the other targets as the set of inactive compounds for p_i during model learning (following a learning from only positive and unlabeled instances framework [113]). To assess the extent to which this framework for building the multi-assay-based models can still lead to improvements over the baseline models when the actual inactives are being predicted we performed an experiment in which the test set consisted of both actual active and actual inactive compounds. These experiments showed that the multi-assay-based models can still improve over the baseline models, achieving an average ROC improvement of 2.2%, 3.9%, and 2.4%, for the semi-supervised learning, multi-task learning, and multi-ranking, respectively.

8.5.3 False Positive Sensitivity

To assess the robustness of the models in the presence of false positives, we performed a series of experiments in which different multi-assay-based and baselines models were built in which a certain fraction of false positives were included as additional positive training instances. We used two different approaches for selecting the false positives. The first selected the false positives from the actual inactives of each target, whereas the second selected the false positives from the ligands of the other targets that are not true positives for the target under consideration (i.e., the same approach that was used to generate the negative instances for the results presented in Section 8.4). The resulting models were then tested on compounds consisting of confirmed actives and randomly selected actives from other targets as inactives. To ensure that the comparisons with the baseline method was done fairly, the same set of training sets (i.e., with false positives) were also used to build the baseline models.

Table 8.7: Performance of chemogenomics- and multi-assay-based approaches relative to the baseline models.

family	scheme	Multi-Task Learning				Multi-Ranking				
		target-to-target similarity	β	m	impr	target-to-target similarity	weighting scheme	α	m	impr
Phosphatases	ChmGnmics	$\mathcal{K}_p^{Aligs}$	0.2	6	-5.2%	$\mathcal{K}_p^{Seq}$	$\text{MWS}_\alpha^{\text{sim}}$	0.9	6	0.2%
	MABfamily	$\mathcal{K}_p^{kligs}$	0.2	1	1.2%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.9	3	-2.3%
	MABglobal	$\mathcal{K}_p^{Aligs}$	0.2	1	**6.9%**	$\mathcal{K}_p^{Aligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	3	6.5%
Nuclear Receptors	ChmGnmics	$\mathcal{K}_p^{kligs}$	0.2	9	14.6%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	9	11.8%
	MABfamily	$\mathcal{K}_p^{kligs}$	0.2	3	14.0%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.5	3	13.3%
	MABglobal	$\mathcal{K}_p^{kligs}$	0.2	5	**18.1%**	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	5	10.6%
Protein Kinases	ChmGnmics	$\mathcal{K}_p^{Aligs}$	0.5	12	4.3%	$\mathcal{K}_p^{Aligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	12	8.2%
	MABfamily	$\mathcal{K}_p^{kligs}$	0.2	5	11.3%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{sim}}$	0.8	7	9.8%
	MABglobal	$\mathcal{K}_p^{kligs}$	0.2	1	**15.3%**	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	7	14.5%
GPCRs	ChmGnmics	$\mathcal{K}_p^{Aligs}$	0.2	14	-3.6%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	14	1.4%
	MABfamily	$\mathcal{K}_p^{Aligs}$	0.2	1	1.4%	$\mathcal{K}_p^{Aligs}$	$\text{MWS}_\alpha^{\text{sim}}$	0.8	7	2.6%
	MABglobal	$\mathcal{K}_p^{Aligs}$	0.2	3	**6.8%**	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	1	3.2%
Proteases	ChmGnmics	$\mathcal{K}_p^{Aligs}$	0.2	14	0.8%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	14	5.1%
	MABfamily	$\mathcal{K}_p^{kligs}$	0.2	1	6.7%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	3	6.4%
	MABglobal	$\mathcal{K}_p^{kligs}$	0.2	5	**12.1%**	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	7	9.3%
Enzymes	ChmGnmics	$\mathcal{K}_p^{Aligs}$	0.2	27	-6.6%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	27	1.3%
	MABfamily	$\mathcal{K}_p^{Aligs}$	0.2	1	0.4%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	7	1.3%
	MABglobal	$\mathcal{K}_p^{kligs}$	0.2	5	2.7%	$\mathcal{K}_p^{kligs}$	$\text{MWS}_\alpha^{\text{exp}}$	0.8	9	**2.9%**

In this table, β is the weight on target similarity for multi-assay-based multi-task learning method, α is the weight for $\text{MWS}_\alpha^{\text{eql}}$, $\text{MWS}_\alpha^{\text{sim}}$ and $\text{MWS}_\alpha^{\text{exp}}$, m is either the number of related targets (multi-assay-based approaches) or size of the protein family (chemogenomics-based approaches), *impr* is the performance of certain multi-assay-based scheme under corresponding combination of parameters for each protein family. *ChemGnmics* denotes the results obtained by the chemogenomics-based approach, *MABfamily* denotes the results obtained by the family-focused multi-assay-based approach, and *MABglobal* denotes the results obtained by the actual multi-assay-based approach. Bold-faced numbers indicate the best performing scheme for each protein family.

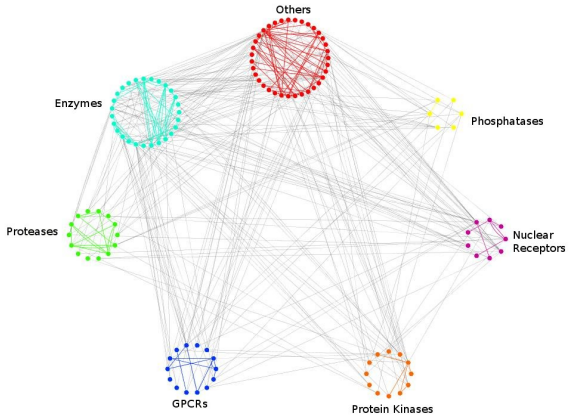


Figure 8.3: Connectivity pattern between the related proteins ($m = 3$) for the different families.

Table 8.8: Performance improvements of the different multi-assay-based methods in the presence of false positives.

	Actual inactives as false positives				Non-actives as false positives			
method	1%	2%	5%	10%	1%	2%	5%	10%
semi-supervised	1.8%	1.6%	1.1%	0.7%	1.8%	1.6%	1.0%	0.7%
learning	(5.88e-2)	(1.32e-1)	(4.29e-1)	(4.58e-1)	(5.47e-2)	(1.35e-1)	(3.97e-1)	(5.49e-1)
multi-task	7.0%	6.4%	3.5%	3.4%	6.9%	6.2%	3.3%	3.2%
learning	(2.23e-4)	(1.23e-3)	(2.05e-2)	(2.60e-2)	(3.61e-4)	(2.46e-3)	(3.75e-2)	(1.96e-2)
multi-ranking	6.9%	5.8%	2.0%	1.5%	6.9%	5.5%	1.9%	1.5%
	(1.14e-10)	(2.82e-8)	(1.12e-4)	(5.79e-2)	(1.38e-10)	(3.15e-8)	(1.21e-4)	(5.84e-2)

For each learning method in the rows, the numbers on top (percentage numbers) are the performance improvements, and the numbers on bottom (powered numbers) are the p-values from paired t-test.

The average ROC improvements of the best parameter combination of the multi-assay-based models over the baseline model for these sets of experiments are shown in Table 8.8 (this table is the analog of Table 8.5). Results for different number of false positives as a percentage of the actual number of positives are presented ranging from 1% up to 10%. These results

show that even in the presence of false positives, the multi-assay-based approaches lead to improvements over the baseline approach. For small percentages of false positives (1%–2%), the relative gains achieved by the multi-task learning and multi-ranking approaches remain considerable. However, as the percentage of false positives increases, the relative performance gains over the baseline approach decreases. For example, at 10% of false positives, the improvements achieved by multi-task learning and multi-ranking drops to around 3.4% and 1.5%, respectively. This suggests that in the presence of a considerable number of false positive instances, these approaches fail to identify a good set of related targets and/or utilize their activity information to improve the quality of the SAR models. Finally, these results suggest that among the multi-assay-based methods, multi-task learning is more robust in dealing with the presence of a considerable number of false positives.

8.5.4 Comparison with Chemogenomics-Based Approaches

As discussed earlier, the quality of target-specific SAR models can be improved by using chemogenomics-based approaches that take into account the activity information from all the proteins in the same family. Within the context of the methods introduced in this chapter, these chemogenomics-based approaches can be viewed as a special case of the multi-assay-based models in which all the proteins of the same family become the set of related proteins. Table 8.7 shows the performance gains achieved by the chemogenomics- and multi-assay-based approaches over the baseline models on the six protein families in our dataset that contain at least 4 members (i.e., the proteins in the "other" class of Figure 8.1 were not included). These results correspond to the parameter combinations that achieved the best performance for the different schemes. The machine learning methods that were used in the study are based on multi-task learning and multi-ranking, which outperform those based on semi-supervised learning. Results for three different schemes are provided, one using the chemogenomics approach (labeled "ChmGnmics") and two using the multi-assay-based approach (labeled "MABfamily" and "MABglobal"). The MABglobal method corresponds to the multi-assay-based schemes that were described in Section 8.2, whereas the MABfamily method corresponds to their variants in which the set of related targets were identified only from the same family. These results show that even though the chemogenomics-based approaches are able to improve the quality of the target-specific SAR models, these improvements are smaller than those obtained by the multi-assay-based approaches. Averaged over the 82 proteins in these six families, the

multi-assay-based approaches achieve a 4.33% improvement over the chemogenomics-based approaches (best multi-assay-based scheme vs best chemogenomics-based scheme). In addition, comparing the performance achieved by the MABfamily variant of the multi-assay-based methods we see that they perform 0.9% better than the chemogenomics-based approaches and 3.3% worse than the actual multi-assay-based approaches (MABglobal). These results show that higher performance gains can be obtained by not utilizing the activity information from all the proteins in the family (MABfamily vs ChmGnmics) and that even further gains can be achieved by utilizing activity information from proteins of different families (MABglobal vs MABfamily).

To illustrate the cross-family nature of the multi-assay-based methods, Figure 8.3 shows the set of related proteins for the different proteins within and across the different families ($\mathcal{K}_p^{kligs}$ and $m = 3$). This figure shows that for nearly all protein targets, a fair number of their related targets (66.5%) come from targets that belong to other families and include proteins that are substantially different from each other (e.g., kinases and GPCRs).

8.5.5 Conclusion

In this chapter, we developed various machine learning methods to improve the quality of the SAR models for a given target by taking into account activity information from other targets. These methods include approaches based on semi-supervised learning that deliberately incorporate selected unlabeled compounds while training the SAR models, approaches based on multi-task learning that attempt to improve the quality of the SAR model by transferring information learned from different targets, and approaches based on multi-ranking that utilize the SAR models of different targets by relying on classifier ensembles. The comprehensive experimental evaluation of these methods on a large dataset of 117 protein targets have shown that substantial performance gains can be obtained as long as the set of targets from which activity information is utilized is properly selected. Among the methods developed, approaches based on multi-task learning and multi-ranking achieve the best overall performance, resulting in a 7.0%–7.2% average improvement over the performance achieved by the standard approaches for building SAR models that rely only on the activity information of the target under consideration. Moreover, these methods by selecting the targets from which activity information will be utilized from the entire dataset, they outperform approaches based on chemogenomics that utilize activity information of protein targets that belong to the same family as that under

consideration.

Chapter 9

Improved Machine Learning Models for Predicting Selective Compounds

The identification of small potent compounds that selectively bind to the target under consideration with high affinities is a critical step towards successful drug discovery. However, there still lacks efficient and accurate computational methods to predict compound selectivity properties. In this chapter, we propose a set of machine learning methods to do compound selectivity prediction. In particular, we propose a novel cascaded learning method and a multi-task learning method. The cascaded method decomposes the selectivity prediction into two steps, one model for each step, so as to effectively filter out non-selective compounds. The multi-task method incorporates both activity and selectivity models into one multi-task model so as to better differentiate compound selectivity properties. We conducted a comprehensive set of experiments and compared the results with other conventional selectivity prediction methods, and our results demonstrated that the cascaded and multi-task methods significantly improve the selectivity prediction performance.

9.1 Introduction

Experimental determination of compound selectivity usually takes place during the later stages of the drug discovery process. Selectivity test can include binding assays or clinical trials [132]. The problem with such an approach is that it defers selectivity assessment to the later stages, which if fails, then significant investments in time and resources get wasted. For this reason, it is

highly desirable to have inexpensive and accurate computational methods to predict compound selectivity at earlier stages in the drug discovery process.

Existing computational methods for building SSR models fall into two general classes. The first contains methods that determine selectivity by using SAR models, and the second contains methods that build a selectivity model by considering only the target of interest. The disadvantage of the first class of methods is that they do not have a good mechanism to effectively differentiate selective active compounds from non-selective active compounds. The disadvantage of the second class of methods is that they largely ignore a rich source of information from multiple other proteins, which if properly explored, could lead to more realistic and accurate selectivity models.

In this chapter we develop two classes of machine learning methods for building SSR models. The first class of methods, referred to as cascaded SSR, builds on previously developed techniques and incorporates a pair of models on two levels. The level one is a standard SAR model which identifies the compounds that bind to the target regardless of their selectivity. The level two is a model that further screens the compounds identified by the level-one model in order to identify only the subset that binds selectively to the target and not to the other proteins. Such methods exhibit a cascaded architecture and by decoupling the requirements of accuracy and selectivity, the respective learning tasks are more focused and easier to learn so as to increase the likelihood of developing accurate models. The second class of methods, referred to as multi-task SSR, incorporates information from multiple targets and multiple prediction tasks, and builds a multi-task SSR model. The key insight is that compound activity/selectivity properties for other proteins can be utilized when building a SSR model for the target of interest. These methods treat activity and selectivity prediction as two different yet related tasks. For the target of interest and multiple other proteins, their SAR and SSR tasks are tied together into one single multi-task model. During model training, the SAR and SSR tasks are learned simultaneously with useful information implicitly transferred across one another, and the compound selectivity against multiple proteins is better captured within the model.

