UNIVERSITY OF CALIFORNIA
SANTA CRUZ

SESSION AWARE RECOMMENDER SYSTEM IN E-COMMERCE

A dissertation submitted in partial satisfaction
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Jian Wang

June 2013

The Dissertation of Jian Wang is
approved:

Professor Yi Zhang, chair

Professor Manfred Warmuth

Professor Dimitris Achlioptas

Neel Sundaresan, Ph.D.

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Contents

List of Figures

List of Tables

Abstract

Session-aware Recommendation System in E-commerce

Jian Wang

As online shopping becomes to be popular, the recommender system in e-commerce sites is an increasingly popular business tool to increase sales. Researchers and industry practitioners are looking for all possible approaches to improve recommendation performance. Even a minor improvement could lead to a big business return.

Traditional approaches of recommender systems include content-based methods and collaborative filtering methods. For example, if a user viewed some cameras in the website, the system learns that the user is interested in cameras and recommends more similar items to the user. Yet it might not match with the user's true purchase intention. In real-world applications, recommender systems could leverage more information from the user, both information within a single session and information across sessions. To solve this problem, we propose to investigate the **session-aware recommender system** in e-commerce. Such system can understand a user's short-term goal and long-term preference, in order to recommend appropriate items accordingly.

We first explore how to integrate the complementary information (e.g. the user's purchase information, search information and so on) within a single session to build a *unified recommender system.* We analyze the available information for the unified model, including the user's history-related information, the search-related information and the product's marketing-related information. Three unified models are proposed and compared to integrate different pieces of information.

To go beyond making recommendations within a single session, we then study how to make better recommendations across sessions. To make recommendations based on a user's previous behavior in earlier sessions, we need to understand how users make purchase decisions across sessions. Earlier research in economics and marketing indicates that a consumer usually makes purchase decision(s) based on the product's marginal net utility (i.e., the marginal utility minus the product price). Utility is defined as the satisfaction or pleasure a user gets when purchasing the corresponding product. A rational consumer chooses the product to purchase in order to maximize the total net utility. To better match users' purchase decisions in the real world, we explore how to recommend products with the highest marginal net utility in e-commerce sites. Inspired by the Cobb-Douglas utility function in consumer behavior theory, we propose a novel utility-based recommendation framework. The framework can be utilized to revamp a family of existing recommendation algorithms.

To further incorporate the time interval between sessions into the system, we propose and study a new problem: how to recommend the right product at the right time? We adapt the proportional hazards model in survival analysis and propose the new *opportunity model* in e-commerce. The new model estimates the joint probability of a user making a follow-up purchase of a particular product at a particular time. This joint purchase probability can be leveraged by recommender systems in various scenarios, including a zero-query pull-based scenario or a proactive push-based email promotion scenario.

We evaluate the soundness of our models with multiple metrics. Experimental results with a real-world e-commerce website (shop.com) show that they have decent predictability of the user's purchase behavior within a session and across sessions. In addition, the models significantly improve the conversion rate in pull-based systems and the user satisfaction/utility in push-based systems.

In this dissertation, we first introduce the motivation of the work. Secondly, we report some state-of-art related work of this topic. Thirdly, models are proposed to tackle the problem, followed by the experimental results. Then we summarize our contribution in both the research field of recommender systems and the e-commerce domain.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my great advisor Prof. Yi Zhang for her continuous support of my Ph.D. study and research. I would like to thank her for her great encouragement when I started the research journey, for her tremendous guidance when I explored various research topics, for her huge effort to help me writing the paper and preparing the presentation, and for her being a great friend in my life. Without her guidance and persistent help this dissertation would not have been possible.

I would like to thank my committee members, Professor Manfred Warmuth, Professor Dimitris Achlioptas, Professor Marilyn Walker and Dr Neel Sundaresan. They gave me insightful comments which make the dissertation more comprehensive. Besides, Dr Neel Sundaresan gave me the great opportunity of internship in eBay research labs. I discovered my interest of studying recommender systems in the e-commerce domain at that time and started this wonderful journey.

In addition, a thank you to Professor Brian Davison, who introduced me to the area of information retrieval. I would like to thank him for his strong guidance of my research and financial support during my master program. He gave me great opportunities in WUME lab in Lehigh university and guided me through diverse exciting projects. I would also like to thank Professor Tao Chen in University of Maryland, who helped me a lot in building the unified recommender system in Section 3.

I am glad that I joined the IRKM lab in the University of California Santa Cruz four years ago. It is a wonderful place to do research. I am grateful for all my friends in the school, Lanbo Zhang, Sarah Tyler, Qi Zhao, Yize Li, Aaron Michelony, Shawn Wolfe, Jonathan Koren, Yunfei Chen. I would like to thank Lanbo for discussing research questions with me, Sarah, Aaron and Shawn for proofreading my papers, and Qi for being a wonderful labmate. I am also grateful for the NSF funding which supported my research all the time.