We conducted a comprehensive set of experiments to assess the performance of these methods and compare them with other previously proposed state-of-the-art methods. A unique feature of our evaluation is that unlike previous studies that utilized a very small number of test

sets, we constructed datasets derived from publicly available resources (ChEMBL[1]) that collectively contained 135 individual SSR prediction tasks. Our experimental evaluations show that the proposed methods outperform those developed previously and that the approach based on multi-task learning performs substantially better than all the other approaches.

9.2 Methods

The methods that we developed for building SSR models are based on machine learning techniques. Within the context of these methods, there are two approaches that can be used to build SSR models. The first approach is for target p_i and each target $p_j \in P_i$ to build a regression model for predicting the binding affinity of a compound (e.g., IC_{50}) for that target. Then a compound c will be predicted as selective if the two conditions of Equation 7.3 are satisfied by the predicted binding affinities. The second approach is to build a classification model that is designed to directly predict if a compound is selective for p_i without first predicting the compound's binding affinities.

Even though the available training data (i.e., compounds with known binding affinities and their labels according to Equation 7.3) can support both of these approaches, the methods developed and presented in this work are based on the second approach. Specifically, we developed methods that employ neural networks as the underlying machine learning mechanism and determine the selectivity of a compound by building different types of binary or multi-class classification models.

The use of classification- over regression-based approaches was motivated by earlier research for the problem of building SAR models, which showed that predicting a compound's binding affinity is considerably harder than that of predicting if a compound is active or not[133].

9.2.1 Baseline SSR Models

Given a target p_i and a challenge set P_i, the compounds for which the activity information with respect to p_i is known belong to one of three sets: S_i^+, S_i^-, and C_i^-. From these sets, three different SSR classification models can potentially be learned using: (i) S_i^+ vs C_i^-, (ii) S_i^+ vs S_i^-, and (iii) S_i^+ vs $S_i^- \cup C_i^-$. These models share the same positive class (first set of compounds,

[1] http://www.ebi.ac.uk/chembl/

i.e., S_i^+) but differ on the compounds that they use to define the negative class (second set of compounds).

The first model, by ignoring the non-selective active compounds (S_i^-) can potentially learn a model that differentiates between actives and inactives, irrespective of whether the active compounds are selective or not. The second model, by ignoring the inactive compounds (C_i^-) can potentially learn a model that predicts as selective compounds that may not even be active against the target under consideration. For these reasons, we did not investigate these models any further but instead used the third model to define a baseline SSR model that will be denoted by SSR_{base}.

SSR_{base} method constructs the SSR model by treating both the inactive and non-selective active compounds as negative training instances, thus allowing it to focus on the selective active compounds while taking into account the other two groups of compounds. A potential limitation of this model is that depending on the relative size of the S_i^- and C_i^- sets, the model learned may be more influenced by one set of compounds. In particular, since in most cases $|C_i^-| > |S_i^-|$, the resulting model may have similar characteristics to the model learned using only C_i^- as the negative class. To overcome this problem, while constructing the negative class, an equal number of compounds from S_i^- and C_i^- were selected. The total number of compounds that are selected to form the negative class was set to be equal to the number of compounds in the positive class ($|S_i^+|$).

9.2.2 Cascaded SSR Models

The SSR_{base} described in Section 9.2.1 tries to build a model that can achieve two things at the same time: learn which compounds are both active and selective. This is significantly harder than trying to learn a single thing at a time, and as such it may lead to poor classification performance. In order to address this shortcoming, we developed a *cascaded* SSR model that takes into account all the compounds (selectives, non-selectives, and inactives) and builds models such that each model is designed to learn one single task.

For a target p_i and a challenge set P_i, the cascaded SSR model consists of two levels. The model on level one is a normal SAR model that tries to differentiate between active and inactive compounds, and the model on level two is a model that tries to differentiate between selective and non-selective compounds. The level-one model serves as a filter for the level-two model so as to filter out those compounds that are not likely to be even active. During

prediction, compounds are first classified by the level-one model, and only those compounds whose prediction value is above a certain threshold, referred to as the *minactivity* threshold, go through the level-two SSR model. Only compounds classified as positive by the level-two SSR model will be considered as selective. This two-level cascaded SSR model is refereed to as SSR_c.

The level-one model is trained using C_i^+ and C_i^- as positive and negative training instances, respectively, and is identical to p_i's SAR model. The level-two model can be trained using S_i^+ and S_i^- as positive and negative training instances, respectively, as it will be used to classify compounds that were predicted as active by the level-one model. However, the overall performance of the SSR_c model can potentially be improved if the SSR_base model described in Section 9.2.1 is used as the level-two model. This is because the SSR_base model also takes into account the inactive compounds while learning to identify selective compounds and as such it can be used as an additional filter to eliminate inactive compounds that were predicted incorrectly by the level-one model.

Note that even though the cascaded SSR_c model is similar in spirit to the two-step approach proposed by Wassermann *et al* [86], it differs in two important ways. First, instead of sending a constant number of the highest ranked compounds (as predicted by the level-one model) to the level-two model, SSR_c uses the *minactivity* threshold to determine the compounds that will be *routed* to the level-two model. Second, instead of using only the S_i^- compounds as the negative class of the level-two model, SSR_c uses the compounds in $S_i^- \cup C_i^-$ as the corresponding negative class. As the experiments presented in Section 9.4.4 show, this change leads to better performance.

9.2.3 Multi-Task SSR Models

Both the baseline and the cascaded SSR models take into account the labels of the training compounds (i.e., selective, non-selective, active, and inactive) as they were determined for the target under consideration (p_i). However, important information can also be derived by taking into account their labels as they were determined for the targets in the challenge set (P_i). For example, if a compound is active for p_i and it is inactive for all the targets in P_i, then there is a higher probability that the compound is also selective for p_i. Similarly, if a compound is selective for one target in P_i, then by definition, this compound is non-selective for p_i.

Motivated by this observation, we developed another model that in addition to the activity

and selectivity information for p_i, it also incorporates the activity and selectivity information for the targets in the challenge set P_i. This additional information is typically not available for the compounds whose selectivity needs to be determined but also needs to be predicted in the course of predicting the compounds' selectivity. Since this model relies on models built to predict related tasks, it falls under the general class of multi-task learning models, and we will refer to this model as SSR_{mt}.

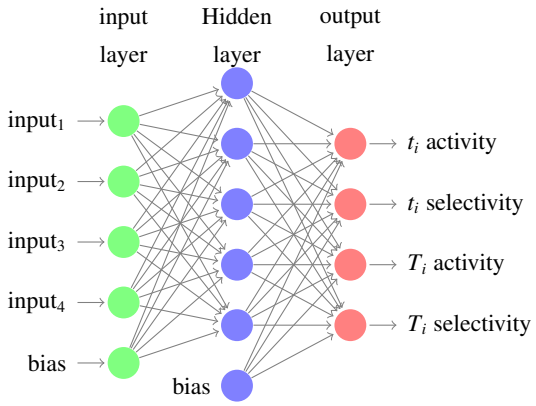


Figure 9.1: A multi-task neural network for target p_i and challenge set P_i.

The SSR_{mt} model extends the model used by the baseline SSR model (Section 9.2.1) by learning compound activity and compound selectivity together. It incorporates these two different learning tasks into a single model so as to facilitate transfer of information during the training of the different models. The learning with information transfer is done by using the neural network model shown in Figure 9.1 which has two pairs of outputs. The first pair corresponds to the activity and selectivity for p_i, whereas the second pair corresponds to the activity and selectivity for P_i (the compound selectivity for each target $p_j \in P_i$ was determined using $\{p_i\}$ as the challenge set). The inputs to this neural network are the various features that describe the chemical structure of the compounds. Each training compound has four labels (one for each output) and during training, the various model parameters are estimated so that to minimize a mean-square-error (MSE) loss function (described in Section 9.3.3) between the predicted and actual four labels at the output layer. The prediction of a compound whose selectivity for p_i needs to be determined is given by the output associated with p_i's selectivity. This model utilizes the same hidden layer to simultaneously learn how to predict the four different tasks (i.e.,

activity and selectivity for p_i and P_i) and as such it can facilitate better information transfer across the different tasks during the model's training stage.

Note that the four labels for each training instance are not independent. For example, if selectivity for p_i is positive (i.e., selective for p_i), then selectivity for any other p_j has to be negative (i.e., a compound cannot be selective for two targets under consideration). Also if activity for p_i is negative, than selectivity for p_i has to be negative (selective compounds have to be active first). We do not explicitly model such dependencies through loss function but rely on the NN system and the learning process to implicitly incorporate such constraints from training instances.

9.2.4 Three-Way SSR Models

The performance of the SSR models described in the previous sections was also compared against the performance of another type of model that has been proposed in the past.

This model is the 3-way classification approach developed by Wassermann *et al*. [86] that operates as follows. For target p_i and its challenge set P_i, it builds three one-vs-rest binary classification models for each one of the selective (S_i^+), non-selective (S_i^-), and inactive (C_i^-) sets of compounds, respectively. During prediction, a compound c is predicted by each one of the three models, leading to three predicted values $f_{S_i^+}(c)$, $f_{S_i^-}(c)$, and $f_{C_i^-}(c)$. A compound is considered to be selective if $f_{S_i^+}(c) = \max(f_{S_i^+}(c), f_{S_i^-}(c), f_{C_i^-}(c))$. Also, if a degree of selectivity is required (i.e., in order to rank a set of predicted compounds), then c's degree of selectivity is given by $f_{S_i^+}(c) - \max(f_{S_i^-}(c), f_{C_i^-}(c))$. The underlying idea of this 3-way classification method is to model different classes separately and to decide the class of a new instance based on how differently it is classified by different models. We will denote this SSR model as SSR_{3way}.

9.2.5 Cross-SAR SSR Models

Another model was motivated by approaches used within the pharmaceutical industry in which the selectivity of a compound c_i against target p_i is determined by comparing the output on c_i from p_i's SAR model against that of the SAR model for each one of the targets in P_i. Specifically, if $f_{t_i}(c_i)$ is the prediction of p_i's SAR model on c_i and $\{f_{t_j}(c_i)|p_j \in P_i\}$ are the predictions of the SAR models for the targets in P_i on c_i, then the extend to which c_i is selective for p_i against P_i is given by $f_{t_i}(c_i) - \max_{t_j}(f_{t_j}(c_i))$. We will denote this SSR model as SSR_{xSAR}.

9.2.6 Three-Class SSR Models

Compound selectivity prediction can also be viewed as a multi-class classification problem, in which each compound c_i has three binary class labels, that is, selectivity, non-selectivity and inactivity against a target p_i. For each target p_i, a three-class classifier is built from its own compounds. Then a compound is predicted by such a multi-class model as one of the three classes. Figure 9.2 shows a three-class neural network. The difference between the three-class neural network classifier and the multi-task neural network classifier as in Figure 9.2 is that the compound activity label against the challenge set P_i is not included.

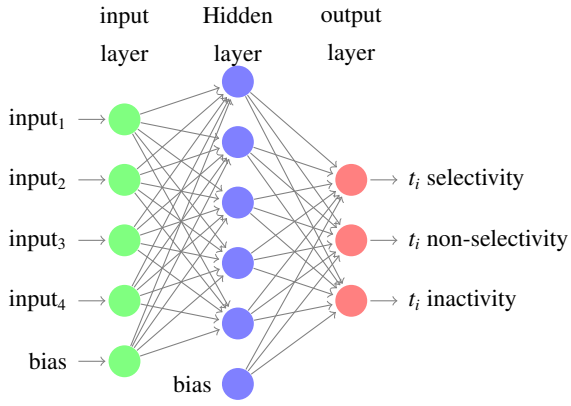


Figure 9.2: A three-class neural network for target p_i.

9.3 Materials

9.3.1 Datasets

We evaluated the performance of the various SSR models on a set of protein targets and their ligands that are extracted from ChEMBL, which is a database of molecular targets and their published assays with bioactive drug-like small molecules. We first selected an initial set of molecular targets and their corresponding ligands from ChEMBL based on the following criteria:

- The target is a single protein.
- The assay for the target is a binding assay.

- For each target p_i, there are at least 20 active compounds.

These criteria ensure that the binding affinities measure how well a compound binds to a single target and also there are sufficient compounds to learn a model. From this initial set of targets, we eliminated those targets if they satisfy any of the following criteria:

- The target does not share any of its active compounds with other targets in the initial set of targets.
- The target has less than 10 selective compounds against any single target in the initial set.

The first condition eliminates targets for which we cannot access if their active compounds are selective or not, whereas the second condition is designed to keep the targets that contain a sufficient number of selective compounds in order to learn a SSR model. These filtering steps resulted in a dataset with 98 protein targets. For each target p_i of these targets, we used all of its known active compounds and constructed an equal-size set of inactive compounds. The inactive compounds were selected from p_i's known inactive compounds if sufficient, otherwise, they were randomly selected from ChEMBL that always show extremely low binding affinities against arbitrary targets. Such compounds are very likely to be truly inactive against p_i even no experimental results are available, because they exhibit some strong drug-unlike properties evidenced by other targets. Note that during the random compound selection for p_i, we guaranteed that the randomly-selected inactive compounds are not in C_j^+ or C_j^- of any other targets (i.e., $i \neq j$). This is done in order to avoid the situation in which a randomly selected compound is determined to be selective for some targets.

Using these 98 targets, we constructed two datasets for experimental testing. The first dataset, referred to as DS1, contains 116 individual SSR prediction tasks involving a single target p_i as the target of interest and another single target p_j as its challenge set (i.e., $P_i = \{p_j\}$). These 116 SSR prediction tasks were identified by considering all possible (i.e., 98×97) SSR prediction tasks of this type and then selecting only those for which (i) targets p_i and p_j have some common active compounds (i.e., those compounds are active for both p_i and p_j) and (ii) when p_j is used as the sole member of p_i's challenge set, the resulting SSR prediction task results in at least 10 selective compounds for p_i. Both of these filtering steps are essential to ensure that there are a sufficiently large number of training compounds to accurately learn and assess the selectivity of the target of interest. In these 116 SSR prediction tasks, the average number of active and selective compounds for the target of interest is 172 and 26, respectively.

Note that each target p_i can potentially be the target of interest in multiple SSR prediction tasks and that a compound c may have different selectivity properties for p_i when different P_is are considered.

The second dataset, referred to as DS2, contains 19 individual SSR prediction tasks involving a single target p_i as the target of interest and multiple targets in its challenge set P_i. The 19 prediction tasks were identified according to the criteria that (i) target p_i and each $p_j \in P_i$ share common active compounds, (ii) $|P_i| \geq 2$ and (iii) there are at least 10 selective compounds for p_i against P_i determined based on Equation 7.3. These criteria result in on average 3.9 targets in each challenge set, and the average number of active and selective compounds for the target of interest is 198 and 27, respectively.

The first dataset is constructed so as to maximize the number of selective compounds for each p_i to train a reliable model. This is also a common practice in other selectivity learning and dataset construction exercise [86, 134] and in real experimental settings. Meanwhile, it maximizes the number of interested targets to test for any statistically significant conclusions. The second dataset is constructed to test the generalizability of SSR models. Additional details on the targets, compounds, and the two datasets are available at [2] . Figure 9.3 shows the dataset DS1 as well as DS2.

9.3.2 Compound Representations

We generated 2048-bit binary Chemaxon compound descriptors[3] for all the compounds extracted as in 9.3.1. Then we applied a PCA-based dimension reduction method such that the 2048 dimensions are reduced to 1000 dimensions. Each compound is then represented by such a 1000-dimension feature vector and thus a `NN` with 1000 input nodes can be trained on such compound representations. We used the Chemaxon software generatemd to generate initial descriptors, and a Matlab dimension reduction toolbox[4] with PCA option to reduce descriptor dimensions.

Note that chemical compounds can be represented by different fingerprints [135]. However, since our study does not aim to evaluate the performance of different fingerprints for compound selectivity, we only applied Chemaxon compound descriptors because it is one of the most

[2] http://www-users.cs.umn.edu/~xning/selectivity/

[3] http://www.chemaxon.com/

[4] http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html

popular choices. Dimensionality reduced is performed since the NN may suffer from the curse of dimensionality [136] if high-dimension inputs are encountered.

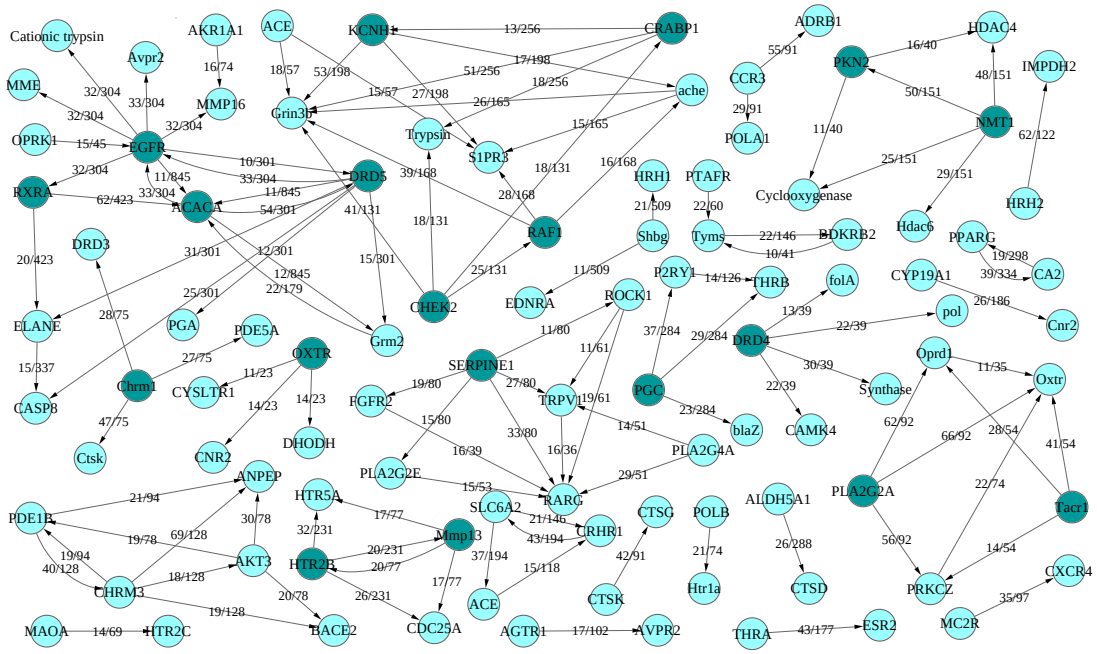


Figure 9.3: The nodes in the graph represent the targets in DS1. The directed edge from target A to target B with a label x/y represents that target A has y active compounds, out of which x compounds are selective for target A against target B. The dark nodes represent the targets that are selected as targets of interest into DS2.

9.3.3 Neural Networks

We used the publicly available neural network software FANN[5] for our neural network implementation. FANN implements fast multi-layer artificial neural networks with support for both fully connected and sparsely connected networks.

NN Training & Parameters

We used Back-Propagation (BP) algorithm for NN training [137]. BP requires a set of learning parameters, and in the following experiments, we specified such learning parameters as follows:

[5] http://leenissen.dk/fann/

learning rate 0.005, maximum number of iterations 100000, steepness 1.0 on hidden layers and output layer, and momentum 0.001.

In the following experiments, we denoted *minMSE* as the desired MSE such that once the training error reaches *minMSE*, the `NN` training process is terminated. Thus, *minMSE* is one of the `NN` training termination conditions, in addition to maximum number of training iterations.

We did a preliminary study on the range of optimal number of hidden layers and optimal number of hidden neurons by performing a grid search on such numbers using SSR_{base} models with 1 and 2 hidden layers, and 64, 128, 256 and 512 hidden neurons on each layer, respectively. The results demonstrated that only one hidden layer suffices to learn a good model. All the experiments that are reported below utilized a neural network with a single hidden layer. Experiments with additional hidden layers did not lead to any improvements so they are not reported.

9.4 Experimental Results

In this section, we present the results for the selectivity studies. We present the detailed results for the first dataset, in which each challenge set has only one target and each target may have multiple challenge sets. In the first dataset, we simply refer to p_j as the challenge target of the interested target p_i since $P_i = \{p_j\}$. In the end, we present an overall performance summary for the second dataset, in which each challenge set has multiple targets and each target has only one challenge set, since the results for the second dataset show a similar trend as in the first dataset.

9.4.1 Compound Similarities

First of all, we conducted a test on compound similarity in the first dataset so as to access the quality of our dataset. Particularly we tested the compound similarities among selectives against selectives, actives against actives, actives against selectives and actives against nonselectives, respectively, using Tanimoto coefficient, which is defined as follows.

$$\text{sim}_c(c_x, c_y) = \frac{\sum_k c_{x,k} c_{y,k}}{\sum_k c_{x,k}^2 + \sum_k c_{y,k}^2 - \sum_k c_{x,k} c_{y,k}}, \tag{9.1}$$

where c_x and c_y are the fixed-length feature vectors for two compounds, k goes over all the dimensions of the feature vector space and $c_{x,k}$ is the value on the k-th dimension. Note that after

dimension reduction, compound feature vectors may have negative values, and thus Tanimoto coefficient can be negative in this case. However, their relative comparison still reflects the differentiation cross compound groups.

The detailed results are available in supplementary materials. Some statistics is available in Table 9.1.

Table 9.1: Compound similarity

sim_s2s	sim_ns2ns	sim_s2ns	sim_a2a	sim_a2s	sim_a2ns
0.570	0.385	0.343	0.371	0.372	0.357

In this table, for each target p_i of interest in the first dataset, "sim_s2s" is the average similarity between selective compounds, "sim_ns2ns" is the average similarity between nonselective compounds, "sim_s2ns" is the average similarity between selective compounds and nonselective compounds, "sim_a2a" is the average compound similarity between active compounds, "sim_a2s" is the average compound similarity between active compounds and selective compounds, "sim_a2ns" is the average similarity between active compounds and nonselective compounds.

We also tried different compound feature vectors/fingerprints to calculate compound similarities, and the trend remains the same, that is, selective compounds are more similar to other selective compounds, nonselective compounds are more similar to other nonselective compounds, and on average active compounds are more similar to selective compounds than to nonselective compounds. This trend of compound similarities indicates that in general, compound selectivity problem is not trivial since we try to identify a small subset of active compounds, which are similar to other active compounds to a large extent. Also the compounds are diverse enough since they have low similarities, and therefore it is a hard set of compounds and they are suitable for testing the proposed methods.

9.4.2 Effects of Dimension Reduction

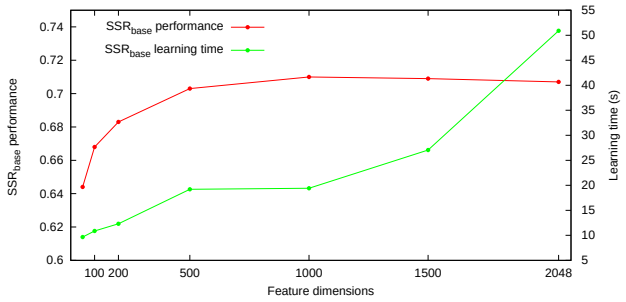


Figure 9.4: Effects of dimension reduction for DS1.

Figure 9.4 shows how the PCA-based dimensionality reduction impacts the performance of the SSR_{base} model. The metrics plotted correspond to the average F_1^b values and model learning time over the 116 SSR tasks of DS1. This plot was obtained by reducing the dimensions of the original binary fingerprints from 2048 to 50, 100, 200, 500, 1000 and 1500, and then trained SSR_{base} models on respective reduced features. For each number of dimensions the reported results correspond to the best prediction performance over all learning parameters (i.e., learning rate, steepness, etc, and number of hidden neurons, as specified in Section 9.3.3).

These results indicate that as the number of dimensions increases, the accuracy of the SSR_{base} model improves. However, the best prediction performance is achieved at 1000 dimensions. Moreover, when 1000 dimensions are used, the amount of time required to learn the `NN` models is about 2/5 of that required when no dimensionality reduction is performed. For these reasons, in all of our subsequent experiments, we used the reduced 1000-dimension features to represent compounds.

9.4.3 Results for Baseline SSR_{base} Models

Table 9.2: SSR_{base} Average F_1^b Scores.

minMSE	DS1			DS2		
	32	64	128	32	64	128
0.01	0.700	0.701	0.699	0.649	0.638	0.646
0.03	0.700	0.704	0.705	0.652	0.641	0.651
0.05	0.707	**0.710**	0.707	0.658	0.654	0.654
0.07	0.704	0.708	0.702	0.655	**0.657**	0.639
0.10	0.704	0.706	0.681	0.648	0.643	0.584

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 32, 64, etc, corresponds to the results using 32, 64, etc, hidden neurons in NNs. The **bold** numbers indicate the best average performance over all *minMSE* values and numbers of hidden neurons for all the targets. There is only 1 hidden layer in all the NNs.

Table 9.2 shows the performance achieved by the SSR_{base} model on the DS1 and DS2 datasets for different number of hidden neurons and different *minMSE* values for stopping `NN` training. The best performance is achieved with 64 hidden neurons and *minMSE* values of 0.05 for DS1 and 0.07 for DS2. These results also show that when *minMSE* decreases, the models tend to overfit the training data and when *minMSE* increases, the models tend to underfit the training data. Similar trends can be observed when the number of hidden neurons increases or decreases.

A promising observation is that the overall best performance of 0.710 for DS1 and 0.657 for DS2 is substantially better than that of a random prediction, indicating that machine learning methods can be utilized to build SSR models. Also the performance on DS2 is lower than that achieved on DS1, indicating that learning SSR models when the challenge set contains multiple targets is considerably harder.

9.4.4 Results for Cascaded SSR_c Models

Recall from Section 9.2.2 that SSR_c uses a SAR model (level-one model) to identify the compounds that have a predicted activity value greater than or equal to the *minactivity* threshold and

then uses a SSR_{base} model (level-two model) to predict which of those compounds are selective for the target of interest. For this reason, our experimental evaluation initially focuses on assessing the performance of the SAR models themselves in order to determine their optimal set of model parameters, and then on the evaluation of model sensitivity to the *minactivity* threshold parameter.

Table 9.3: SAR Average F_1^b Scores.

minMSE	DS1			DS2		
	64	128	256	128	256	512
0.001	0.906	0.906	0.904	0.912	0.913	0.901
0.003	0.904	**0.906**	0.906	0.914	0.917	0.907
0.005	0.904	0.905	0.904	0.913	**0.918**	0.910
0.007	0.902	0.903	0.903	0.916	0.910	0.906
0.010	0.901	0.901	0.899	0.910	0.911	0.899

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 64, 128, etc, corresponds to the results using 64, 128, etc, hidden neurons in NNs. The **bold** numbers indicate the best average performance over all *minMSE* values and numbers of hidden neurons for all the targets. There is only 1 hidden layer in all the NNs.

Table 9.3 shows the performance of the SAR models for the two datasets. The best average performance for DS1 (0.906) is achieved for 128 hidden neurons and a *minMSE* value of 0.003, whereas the best average performance for DS2 (0.918) is achieved for 256 hidden neurons and a *minMSE* value of 0.005. These high F_1 scores, which result from high values of the underlying precision and recalls measures, are encouraging for two reasons. First, the compounds that will be filtered out will be predominately inactives (high precision), which makes the prediction task of the level-two model easier as it does not need to consider a large number of inactive compounds. Second, most of the selective compounds will be passed through to the level-two model (high recall), which ensures that most of the selective compounds will be considered (i.e., asked to be predicted) by that model. This is important as the selectivity determination is done only by the level-two model for only those compounds that pass the *minactivity*-threshold filter of the level-one model.

Table 9.4: Cascaded Model Average F_1^b Scores.

scheme	dataset	*minactivity*				
		0.3	0.4	0.5	0.6	0.7
SSR_c	DS1	0.727	**0.729**	0.728	0.727	0.725
	DS2	0.671	**0.676**	0.674	0.673	0.671
Wass	DS1	0.721	**0.723**	0.723	0.723	0.722
	DS2	0.631	**0.631**	0.631	0.630	0.628

The rows corresponding to SSR_c show the results when the level-two model is trained using S_i^+ as positive training instances and $S_i^- \bigcup C_i^-$ as negative training instances (i.e., SSR_{base}). The rows corresponding to Wass show the results when the level-two model is trained using S_i^+ as positive training instances and S_i^- as negative training instances [86]. Rows for DS1 and DS2 show the results for dataset one and dataset two, respectively. Each column corresponds to the results with corresponding *minactivity* threshold used. The **bold** numbers indicate the best average performance over all *minactivity* thresholds. For dataset one, level-one SAR models have 128 hidden nodes and *minMSE* 0.003, and level-two SSR_{base} models have 64 hidden nodes and *minMSE* 0.05 for both SSR_c and Wass models. For dataset two, level-one SAR models have 256 hidden nodes and *minMSE* 0.005, and level-two SSR_{base} models have 64 hidden nodes and *minMSE* 0.07 for both SSR_c and Wass methods.

The first two rows of Table 9.4 show the performance achieved by the SSR_c models for different values of the *minactivity* threshold parameter. In these experiments, the level-one models correspond to the SAR model with the optimal parameter combination that achieved the best results in Table 9.3 (i.e., DS1: 128 hidden neurons and *minMSE* 0.003; DS2: 256 hidden neurons and *minMSE* 0.005) and the level-two models correspond to the SSR_{base} model with the optimal parameter combination that achieved the best results in Table 9.2 (i.e., DS1: 64 hidden neurons and *minMSE* 0.05; DS2: 64 hidden neurons and *minMSE* 0.07). The overall best average performance achieved by SSR_c is 0.729 and 0.679 for the DS1 and DS2 datasets, respectively and occurs when the *minactivity* threshold value is 0.4. Also, these results show that as *minactivity* changes, the performance of the resulting SSR_c models changes as well. However, these results show that for a relatively large number of reasonable *minactivity* threshold values, the overall performance remains relatively similar. Of course, if *minactivity* is too small or too large,

then the resulting model either becomes identical to SSR_{base} or may fail to identify selective compounds due to low recall.

The second two rows of Table 9.4 show the performance of a cascaded SSR model in which the level-two model uses only the nonselective compounds as the negative class. This is similar to the model used by the two-step approach developed by Wassermann *et al* [86]. Note that these results were obtained using the same level-one model as that used by SSR_c and the same `NN` model/learning parameters used by SSR_c. The best average performance achieved by this alternate approach is 0.723 and 0.631 for DS1 and DS2, respectively; both of which are worse than those achieved by SSR_c. These results indicate that taking into account the inactive compounds in the level-two model leads to better SSR prediction results.

9.4.5 Results for Multi-Task SSR_{mt} Models

Table 9.5: SSR_{mt} Average F_1^b Scores.

minMSE	DS1			DS2		
	128	256	512	64	128	256
0.01	0.484	0.426	0.414	0.590	0.611	0.615
0.03	0.753	0.757	0.756	0.649	0.662	0.657
0.05	0.756	**0.759**	0.754	0.667	0.664	0.671
0.07	0.747	0.747	0.746	0.672	**0.681**	0.660
0.10	0.738	0.735	0.737	0.662	0.671	0.656

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 64, 128, etc, corresponds to the results using 64, 128, etc, hidden neurons in NNs. The **bold** numbers indicate the best average performance over all *minMSE* values and numbers of hidden neurons for all the targets. There is only 1 hidden layer in all the NNs.

Table 9.5 shows the performance achieved by the SSR_{mt} model for different number of hidden neurons and *minMSE* values. The best average performance for DS1 (0.759) happens for 256 hidden neurons and a *minMSE* value of 0.05; whereas the best performance for DS2 (0.681) happens for 128 hidden neurons and a *minMSE* value of 0.07. The performance characteristics

of the SSR_{mt} model as a function of the number of hidden neurons and the *minMSE* value are similar to those observed earlier for the SSR_{base} model. As the number of hidden neurons decreases/increases (or the *minMSE* values increases/decreases) the performance of the resulting model degrades due to under- and over-fitting.

9.4.6 Results for Three-Way Models

Table 9.6: SSR_{3way} Average F_1^b Scores.

minMSE	DS1			DS2		
	32	64	128	32	64	128
0.01	0.707	0.711	0.712	0.640	0.649	0.637
0.03	0.707	0.712	0.707	0.653	**0.664**	0.643
0.05	0.718	**0.722**	0.713	0.636	0.650	0.616
0.07	0.721	0.717	0.697	0.626	0.641	0.563
0.10	0.711	0.698	0.641	0.610	0.582	0.508

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 32, 64, etc, corresponds to the results using 32, 64, etc, hidden neurons in NNs. The **bold** numbers indicate the best average performance over all *minMSE* values and numbers of hidden neurons for all the targets. There is only 1 hidden layer in all the NNs.

Table 9.6 shows the performance achieved by the SSR_{3way} models for different number of hidden neurons and *minMSE* values. These results were obtained by using the same set of model and learning parameters (i.e., number of hidden neurons and *minMSE* value) for each one of the three binary models involved (i.e, S_i^+ vs the rest, S_i^- vs the rest, and C_i^- vs the rest). The best average performance for DS1 (0.722) happens for 64 hidden neurons and a *minMSE* value of 0.05; whereas the best performance for DS2 (0.664) happens for 64 hidden neurons and a *minMSE* value of 0.03.

9.4.7 Results for Cross-SSR SSR_{xSAR} Models

Table 9.7: SSR_{xSAR} Average F_1^b Scores.

minMSE	DS1			DS2		
	32	64	128	32	64	128
0.01	0.710	0.705	0.700	0.663	0.661	0.662
0.03	**0.715**	0.704	0.700	0.663	0.662	0.661
0.05	0.707	0.690	0.677	0.662	0.661	0.662
0.07	0.696	0.673	0.649	0.664	**0.664**	0.663
0.10	0.656	0.638	0.636	0.662	0.664	0.663

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 32, 64, etc, corresponds to the results using 32, 64, etc, hidden neurons in NNs of P_i. The **bold** numbers indicate the best average performance over all *minMSE* values and numbers of hidden neurons for all the targets. There is only 1 hidden layer in all the NNs. The SSR_{base} models are the best ones as in Table 9.2.

Table 9.7 shows the performance of SSR_{xSAR} models. In the SSR_{xSAR} models for p_i, p_i's only SAR model is its best baseline SAR model as identified from Table 9.2. The performance shown in Table 9.7 are achieved by using each p_i's best baseline SAR model and different number of hidden neurons from P_i's SAR models and *minMSE* values. The best average performance for DS1 (0.715) happens for 32 hidden neurons in P_i's `NN` and a *minMSE* value of 0.03; whereas the best performance for DS2 (0.664) happens for 64 hidden neurons in P_i's `NN` and a *minMSE* value of 0.07.

9.4.8 Results for Multi-Class SSR_{3class} Models

Table 9.8: SSR_{3class} Average F_1^b Scores.

minMSE	DS1			DS2		
	64	128	512	32	64	128
0.01	0.708	0.707	0.710	0.639	0.617	0.612
0.03	0.734	0.733	0.737	0.664	**0.670**	0.657
0.05	0.740	0.739	0.736	0.661	0.667	0.650
0.07	0.739	**0.741**	0.741	0.659	0.660	0.654
0.10	0.737	0.737	0.737	0.644	0.647	0.650

minMSE is the minimum MSE within stop criteria for model training. Columns under DS1 and DS2 correspond to the results for dataset one and dataset two, respectively. Each column under 32, 64, etc, corresponds to the results using 32, 64, etc, hidden neurons in NNs. The **bold** numbers indicate the best average performance over all *minMSE* values and numbers of hidden neurons for all the targets. There is only 1 hidden layer in all the NNs.

Table 9.8 shows the performance of SSR_{3class} models achieved by different number of hidden neurons and *minMSE* values. The best average performance for DS1 (0.741) is achieved by 128 hidden neurons and a *minMSE* value of 0.07; whereas the best performance for DS2 (0.670) is achieved by 64 hidden neurons and a *minMSE* value of 0.03.

9.4.9 Overall Comparison

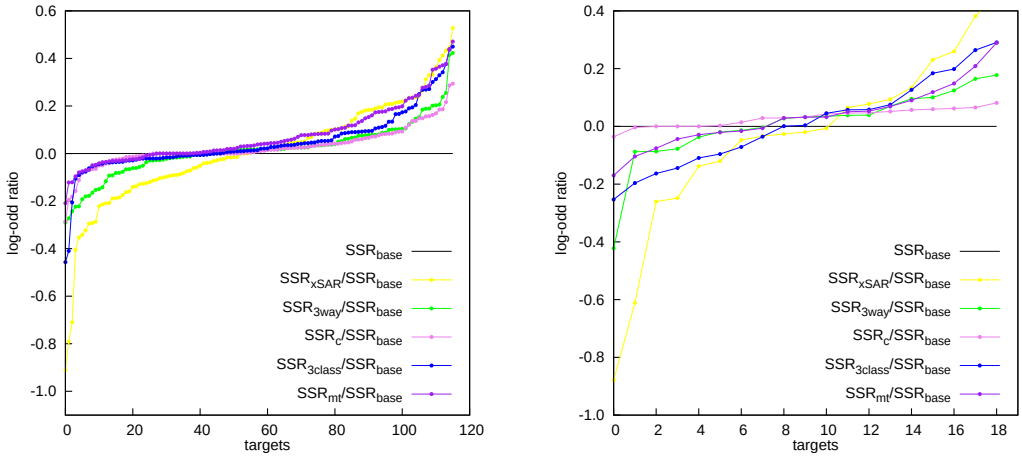
Table 9.9 summarizes the best average F_1^b results achieved from SSR_{base}, SSR_{xSAR}, SSR_{3way}, SSR_c, SSR_{3class} and SSR_{mt} models on the DS1 and DS2 datasets. These results correspond to the bold-faced entries of Tables 9.2, 9.7, 9.6, 9.4, 9.8 and 9.5, respectively. In addition, for each scheme other than SSR_{base}, the rows labeled "#imprv" show the number of prediction tasks for which the corresponding scheme outperforms the SSR_{base} models. Similarly, the rows labeled "imprv" show the average performance improvement achieved by that scheme of the SSR_{base} models. The performance improvement is calculated as the geometric mean of pairwise performance improvement over SSR_{base}.

Table 9.9: SSR Model Performance Comparison.

dataset	pfm/imprv	scheme					
		SSR_{base}	SSR_{xSAR}	SSR_{3way}	SSR_{c}	SSR_{3class}	SSR_{mt}
DS1	best	0.710	0.715	0.722	0.729	0.741	0.759
	#imprv	-	61	72	72	71	79
	imprv	-	0.6%	1.6%	2.6%	4.0%	7.5%
DS2	best	0.657	0.664	0.664	0.676	0.670	0.681
	#imprv	-	11	11	15	11	11
	imprv	-	2.9%	0.8%	3.2%	2.9%	3.5%

The rows labeled best correspond to the best performance from the corresponding SSR model given *minMSE* and number of hidden neurons fixed for all prediction tasks. The rows labeled #imprv present the number of prediction tasks for each the corresponding SSR method performs better than baseline SSR_{base}. The rows labeled imprv present the geometric mean of pairwise performance improvement over baseline SSR_{base} model.

For both the datasets, the SSR_{xSAR}, SSR_{3way}, SSR_{c}, SSR_{3class} and SSR_{mt} models outperform the SSR_{base} model. This indicates that by incorporating additional information (i.e., compound activity properties for the target of interest, compound activity and selectivity properties against challenge sets, etc) rather than focusing on selectivity property alone improves selectivity prediction performance. Among the different schemes, the SSR_{mt} models achieve the best SSR prediction results. Their average improvement over SSR_{base} for DS1 and DS2 is 7.5% and 3.5%, respectively. The SSR_{3class} models achieve the second best performance for DS1, which corresponds to an average improvement of 4.0%, and the third best performance for DS2, which corresponds to an average improvement of 2.9%. The SSR_{c} models achieve the third best performance for DS1, which corresponds to an average improvement of 2.6%, and the second best performance for DS2, which corresponds to an average improvement of 3.2%. Finally, even though SSR_{3way} and SSR_{xSAR} improve upon the SSR_{base} model, the gains achieved are rather modest (1.6% for DS1 and 0.8% for DS2 for SSR_{3way}, 0.6% for DS1 for SSR_{xSAR}).



(a) DS1 (b) DS2

Figure 9.5: Pairwise Improvement.

A finner-grain picture of the performance of the different methods on the different SSR prediction tasks involved in DS1 and DS2 is shown in the plots of Figure 9.5. These plots show the log-ratios of the F_1^b scores achieved by each model over that achieved by the baseline model for the 116 SSR tasks of DS1 (Figure 9.5(a)) and the 19 SSR tasks of DS2 (Figure 9.5(b)). For each SSR model, the results in Figure 9.5 are presented in a non-increasing order according to these log-ratios. Figure 9.5 shows that SSR_{mt} leads to higher improvements for more individual SSR prediction tasks than SSR_{3way} and SSR_c, and that SSR_c performs slightly better than SSR_{3way}. SSR_{xSAR} and SSR_{3class} have more dynamic than the other models. The actual number of SSR prediction tasks for which each method outperforms the baseline are shown in the row labeled "#imprv" of Table 9.9.

Table 9.10 presents the paired *t*-test across different SSR methods. It shows that for DS1, SSR_c, SSR_{3class} and SSR_{mt} all outperform SSR_{base} significantly. In addition, SSR_{3class} outperforms other SSR methods significantly except SSR_c, and SSR_{mt} outperforms all the other SSR methods significantly. However, for DS2, the statistical test did not show significant difference among all the SSR methods, even though SSR_{mt} outperforms others in terms of F_1^b.

Comparing the performance across the two datasets we see that the proposed methods are able to achieve considerably better improvements for DS1 than DS2. We believe that this is due

to the fact that the underlying learning problem associated with the prediction tasks in DS2 are harder, since the challenge sets contain more than one target.

Table 9.10: Paired *t*-Test.

dataset	scheme	SSR_{base}	SSR_{xSAR}	SSR_{3way}	SSR_{c}	SSR_{3class}	SSR_{mt}
	SSR_{base}	-/-	7.102e-01/0	6.380e-02/0	1.697e-04/1	7.019e-06/1	1.642e-10/1
	SSR_{xSAR}		-/-	5.766e-01/0	2.169e-01/0	3.402e-02/1	1.719e-04/1
DS1	SSR_{3way}			-/-	4.157e-01/0	6.612e-03/1	2.179e-07/1
(116 tasks)	SSR_{c}				-/-	7.168e-02/0	1.226e-05/1
	SSR_{3class}					-/-	6.237e-06/1
	SSR_{mt}						-/-
	SSR_{base}	-/-	8.697e-01/0	6.896e-01/0	1.356e-03/1	5.840e-01/0	1.171e-01/0
	SSR_{xSAR}		-/-	9.973e-01/0	7.720e-01/0	8.715e-01/0	6.414e-01/0
DS2	SSR_{3way}			-/-	4.705e-01/0	7.926e-01/0	4.188e-01/0
(19 tasks)	SSR_{c}				-/-	7.759e-01/0	7.667e-01/0
	SSR_{3class}					-/-	4.702e-01/0
	SSR_{mt}						-/-

The x/y values correspond to the the p-value (i.e., x) and whether the null hypothesis (i.e., the SSR method of the column performs statistically the same as the SSR method of the row) is rejected (i.e., $y = 1$) or not (i.e., $y = 0$) at the 5% significance level, respectively.

9.5 Discussion & Conclusions

In this chapter, we developed two machine learning methods SSR_{c} and SSR_{mt} for building SSR models, and experimentally evaluated them against the previously developed methods SSR_{base}, SSR_{xSAR}, SSR_{3way} and SSR_{3class}. Our results showed that the SSR_{mt} approaches achieve the best results, substantially outperforming other methods on a large number of different SSR prediction tasks. This multi-task model combines activity and selectivity models for multiple proteins into a single model such that during training, the two models are learned simultaneously, and compound preference over targets is learned and transferred across. This suggests that collectively considering information from multiple targets and also compound activity and selectivity properties for each target benefits selectivity prediction.

Our experiments showed that even though the multi-task learning-based SSR models can achieve good performance for the SSR prediction tasks in which the challenge set contains a single other target, their performance for multi-target challenge sets is considerably lower. This indicates that future research is required for developing methods which better predict SSR with multi-target challenge sets. A potential approach is that SSR_{mt} models can be constructed to have outputs for each of the targets in challenge set such that more information is expected to be learned through the multi-task learning.

We use neural networks as the learning algorithms due to their flexible and adaptive nature for multi-task learning, despite that there exist some other (stronger in general) learning algorithms. Another widely used algorithm for compound classification is Support Vector Machines (`SVM`) [8]. In our primary studies, we conducted experiments using `SVM` with different kernel functions on the reduced 1000-bit compound features and original 2048-bit compound features for the SSR_{base} and SSR_{mt} methods on DS1. Our results demonstrated that in both SSR_{base} and SSR_{mt}, `SVM` did not perform better than `NN` on our dataset, so we did not apply `SVM` for SSR models.

Chapter 10

Conclusion

10.1 Thesis Summary

10.1.1 Recommender Systems

Recommender systems represent a set of computational methods that produce recommendations of interesting entities (e.g., products, friends, etc) from a large collection of such entities by retrieving/filtering/learning information from their own properties (e.g., product attributes, personal profiles, etc) and/or the interactions between these entities and other parties (e.g., user-product ratings, friend-friend trust relations, etc). Recommender systems are particularly important for e-commerce applications, where the overwhelming amount of items makes it extremely difficult for users to manually identify those items that best fit their personal preferences. There are two core tasks for recommender systems. The first is to generate a short ranked list of items that best suit a user's personal tastes. This is referred to as *top-N recommendation*. The second is to predict how a user likes an item in terms of a numerical rating. This is referred to as *rating prediction*.

The top-*N* recommendation and rating prediction problems are typically solved through two different classes of approaches. The first is purely content based and only the intrinsic information of the items and/or the user is used to make recommendations. In the content-based approaches, no relations between multiple users or multiple items are used. The second is via collaborative filtering, which generates recommendations for a user by taking into account the purchases/ratings of other users. Collaborative filtering approaches can be further divided

into neighborhood based and model based, according to the methods used to derive the recommendations. The neighborhood-based methods explicitly construct user or item neighborhoods from the data and use the information from the neighborhood to calculate recommendations. The model-based methods learn a model (e.g., regression model, matrix factorization model) from the data, and recommendations are made from the model. In this thesis, different methods have been developed to address the top-N recommendation and rating prediction problems, respectively.

Sparse Linear Methods for Top-N Recommendation

The state-of-the-art methods for top-N recommendation are based on matrix factorization as they produce high-quality recommendations. However, the recommendation step of matrix factorization models is quite slow. On the other hand, neighborhood-based collaborative filtering methods are significantly faster, but they suffer from low recommendation accuracy as they rely on a pre-determined similarity measure, which does not adapt to the characteristics of the data.

In this thesis, both the quality and the run-time performance of top-N recommender systems have been addressed simultaneously. In particular, a Sparse LInear Method (`SLIM`) [138] have been developed that combines the underlying ideas of both the item-based neighborhood methods and the model-based methods. Similar to item-based methods, `SLIM` generates recommendations for a user by aggregating the items that are similar to those purchased by the user. However, unlike those item-based methods, `SLIM` learns the item-item similarity relations directly from the data, which allows it to generate high-quality top-N recommendations. `SLIM` formulates the learning process as a least-squares error minimization problem with an ℓ_2-norm and an ℓ_1-norm regularizors. The ℓ_2-norm regularization prevents overfitting whereas the ℓ_1-norm regularization leads to a sparse model that contributes to `SLIM`'s low computational complexity for generating top-N recommendations. A comprehensive set of experiments was conducted by comparing `SLIM` with other state-of-the-art top-N recommendation methods. The experiments showed that `SLIM` achieves significant improvements both in recommendation quality and run time performance over the best existing methods.

Sparse Linear Methods for Top-N Recommendation with Side Information

A new sparse linear method [139, 140] has been developed, referred to as `SSLIM`, which not only uses the user-item purchase data, but also utilizes other intrinsic information on users/items (e.g., users' social networks, items' categories, etc). Such information is referred to as side information. The `SSLIM` method takes side information as regularizors for the model learned from `SLIM`. By doing this, the recommendation algorithm is biased towards generating recommendations that conform to side information as well. The experiential results demonstrated that `SSLIM` outperforms or performs comparable to `SLIM`, and outperforms other existing methods that utilize side information.

Multi-task Learning for Rating Predictions

The central hypothesis of collaborative filtering methods has been that a user belongs to one or more like-minded groups, and thus the user's own historical information in conjunction with the historical information from the users in these groups can be used to derive recommendations/rating predictions for that user. In this thesis, it has been investigated if better rating predictions can be achieved by leveraging the collaborative filtering models (not the explicit historical information) of the users from the groups. To this end, a class of collaborative-filtering-based multi-task learning methods for rating prediction has been developed [141].

In the multi-task learning approaches, for each user under consideration (referred to as active user), a group of other users (referred to as related users) is explicitly selected based on their similarities to the active user so as to construct a personalized user group. Then a regression-based collaborative filtering recommendation model is learned for the active user and each other user in the group. However, these multiple models are not learned independently but by using a multi-task learning method. By doing this, preferences from all the users can be captured and transferred across the models simultaneously, and such information help improve the model performance for the active user. A comprehensive set of experiments showed that multi-task learning approaches lead to significant performance improvement over conventional collaborative filtering methods. Note that the multi-task learning approaches for rating prediction have a close connection with the multi-assay-based SAR methods in terms of the related user/related target selection and the multi-task/multi-assay-based learning methodologies, and also relate to the multi-task SSR methods.

10.1.2 Chemical Informatics

Chemical informatics is an interdisciplinary research area in which computational and information technologies are developed and applied to aid the investigation of chemical problems. It is key to the management, analysis and use of chemical data generated by techniques such as combinatorial chemistry and high-throughput screening. Two areas in which chemical informatics plays a critical role are those of chemical genetics and small molecule drug discovery. Chemical genetics seeks to identify small organic molecules (referred to as chemical probes) that can be used to modulate the function of a protein and thus aid in the study and understanding of complex biological systems. Small molecule drug discovery seeks to identify small organic molecules that can modulate the function of pharmaceutically relevant proteins. Within these two areas, chemical informatics provides computational tools to inform the initial and secondary experimental screens and the iterative optimization of the initial compounds (referred to as lead compounds) in order to achieve the desired properties.

The thesis is focusing on developing new computational models and approaches for addressing two important problems in target-based drug discovery. The first is to predict if a compound is *active* for a specific target (i.e., if it binds to the target with high affinity), and the second is to predict if an active compound is *selective* (i.e., if its binding affinities to any other targets are low). Accurate prediction of a compound's activity and selectivity can significantly reduce the time and cost associated with drug/probe development as it allows the lead optimization process to focus on compounds that have high probability of being potent while at the same time have minimal side effects.

The most successful approaches for building computational models to predict the activity and selectivity of compounds are based on supervised learning methods that take into account the molecular structure of the compounds. The models used to predict the activity of a compound are called Structure-Activity-Relationship (SAR) models, whereas the models used to predict the selectivity of a compound are called Structure-Selectivity-Relationship (SSR) models. The rationale of the SAR/SSR models is that the biological properties (e.g., activity, selectivity) of a chemical compound can be expressed as a function of its molecular structure and physicochemical properties. Since these models are built using supervised learning approaches, they utilize as training data compounds whose activity and selectivity is already known.

Multi-Assay-Based SAR **Models**

The state-of-the-art SAR models follow the chemogenomics paradigm, in which the SAR model for a target is built based on the ligands (i.e., compounds that bind to a protein) of that target and the ligands of the proteins that belong to the same protein family (e.g., GPCRs, kinases) as the target. The success of the chemogenomics paradigm illustrates that the ligands of the proteins from the same family share common characteristics, which can be used to improve the quality of the SAR models. The research on this problem was motivated by asking the question whether the performance of SAR models can be improved by utilizing information from other proteins that are not necessarily members of the same family. Driven by this, a new class of methods for building SAR models has been developed, referred to as *multi-assay based* [44], that utilize compound activity information from various targets that may belong to different families but share commonalities in terms of target-ligand binding patterns with the target under consideration. These methods first identify a set of targets that are related (similar) to the target under consideration from all possible protein families, and then they employ various machine learning techniques that incorporate activity information from these targets to build the desired SAR model. Different methods for identifying the set of related targets have been developed, which take into account the primary sequences of the targets or the structure of their ligands. Different machine learning techniques that were derived based on principles of semi-supervised learning, multi-task learning, and classifier ensembles to build the SAR models have also developed. The comprehensive evaluation of these methods on a set of Pubmed activity assays showed that the multi-assay-based SAR models considerably outperform the existing SAR models. In addition, a comparison with chemogenomics-based SAR models showed that the multi-assay-based SAR methods significantly outperform chemogenomics-based methods as well. The results demonstrated that activity information from different protein families can be effectively transferred and utilized to improve SAR models of the targets from other protein families. This work has recently attracted interests from both pharmaceutical industry and academia (e.g., INERIS (French National Institute for Industrial Environment and Risks)).

Models for Predicting Selective Compounds

Most of the current SSR models are built based solely on the ligands of the target under consideration, largely ignoring the targets (referred to as challenge targets) that those ligands loosely

bind to. The activity information from the challenge targets is a rich source of information, which if properly explored, can lead to more accurate selectivity models. In this thesis, a set of machine learning methods has been developed based on multi-task learning to build better SSR models that also incorporate activity information from the challenge targets [142]. In these multi-task methods, the SAR models and the SSR models for the target and the challenge targets are learned in parallel, thus allowing for the transfer of information between the various models. In addition, an alternative method has been developed based on cascading learning that decomposes the selectivity prediction into two steps, one model for each step, so as to effectively filter out non-selective compounds. A comprehensive set of experiments have been conducted and the results demonstrated that the cascaded and multi-task methods significantly improve the selectivity prediction performance over other conventional selectivity prediction methods.

10.2 Future Research Directions

Recommender systems and chemical informatics represent two application areas that have a rich set of challenging machine learning problems with broad and important applications. The rest of this chapter provides an outline of some of the specific problems for the future research.

10.2.1 Recommender Systems

Despite the tremendous advances that occurred in recommender systems in the last ten years, only some of the problems associated with the field have been addressed and there are a lot of opportunities for innovative research that can lead to dramatic quality improvements and also broaden the applicability of these techniques. These opportunities stem from two primary factors: (i) there is an increasing amount of online information that can be used to better understand a user's likes and dislikes (e.g., product reviews, blogs, trust relations, social networks, etc.) and (ii) there is an increasing interest in developing recommender systems for applications in which the things being recommended are complex entities (e.g., travel packages, portfolios, music list, etc). In the future research, both of the factors can be explored and the recommender system methods that leverage the increasing amount of relevant user-generated information can be developed and the general approaches that can be used to develop recommender systems for complex entities can be investigated.

Recommender Systems with Social Information

The future research with respect to the first factor will focus on utilizing social information to improve recommender systems. In general, users establish social relations with other users who have common interests, preferences and behaviors. Thus, recommender systems incorporating users' social relations can better model user preferences. Inspired by this, new methods for recommender systems that leverage social relations and social network-based influence models can be developed. Social network-based influence models can be used to understand how a user's decisions are affected by that of other users, and represent a source of information that complements the information derived from the user's own preferences. Developing recommender algorithms that properly combine this information holds the promise of improving the quality of the recommendations.

Recommender Systems with User Reviews

With respective to user-generated content, the future research will focus on leveraging the rich information that exists in user-provided online reviews (e.g., amazon product reviews, yelp, etc.) in order to better understand a user's preference vector, that is, the various properties of the items that a user finds important and are critical in determining the overall user-centered utility of the items. These preference vectors, when coupled with the corresponding property vectors derived for the items (i.e., what are the properties of the items as they relate to the utility derived by the users who reviewed them) can be used to better inform recommender systems as to what a user values and what an item provides and thus lead to better recommendations.

Recommender Systems for Complex Entities

The future research on the second factor will center around developing effective approaches for recommending complex entities. The complex entities are either combinations of multiple homogeneous/heterogeneous items (e.g., vacation packages), or the entities exhibit various distinct properties that cannot be considered independently for the recommendation purposes (e.g., online dating recommendations). Recommending such complex entities is more challenging than recommending individual items, because the recommended entities need to be feasible (e.g., no conflicts/duplicates/missings among its component items), personalized, diverse (e.g., multiple choices of restaurant and shopping mall combinations) while simultaneously satisfying user

supplied constraints (e.g., early flights and 5-star hotels) and optimize certain objectives (e.g., minimum expense on the vacations). The approaches that exploit the underlying characteristics and combinations among component items/properties of complex entities can be investigated with collective consideration on all the above requirements and optimization for final recommendations.

10.2.2 Drug Repositioning

Motivated by the need to discover drugs for diseases with no known drugs, referred to as orphan diseases, there is a high demand on developing computational methods that can aid in the discovery of such drugs. The research can focus both on extending the multi-task learning framework that has been developed to pharmaceutically relevant protein targets with no known active compounds, as well as developing computational methods for drug repositioning for orphan diseases. Drug repositioning aims to discover new therapeutic applications for an existing drug. It offers several advantages over the traditional drug development process as it circumvents some of the most expensive and time-consuming stages in the drug discovery process (i.e., lead search, probe optimization, toxicity testing, etc) and shortens the path for drug approval. Current computational approaches for drug repositioning exploit chemical similarities between the diseases' existing drugs. In recent years, these approaches have had reasonable success in identifying novel therapeutic uses of existing drugs, and the progress has contributed to the increased interest in these methods. However, this paradigm can not be used to reposition existing drugs to orphan diseases. Thus, the future research on drug repositioning for orphan diseases will focus on developing alternative methods that utilize available information and compensate for the lack of chemical similarities. In particular, the future research will focus on the following two aspects.

First, it is important to study the properties of the known drug-disease interaction networks constructed using the information that is currently known about the different chemical fragments, drugs and diseases. The understanding of the global and local properties of these networks, the biological/chemical properties and their relations of the entities involved, particularly the known drug-repositioning instances, can reveal useful insights on the repositioning mechanism. Second, using these insights, it is critical to develop computational methods that leverage multiple sources of heterogeneous information of drugs and diseases and build models for drug repositioning. The key idea behind these methods is to learn and establish connections between

the orphan diseases and the diseases with known drugs by utilizing information derived from various diverse sources such as phenotypic assays, gene expression data, disease progression, biomarkers, scientific literature, etc.

References

[1] Monique Laurent. Matrix completion problems. *Linear Algebra and its Applications*, 430(8-9):2511–2540, 2009.

[2] Michael P. Friedlander. Bcls: A large-scale solver for bound-constrained least squares., 2006.

[3] M. P. Friedlander and K. Hatz. Computing nonnegative tensor factorizations. *Computational Optimization and Applications*, 23(4):631–647, March 2008.

[4] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1 edition, 1996.

[5] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust Region Methods*. Society for Industrial and Applied Mathematics, Philadephia, PA, 2000.

[6] Christopher C. Paige and Michael A. Saunders. Lsqr: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.*, 8(1):43–71, March 1982.

[7] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[8] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[9] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of the European Conference on Machine Learning*, 1998.

[10] Peichung Shih and Chengjun Liu. Face detection using discriminating feature analysis and support vector machine. *Pattern Recognition*, 39(2):260 – 276, 2006. ¡ce:title¿Part

Special Issue: Complexity Reduction¡/ce:title¿ ¡xocs:full-name¿Part Special Issue: Complexity Reduction¡/xocs:full-name¿.

[11] Kristin P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1(1):23–34, 1992, http://www.tandfonline.com/doi/pdf/10.1080/10556789208805504.

[12] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:pp. 415–446, 1909.

[13] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[14] A. Smola and B. Scholkopf. A tutorial on support vector regression. *NeuroCOLT2*, NC2-TR-1998-030, 1998.

[15] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22:143–177, January 2004.

[16] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[17] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 502–511, Washington, DC, USA, 2008. IEEE Computer Society.

[18] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[19] Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 713–719, New York, NY, USA, 2005. ACM.

[20] Nathan Srebro and Tommi Jaakkola. Generalization error bounds for collaborative prediction with low-rank matrices. In *In Advances In Neural Information Processing Systems 17*, pages 5–27. MIT Press, 2005.

[21] Vikas Sindhwani, Serhat S. Bucak, Jianying Hu, and Aleksandra Mojsilovic. One-class matrix completion with low-density factorizations. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 1055–1060, Washington, DC, USA, 2010. IEEE Computer Society.

[22] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, January 2004.

[23] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Schmidt-Thie Lars. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[25] Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *Proceeding of the 14th ACM International Conference on Knowledge Discovery and Data Mining*, pages 650–658, 2008.

[26] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining*, pages 19–28, 2009.

[27] Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 537–546, New York, NY, USA, 2011. ACM.

[28] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86, 2010.

[29] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, SIGIR '11, pages 635–644, New York, NY, USA, 2011. ACM.

[30] Asela Gunawardana and Christopher Meek. A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 117–124, New York, NY, USA, 2009. ACM.

[31] Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, and Miguel A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785 – 799, 2010.

[32] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufmann, 1998.

[33] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[34] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[35] Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *International Conference on Machine Learning*, pages 489–497, Bari, Italy, 1996. Morgan Kaufmann.

[36] Edwin Bonilla, Felix Agakov, and Christopher Williams. Kernel multi-task learning using task-specific features. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 41–48, San Juan, Puerto Rico, 2007. Omnipress.

[37] Richard A. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48, MA, USA, 1993. Morgan Kaufmann.

[38] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005.

[39] Rich Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7*, pages 657–664, Denver, CO, USA, 1995. Morgan Kaufmann.

[40] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of 22nd International Conference on Machine Learning*, pages 1012–1019, Bonn, Germany, 2005. ACM Press.

[41] Kai Ni, Lawrence Carin, and David Dunson. Multi-task learning for sequential data via ihmms and the nested dirichlet process. In *Proceedings of the 24th international conference on Machine learning*, pages 689–696, New York, NY, USA, 2007. ACM.

[42] Alekh Agarwal, Alexander Rakhlin, and Peter Bartlett. Matrix regularization techniques for online multitask learning. Technical Report UCB/EECS-2008-138, EECS Department, University of California, Berkeley, Oct 2008.

[43] Laurent Jacob, Brice Hoffmann, Veronique Stoven, and Jean-Philippe Vert. Virtual screening of gpcrs: An in silico chemogenomics approach. *BMC Bioinformatics*, 9(1):363, 2008.

[44] Xia Ning, Huzefa Rangwala, and George Karypis. Multi-assay-based structure-activity relationship models: improving structure-activity relationship models by incorporating activity information from related targets. *J Chem Inf Model*, 49(11):2444–2456, Nov 2009.

[45] Feng Jin and Shiliang Sun. A multitask learning approach to face recognition based on neural networks. In *IDEAL '08: Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning*, pages 24–31, Berlin, Heidelberg, 2008. Springer-Verlag.

[46] Mikaela Keller and Samy Bengio. A multitask learning approach to document representation using unlabeled data. IDIAP-RR 44, IDIAP, 2006.

[47] Shipeng Yu, Kai Yu, Volker Tresp, and Hans-Peter Kriegel. Collaborative ordinal regression. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 1089–1096, New York, NY, USA, 2006. ACM.

[48] Kai Yu and Volker Tresp. Learning to learn and collaborative filtering. In *In Neural Information Processing Systems Workshop on Inductive Transfer: 10 Years Later*, 2005.

[49] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3):243–272, 2008.

[50] Jacob Abernethy, Francis Bach, Theodoros Evgeniou, and Jean-Philippe Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *J. Mach. Learn. Res.*, 10:803–826, 2009.

[51] Nguyon Du Phuong and Tu Minh Phuong. Collaborative filtering by multi-task learning. In *IEEE International Conference on Research, Innovation and Vision for the Future.*, pages 227–232, 2008.

[52] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[53] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[54] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal Of The Royal Statistical Society Series B*, 67(2):301–320, 2005.

[55] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Statistics*, pages 2–5, 2007.

[56] Francis Bach, Julien Mairal, and Jean Ponce. Convex sparse matrix factorizations. *CoRR*, abs/0812.1869, 2008.

[57] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.*, 11:19–60, March 2010.

[58] Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, December 2004.

[59] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. *IEEE International Conference on Data Mining*, pages 176–185, 2010.

[60] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[61] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Proceedings of 11th IEEE International Conference on Data Mining*, pages 497–506, 2011.

[62] Deepak Agarwal, Bee-Chung Chen, and Bo Long. Localized factor models for multi-context recommendation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 609–617, New York, NY, USA, 2011. ACM.

[63] Alex J. Smola, Bernhard Schölkopf, and Bernhard Schölkopf. A tutorial on support vector regression. Technical report, Statistics and Computing, 2003.

[64] Dumitru Erhan, Pierre-Jean L'Heureux, Shi Yi Yue, and Yoshua Bengio. Collaborative filtering on a family of biological targets. *Journal of Chemical Information and Modeling*, 46(2):626–635, 2006, http://pubs.acs.org/doi/pdf/10.1021/ci050367t.

[65] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM.

[66] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[67] Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004, http://bioinformatics.oxfordjournals.org/cgi/reprint/20/11/1682.pdf.

[68] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[69] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, 10:623–656, June 2009.

[70] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD*, 4(1), 2010.

[71] Manfred K. Warmuth, Jun Liao, Gunnar Ratsch, Michael Mathieson, Santosh Putta, and Christian Lemmen. Active learning with support vector machines in the drug discovery process. *Journal of Chemical Information and Computer Sciences*, 43(2):667–673, 2003, http://pubs.acs.org/doi/pdf/10.1021/ci025620t.

[72] S. Frye. Structure-activity relationship homology(sarah): a conceptual framework for drug discovery in the genomic era. *Chemistry and Biology*, pages R3–R7, 1999.

[73] P. R. Caron, M. D. Mullican, R. D. Mashal, K. P. Wilson, M. S. Su, and M. A. Murcko. Chemogenomic approaches to drug discovery. *Current Opinion in Chemical Biology*, 5(4):464–70, 2001.

[74] Thomas Klabunde. Chemogenomic approaches to drug discovery: similar receptors bind similar ligands. *British Journal of Pharmacology*, 152(1):5–7, May 2007.

[75] J.R. Bock and D.A. Gough. Virtual screen for ligands of orphan g protein-coupled receptors. *Journal of Chemical Information and Modeling*, 45(5):1402–1414, 2005.

[76] Maris Lapinsh, Peteris Prusis, Staffan Uhlen, and Jarl E. S. Wikberg. Improved approach for proteochemometrics modeling: application to organic compound–amine g protein-coupled receptor interactions. *Bioinformatics*, 21(23):4289–4296, 2005, http://bioinformatics.oxfordjournals.org/cgi/reprint/21/23/4289.pdf.

[77] Anton Lindström, Fredrik Pettersson, Fredrik Almqvist, Anders Berglund, Jan Kihlberg, and Anna Linusson. Hierarchical pls modeling for predicting the binding of a comprehensive set of structurally diverse protein-ligand complexes. *Journal of Chemical Information and Modeling*, 46(3):1154–1167, 2006, http://pubs.acs.org/doi/pdf/10.1021/ci050323k.

[78] Laurent Jacob and Jean-Philippe Vert. Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, 24(19):2149–2156, 2008, http://bioinformatics.oxfordjournals.org/cgi/reprint/24/19/2149.pdf.

[79] Helena Strömbergsson, Pawel Daniluk, Andriy Kryshtafovych, Krzysztof Fidelis, Jarl E. S. Wikberg, Gerard J. Kleywegt, and Torgeir R. Hvidsten. Interaction model based on local protein substructures generalizes to the entire structural enzyme-ligand space. *Journal of Chemical Information and Modeling*, 48(11):2278–2288, 2008, http://pubs.acs.org/doi/pdf/10.1021/ci800200e.

[80] Zhan Deng, Claudio Chuaqui, and Juswinder Singh. Structural interaction fingerprint (sift): a novel method for analyzing three-dimensional protein-ligand binding interactions. *Journal of Medicinal Chemistry*, 47(2):337–344, Jan 2004.

[81] Nathanael Weill and Didier Rognan. Development and validation of a novel protein-ligand fingerprint to mine chemogenomic space: Application to g protein-coupled receptors and their ligands. *Journal of Chemical Information and Modeling*, 0(0), 2009, http://pubs.acs.org/doi/pdf/10.1021/ci800447g.

[82] Hanna Geppert, Jens Humrich, Dagmar Stumpfe, Thomas Gartner, and Jurgen Bajorath. Ligand prediction from protein sequence and small molecule information using support vector machines and fingerprint descriptors. *Journal of Chemical Information and Modeling*, 0(0), 2009, http://pubs.acs.org/doi/pdf/10.1021/ci900004a.

[83] Alex J. Smola and Bernhard Sch Olkopf. A tutorial on support vector regression. Technical report, Statistics and Computing, 1998.

[84] Ingo Vogt, Dagmar Stumpfe, Hany E A Ahmed, and Jrgen Bajorath. Methods for computer-aided chemical biology. part 2: Evaluation of compound selectivity using 2d molecular fingerprints. *Chem Biol Drug Des*, 70(3):195–205, Sep 2007.

[85] Dagmar Stumpfe, Hanna Geppert, and Jrgen Bajorath. Methods for computer-aided chemical biology. part 3: analysis of structure-selectivity relationships through single- or dual-step selectivity searching and bayesian classification. *Chem Biol Drug Des*, 71(6):518–528, Jun 2008.

[86] Anne Mai Wassermann, Hanna Geppert, and Jrgen Bajorath. Searching for target-selective compounds using different combinations of multiclass support vector machine ranking methods, kernel functions, and fingerprint descriptors. *J Chem Inf Model*, 49(3):582–592, Mar 2009.

[87] Anne Mai Wassermann, Hanna Geppert, and Jrgen Bajorath. Application of support vector machine-based ranking strategies to search for target-selective compounds. *Methods Mol Biol*, 672:517–530, 2011.

[88] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

[89] Anil K. Jain, Jianchang Mao, and K. Mohiuddin. Artificial neural networks: A tutorial. *IEEE Computer*, 29:31–44, 1996.

[90] Leo Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.

[91] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.

[92] Florian Nigsch, Andreas Bender, Jeremy L. Jenkins, and John B. O. Mitchell. Ligand-target prediction using winnow and naive bayesian algorithms and the implications of overall performance statistics. *Journal of Chemical Information and Modeling*, 48(12):2313–2325, 2008, http://pubs.acs.org/doi/pdf/10.1021/ci800079x.

[93] Florian Nigsch and John B. O. Mitchell. How to winnow actives from inactives: Introducing molecular orthogonal sparse bigrams (mosbs) and multiclass winnow. *Journal of Chemical Information and Modeling*, 48(2):306–318, 2008, http://pubs.acs.org/doi/pdf/10.1021/ci700350n.

[94] Helena Strömbergsson, Peteris Prusis, Herman Midelfart, Maris Lapinsh, Jarl E.S. Wikberg, and Jan Komorowski. Rough set-based proteochemometrics modeling of g-protein-coupled receptor-ligand interactions. *Proteins: Structure, Function, and Bioinformatics*, 63(1):24–34, 2006.

[95] J. F. Peters and A. Skowron. A rough set approach to reasoning about data. *International Journal of Intelligent Systems*, 16(1):1–2, 2001.

[96] Gianpaolo Bravi, Emanuela Gancia ; Darren Green, V.S. Hann, and M. Mike. Modelling structure-activity relationship. In H.J. Bohm and G. Schneider, editors, *Virtual Screening for Bioactive Molecules*, volume 10. Wiley-VCH, New York, NY, USA, August 2000.

[97] J.M.Barnard P. Willett and G.M.Downs. Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, 38:983–997, 1998.

[98] Dagmar Stumpfe, Hany E A Ahmed, Ingo Vogt, and Jrgen Bajorath. Methods for computer-aided chemical biology. part 1: Design of a benchmark system for the evaluation of compound selectivity. *Chem Biol Drug Des*, 70(3):182–194, Sep 2007.

[99] Nikil Wale, Ian A. Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

[100] Nikil Wale and George Karypis. Afgen. Technical report, Department of Computer Science & Enigneering, University of Minnesota, 2007. www.cs.umn.edu/ karypis.

[101] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004.

[102] C. Hansch, P. P. Maolney, T. Fujita, and R. M. Muir. Correlation of biological activity of phenoxyacetic acids with hammett substituent constants and partition coefficients. *Nature*, 194:178–180, 1962.

[103] C. Hansch, R. M. Muir, T. Fujita, C. F. Maloney, and M. Streich. The correlation of biological activity of plant growth-regulators and chloromycetin derivatives with hammett constants and partition coefficients. *Journal of American Chemical Society*, 85:2817–1824, 1963.

[104] D.K. Agrafiotis, D. Bandyopadhyay, J.K. Wegner, and H. van Vlijmen. Recent advances in chemoinformatics. *Journal of Chemical Information and Modeling*, 47(4):1279–1293, 2007.

[105] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[106] Sebastian Thurn. Is learning the *n*-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, pages 640–646, NIPS Foundation, 1996. The MIT Press.

[107] Rosemarie Swanson and Jerry Tsai. Pretty Good Guessing: Protein Structure Prediction at CASP5. *J. Bacteriol.*, 185(14):3990–3993, 2003, http://jb.asm.org/cgi/reprint/185/14/3990.pdf.

[108] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach. Learn.*, 51(2):181–207, 2003.

[109] Catherine A. Shipp and Ludmila I. Kuncheva. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, 3(2):135 – 148, 2002.

[110] H. Yu, J. Han, and K. Chang. Pebl: Positive example based learning for web page classification using svm. In *ACM KDD*, pages 239–248, New York, NY, USA, 2002. ACM.

[111] B. Liu, Y. Dai, X. Li, W.S. Lee, and P.S. Yu. Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd ICDM Conference*, pages 179–188, Florida, USA, 2003. IEEE Computer Society Press.

[112] Chunlin Wang, Chris Ding, Richard F Meraz, and Stephen R Holbrook. Psol: a positive sample only learning algorithm for finding non-coding rna genes. *Bioinformatics*, 22(21):2590–2596, Nov 2006.

[113] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on*

Knowledge discovery and data mining, pages 213–220, New York, NY, USA, 2008. ACM.

[114] E. K. Davies and C. Briant. *Combinatorial Chemistry Library Design Using Pharmacophore Diversity*, volume 118, pages 309–316. American Chemical Society, US, January 1996.

[115] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[116] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI–BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[117] A. Heger and L. Holm. Picasso:generating a covering set of protein family profiles. *Bioinformatics*, 17(3):272–279, 2001.

[118] Huzefa Rangwala and George Karypis. Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239–4247, 2005, http://bioinformatics.oxfordjournals.org/cgi/reprint/21/23/4239.pdf.

[119] Xiaojin Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005.

[120] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2002.

[121] Yvonne C. Martin, James L. Kofron, and Linda M. Traphagen. Do structurally similar molecules have similar biological activity? *Journal of Medicinal Chemistry*, 45(19):4350–4358, 2002, http://pubs.acs.org/doi/pdf/10.1021/jm020155c.

[122] J. Weston, A. Elisseeff, D. Zhou, C. Leslie, and W. S. Noble. Protein ranking: from local to global structure in protein similarity network. *PNAS USA*, 101:6559–6563, 2004.

[123] Igor V Tetko and Vsevolod Yu Tanchuk. Application of associative neural networks for prediction of lipophilicity in alogps 2.1 program. *Journal of Chemical Information and Computer Sciences*, 42(5):1136–1145, 2002.

[124] Igor V Tetko, Iwona Jaroszewicz, James A Platts, and Janina Kuduk-Jaworska. Calculation of lipophilicity for pt(ii) complexes: experimental comparison of several methods. *Journal of Inorganic Biochemistry*, 102(7):1424–1437, Jul 2008.

[125] Alexandre Varnek, Cédric Gaudin, Gilles Marcou, Igor Baskin, Anil Kumar Pandey, and Igor V Tetko. Inductive transfer of knowledge: Application of multi-task learning and feature net approaches to model tissue-air partition coefficients. *Journal of Chemical Information and Modeling*, 49(1):133–144, Jan 2009.

[126] G. R. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Pacific Symposium on Biocomputing*, pages 300–11, Berkeley, USA, 2004. World Scientific Publishing.

[127] S. Sonnenburg, G. Ratsch, and C. Schafer. A general and efficient multiple kernel learning algorithm. *Proceedings of the 2005 Neural Information Processing Systems*, 2005.

[128] I. W. Tsang and J. T . Kwok. Efficient hyperkernel learning using second-order cone programming. *IEEE Transactions on Neural Networks*, 17(1):48–58, 2006.

[129] Huzefa Rangwala and George Karypis. frmsdalign: Protein sequence alignment using predicted local structure information for pairs with low sequence identity. In Alvis Brazma, Satoru Miyano, and Tatsuya Akutsu, editors, *APBC*, volume 6 of *Advances in Bioinformatics and Computational Biology*, pages 111–122, Kyoto, Japan, 2008. Imperial College Press.

[130] Christopher P Austin, Linda S Brady, Thomas R Insel, and Francis S Collins. Nih molecular libraries initiative. *Science*, 306(5699):1138–1139, Nov 2004.

[131] Thorsten Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

[132] Mazen W Karaman, Sanna Herrgard, Daniel K Treiber, Paul Gallant, Corey E Atteridge, Brian T Campbell, Katrina W Chan, Pietro Ciceri, Mindy I Davis, Philip T Edeen, Raffaella Faraoni, Mark Floyd, Jeremy P Hunt, Daniel J Lockhart, Zdravko V Milanov, Michael J Morrison, Gabriel Pallares, Hitesh K Patel, Stephanie Pritchard, Lisa M Wodicka, and Patrick P Zarrinkar. A quantitative analysis of kinase inhibitor selectivity. *Nat Biotechnol*, 26(1):127–132, Jan 2008.

[133] Ryangguk Kim and Jeffrey Skolnick. Assessment of programs for ligand binding affinity prediction. *J Comput Chem*, 29(8):1316–1331, Jun 2008.

[134] Dagmar Stumpfe, Eugen Lounkine, and Jrgen Bajorath. Molecular test systems for computational selectivity studies and systematic analysis of compound selectivity profiles. *Methods Mol Biol*, 672:503–515, 2011.

[135] Nikil Wale, Xia Ning, and George Karypis. Trends in chemical graph data mining. In Ahmed K. Elmagarmid, Charu C. Aggarwal, and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 581–606. Springer US, New York, NY, US, 2010.

[136] Robert Clarke, Habtom W Ressom, Antai Wang, Jianhua Xuan, Minetta C Liu, Edmund A Gehan, and Yue Wang. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nat Rev Cancer*, 8(1):37–49, Jan 2008.

[137] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Mit Press Computational Models Of Cognition And Perception Series*, pages 318–362, 1986.

[138] Xia Ning and George Karypis. Slim: Sparse linear models for top-n recommender systems. In *IEEE International Conference on Data Mining*, 2011.

[139] Xia Ning and George Karypis. Sparse linear models with side-information for top-n recommender systems. WWW2012, accepted as poster, 2012.

[140] Xia Ning and George Karypis. Sparse linear models with side-information for top-n recommender systems. RecSys2012, accepted, 2012.

[141] Xia Ning and George Karypis. Multi-task learning for recommender systems. In *Journal of Machine Learning Research Workshop and Conference Proceedings (ACML2010)*, volume 13, pages 269–284. Microtome Publishing, 2010.

[142] Xia Ning, Michael Walters, and George Karypis. Improved machine learning models for predicting selective compounds. *Journal of Chemical Information and Modelling*, 2011.

Appendix A

The Label Propagation Algorithm

Algorithm 1 *LABEL_PROPAGATION($C_i \cup C_i^-$, U_i)*

1: **procedure** LP ($C_i \cup C_i^-$, U_i)
2: **for all** $c_p, c_q \in C_i \cup C_i^- \cup U_i$ **do**
3: $w_{p,q} = \text{sim}_c(c_p, c_q)$
4: **end for**
5: **for all** $c_p, c_q \in C_i \cup C_i^- \cup U_i$ **do**
6: $P_{p,q} = \frac{w_{p,q}}{\sum_{k=1}^{|\mathbb{C}|} w_{k,q}}$
7: **end for**
8: **for all** $c_p \in C_i$ **do**
9: $L(p, 0) \leftarrow 1$
10: $L(p, 1) \leftarrow 0$
11: **end for**
12: **for all** $c_q \in C_i^-$ **do**
13: $L(p, 0) \leftarrow 0$
14: $L(p, 1) \leftarrow 1$
15: **end for**
16: **for all** $c_p \in U_i$ **do**
17: $L(q, 0) \leftarrow 0.5$
18: $L(q, 1) \leftarrow 0.5$
19: **end for**
20: **while** !L converges **do**
21: $L \leftarrow TL$
22: Row-normalize L

```
23:     end while
24:     for all c_p ∈ U_i do
25:         if L(p,0) > L(p,1) then
26:             L(p,0) ← 1
27:             L(p,1) ← 0
28:         else
29:             L(p,0) ← 0
30:             L(p,1) ← 1
31:         end if
32:     end forreturn L
33: end procedure

end
```

Appendix B

The $\mathrm{USM}_{\mathrm{cvg}}$ Algorithm

Algorithm 2 *USMcvg*

1: **procedure** USMcvg($(u_{*_1}, u_{*_2}, \ldots, u_{*_{2m}})$, m, $\mathcal{T}_*$)
2: $\mathcal{N}_* \leftarrow \{u_{*_1}\}$
3: $i = n = 1$
4: **while** $(i < 2m)$ **do**
5: **for** $j = 1$ to WINDOWS_SIZE **do**
6: $k = i + j$
7: **if** $k > 2m$ **then**
8: goto **L**
9: **else**
10: count the number of items that have been rated by u_{*_k}
11: but not rated by u_* or $\forall u \in \mathcal{N}_*$
12: **end if**
13: **end for**
14: **end while**
15: look for $u_{*_{k_{max}}}$ with maximum non-zero count in this window
16: **if** such user $u_{*_{k_{max}}}$ exists **then**
17: $\mathcal{N}_* = \mathcal{N}_* \cup \{u_{*_{k_{max}}}\}$
18: $n = n + 1$
19: $i = k_{max}$
20: **else**
21: $i = i$+WINDOWS_SIZE
22: **end if**

```
23:     if n ≥ m then
24:         break
25:     end if
26:     L:
27:     if n < m then
28:         select m − n most similar users that are not selected into N*
29:     end ifreturn N*
30: end procedure

end
```

Appendix C

Solving SSLIM

C.1 Reformulation of cSLIM

The formulation of cSLIM

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \frac{1}{2}\|M - MS\|_F^2 + \frac{\alpha}{2}\|F - FS\|_F^2 \\ & + \frac{\beta}{2}\|S\|_F^2 + \lambda\|S\|_1 \\ \text{subject to} \quad & S \geq 0, \\ & \text{diag}(S) = 0, \end{aligned} \tag{C.1}$$

can be reformulated as follows,

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \frac{1}{2}\|M' - M'S\|_F^2 + \frac{\beta}{2}\|S\|_F^2 + \lambda\|S\|_1 \\ \text{subject to} \quad & S \geq 0, \\ & \text{diag}(S) = 0, \end{aligned} \tag{C.2}$$

by setting

$$M' = \begin{bmatrix} M \\ \sqrt{\alpha}F \end{bmatrix},$$

Equation C.2 is essentially identical to Equation C.1.

C.2 Reformulation of `rcSLIM`

The formulation of `rcSLIM`

$$\begin{aligned} \underset{S,Q}{\text{minimize}} \quad & \frac{1}{2}\|M - MS\|_F^2 + \frac{\alpha}{2}\|F - FQ\|_F^2 \\ & + \frac{\beta_1}{2}\|S - Q\|_F^2 + \frac{\beta_2}{2}(\|S\|_F^2 + \|Q\|_F^2) \\ & + \lambda(\|S\|_1 + \|Q\|_1) \\ \text{subject to} \quad & S \geq 0, \\ & Q \geq 0, \\ & \text{diag}(S) = 0, \\ & \text{diag}(Q) = 0. \end{aligned} \tag{C.3}$$

can be solved using an alterating approach as in Algorithm 3.

Algorithm 3 *Alternating Method for* `rcSLIM`

1: **procedure** `rcSLIM` (M, F, α, β_1, β_2, λ, S_0, Q_0, $maxIters$)
2: $S \leftarrow S_0$ ▷ Initialize S
3: $Q \leftarrow Q_0$ ▷ Initialize Q
4: $niters \leftarrow 0$
5: **repeat**
6: $niters++$
7: Fix Q and solve the problem C.4 for S

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \frac{1}{2}\|M - MS\|_F^2 + \frac{\beta_1}{2}\|Q - S\|_F^2 \\ & + \frac{\beta_2}{2}\|S\|_F^2 + \lambda\|S\|_1 \\ \text{subject to} \quad & S \geq 0, \\ & \text{diag}(S) = 0 \end{aligned} \tag{C.4}$$

8: Fix S and solve the problem C.5 for Q

$$\begin{aligned} \underset{Q}{\text{minimize}} \quad & \frac{\alpha}{2}\|F - FQ\|_F^2 + \frac{\beta_1}{2}\|S - Q\|_F^2 \\ & + \frac{\beta_2}{2}\|Q\|_F^2 + \lambda\|Q\|_1 \\ \text{subject to} \quad & Q \geq 0, \\ & \text{diag}(Q) = 0. \end{aligned} \tag{C.5}$$

9: **until** $niters \geq maxIters$ or S and Q converge

10: **end procedure**

end

C.3 Reformulation of `fSLIM`

The formulation of `fSLIM`

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \frac{1}{2}\|M - M(FW - D)\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|FW\|_1 \\ \text{subject to} \quad & W \geq 0 \\ & D = \text{diag}(\text{diag}(FW)), \end{aligned} \tag{C.6}$$

can be reformulated as follows. Since

$$D_{ij} = \begin{cases} \sum_{k=1}^{l} F_{ik}W_{kj}, & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases}$$

for the j-th column of W, we have

$$
\begin{aligned}
& M(F\mathbf{w}_j - \mathbf{d}_j) \\
= & \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1,m-1} & m_{1m} \\ & & \cdots & & \\ m_{n1} & m_{n2} & \cdots & m_{n,m-1} & m_{nm} \end{bmatrix} \times \\
& \left(\begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1,l-1} & f_{1l} \\ & & \cdots & & \\ f_{j-1,1} & f_{j-1,2} & \cdots & f_{j-1,l-1} & f_{j-1,l} \\ f_{j1} & f_{j2} & \cdots & f_{j,l-1} & f_{jl} \\ f_{j+1,1} & f_{j+1,2} & \cdots & f_{j+1,l-1} & f_{j+1,l} \\ & & \cdots & & \\ f_{m1} & f_{m2} & \cdots & f_{m,l-1} & f_{ml} \end{bmatrix} \begin{bmatrix} w_{1j} \\ w_{2j} \\ \cdots \\ w_{jj} \\ \cdots \\ w_{l-1,j} \\ w_{l,j} \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ & & \cdots & & \\ 0 & 0 & \cdots & 0 & 0 \\ f_{j1} & f_{j2} & \cdots & f_{j,l-1} & f_{jl} \\ 0 & 0 & \cdots & 0 & 0 \\ & & \cdots & & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} w_{1j} \\ w_{2j} \\ \cdots \\ w_{jj} \\ \cdots \\ w_{l-1,j} \\ w_{l,j} \end{bmatrix} \right) \qquad \text{(C.7)} \\
= & \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1,m-1} & m_{1m} \\ & & \cdots & & \\ m_{n1} & m_{n2} & \cdots & m_{n,m-1} & m_{nm} \end{bmatrix} \times \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1,l-1} & f_{1l} \\ & & \cdots & & \\ f_{j-1,1} & f_{j-1,2} & \cdots & f_{j-1,l-1} & f_{j-1,l} \\ 0 & 0 & \cdots & 0 & 0 \\ f_{j+1,1} & f_{j+1,2} & \cdots & f_{j+1,l-1} & f_{j+1,l} \\ & & \cdots & & \\ f_{m1} & f_{m2} & \cdots & f_{m,l-1} & f_{ml} \end{bmatrix} \begin{bmatrix} w_{1j} \\ w_{2j} \\ \cdots \\ w_{jj} \\ \cdots \\ w_{l-1,j} \\ w_{l,j} \end{bmatrix}
\end{aligned}
$$

Let

$$
F_{-j} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1,l-1} & f_{1l} \\ & & \cdots & & \\ f_{j-1,1} & f_{j-1,2} & \cdots & f_{j-1,l-1} & f_{j-1,l} \\ 0 & 0 & \cdots & 0 & 0 \\ f_{j+1,1} & f_{j+1,2} & \cdots & f_{j+1,l-1} & f_{j+1,l} \\ & & \cdots & & \\ f_{m1} & f_{m2} & \cdots & f_{m,l-1} & f_{ml} \end{bmatrix}
$$

In addition, since $W \geq 0$ and $F \geq 0$,

$$
\begin{aligned}
& \|F\mathbf{w}_j\|_1 \\
= & \left\| \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1,l-1} & f_{1l} \\ & & \cdots & & \\ f_{j-1,1} & f_{j-1,2} & \cdots & f_{j-1,l-1} & f_{j-1,l} \\ f_{j1} & f_{j2} & \cdots & f_{j,l-1} & f_{jl} \\ f_{j+1,1} & f_{j+1,2} & \cdots & f_{j+1,l-1} & f_{j+1,l} \\ & & \cdots & & \\ f_{m1} & f_{m2} & \cdots & f_{m,l-1} & f_{ml} \end{bmatrix} \begin{bmatrix} w_{1j} \\ w_{2j} \\ \cdots \\ w_{jj} \\ \cdots \\ w_{l-1,j} \\ w_{l,j} \end{bmatrix} \right\|_1 \\
= & \sum_{k=1}^{m} \sum_{p=1}^{l} f_{kp} w_{pj} = \sum_{p=1}^{l} w_{pj} \sum_{k=1}^{m} f_{kp} = \sum_{p=1}^{l} w_{pj} c_p \\
= & \begin{bmatrix} c_1, c_2, \cdots, c_{l-1}, c_l \end{bmatrix} \begin{bmatrix} w_{1j} \\ w_{2j} \\ \cdots \\ w_{jj} \\ \cdots \\ w_{l-1,j} \\ w_{l,j} \end{bmatrix}
\end{aligned}
\tag{C.8}
$$

where $c_p = \sum_{k=1}^{m} f_{kp}$ is the sum of the p-th column of F. The j-th column of W can be solved

$$
\begin{aligned}
\underset{\mathbf{w}_j}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{m}_j - MF_{-j}\mathbf{w}_j\|_F^2 + \frac{\beta}{2}\|\mathbf{w}_j\|_F^2 + \lambda\|\mathbf{c}\mathbf{w}_j\|_1 \\
\text{subject to} \quad & \mathbf{w}_j \geq 0.
\end{aligned}
\tag{C.9}
$$

C.4 Reformulation of `f2SLIM`

The formulation of `f2SLIM`

$$\begin{aligned} \underset{S,W}{\text{minimize}} \quad & \frac{1}{2}\|M - MS - M(FW - D)\|_F^2 \\ & + \frac{\beta}{2}(\|S\|_F^2 + \|W\|_F^2) + \lambda(\|S\|_1 + \|FW\|_1) \\ \text{subject to} \quad & S \geq 0, \\ & W \geq 0, \\ & \text{diag}(S) = 0, \\ & D = \text{diag}(\text{diag}(FW)). \end{aligned} \tag{C.10}$$

is equivalent to the following one

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \frac{1}{2}\|M - M(F'W' - D')\|_F^2 \\ & + \frac{\beta}{2}\|W'\|_F^2 + \lambda\|F'W'\|_1 \\ \text{subject to} \quad & W' \geq 0, \\ & D' = \text{diag}(\text{diag}(F'W')). \end{aligned}$$

where $F' = [F, I]$ in where I is a size $n \times n$ identity matrix, $W' = \begin{bmatrix} W \\ S \end{bmatrix}$, $D' = diag(FW + S)$ and W and S are the solution for problem C.10. This is because for the objective function

$$\begin{aligned} & M - M(F'W' - D') \\ = \; & M - M\left(\begin{bmatrix} F & I \end{bmatrix}\begin{bmatrix} W \\ S \end{bmatrix} - D'\right) \\ = \; & M - M(FW + S - D') \end{aligned} \tag{C.11}$$

Since $W' \geq 0$, then $W \geq 0$ and $S \geq 0$, $\|W'\|_F^2 = \|W\|_F^2 + \|S\|_F^2$, and $\|W'\|_1 = \|W\|_1 + \|S\|_1$. For the diagonal matrix D'

$$\begin{aligned} D'_{jj} = \; & (FW + S)_{jj} \\ = \; & (FW)_{jj} + s_{jj} \\ = \; & \sum_{k=1}^{l} f_{jk} w_{kj} + s_{jj} \end{aligned} \tag{C.12}$$

where $\left[\sum_{k=1}^{l} f_{jk} w_{kj}\right] = diag(FW)$ and $\left[s_{jj}\right] = diag(S)$. Then Equation C.11 becomes

$$\begin{aligned} & M - M(F'W' - D') \\ = \; & M - M(FW - diag(FW) + S - diag(S)) \\ = \; & M - M(S - diag(S)) - M(FW - diag(FW)) \\ = \; & M - M(S - diag(S)) - M(FW - D) \end{aligned} \tag{C.13}$$

and this is equivelant to

$$M - MS - M(FW - D) \tag{C.14}$$

with the constraint $diag(S) = 0$.