The success of my Ph.D. study also depends largely on the encouragement and support from my family. I would like to express my deep gratitude to my parents, HuaYang and JianNa. They gave me life and wonderful love which I can feel all the time, no matter where I am. They showed me the beauty of science when I was young, which I enjoy so much everyday. I would like to express my gratitude to my parents-in-law, GuoHua and YePing. They gave me strong mental support during my graduate study. Last but not least, I am deeply indebted to my dear husband Tian for his tremendous love, support and encouragement. He is my strong motivation of my study and invaluable mental support during all my ups and downs. This dissertation is dedicated to my dear family.

To Tian and our baby

&

my parents HuaYang and JianNa

Chapter 1

Introduction

1.1 Traditional Recommender Systems and Their Limitation

Recommender systems are popular research topics in the information retrieval community. Host of academic and industrial incarnations of recommender systems exists in domains such as movies (Netflix), music (Pandora), e-commerce product recommendations (eBay, Amazon) and so on.

With ever increasing e-commerce websites and online shoppers, the recommender system is becoming popular among internet users. It helps consumers to make purchase decisions, mainly by gathering information from other users. The recommender system is an essential part of the e-commerce website. For repurchase customers, it can provide "subscribe & save" service to the user. For variety-seeking customers, it can find good follow-up products for cross-sell promotion. For customers that want to buy new products, it can save users' time and effort to find what they are looking for. In the long term, it helps to increase users' satisfaction rate and loyalty to the site.

A major task of the recommender system is to present good item suggestions to the user proactively, in order to save the user's time and effort. The task is usually conducted by first predicting a user's ratings for each item and then ranking all items in descending order. There are different information sources to find the relevant recommendation, including the item's content, the user's behavior history, the user's demographical information and so forth. There are two major recommendation approaches: content-based filtering and collaborative filtering.

Content-based recommendations are based on the assumption that descriptive features of an item (meta data, words in description, price, tags, etc.) tell much about a user's preferences to the item. Thus a recommender system makes a decision for a user based on the descriptive features of other items the user likes or dislikes. Usually, the system recommends items that are similar to what the user liked before. A user profile contains his/her previous transaction history, such as what he/she viewed or purchased before. How to determine the similarity between an item and the profile is the key challenge. Cosine similarity with TF.IDF term weights, the language modeling approach, Bayesian classifiers, clustering, etc., have been proposed [41, 55, 56]. Largely influenced by the TREC filtering track, most of the early research on content-based filtering is about filtering text documents.

In the collaborative filtering approach, user behavior history is leveraged to make recommendations. This approach is based on the assumption that users with similar tastes on some items may also have similar preferences on other items. Thus the main idea is to leverage the behavior history from other like-minded users to provide the current user with good recommendations. This approach usually operates on a user-item matrix, in which each row is a user vector and each column is an item vector. User-based methods find similar users to the current user, and fall into the category of **memory-based** approaches in the literature [36, 66, 71, 62, 8, 26]. However, these methods suffer from poor quality and scalability issues [65, 10]. Item-based methods directly find items that are similar to the items a user has rated/purchased [65, 34, 17, 46]. There are two challenges. First, how to determine the item-item similarity. Second, how to select recommendations from all item neighbors. For the first one, Various similarity measures, such as cosine similarity, Pearson correlation coefficient, conditional probability-based similarity, have been proposed to determine the similarity between items. For the second one, the solution is to leverage both the current user's existing ratings and similarity of item neighbors to make the prediction. In addition, a group of model-based approaches have been developed in recent years. Model-based approaches use the collection of user behavior (ratings, purchases, etc.) to learn a model, and make predictions based on the learned model. Examples are Probabilistic Latent Semantic Indexing (PLSI) [29], Flexible Mixture Models [74], Decoupled Models [32], Multiple Multiplicative Factor Model [50], etc.

Research on collaborative filtering algorithms has reached a peak due to the 1 million dollar Netflix movie recommendation competition [5]. Factorization-based collaborative filtering approaches, such as the regularized Singular Value Decomposition, perform well on this competition, significantly better than the Netflix own well-tuned Pearson correlation coefficient (nearest neighbors) algorithm. A common theme of these best-performing models is to introduce user latent factors and/or product latent factors to solve the data sparsity issue. The top two finalists were hybrids that combined many complementary models together. There are many extensions (AsySVD, SVD++, etc.) to the basic SVD in the literature [16, 37].

Recommendation in the e-commerce domain has received much attention [66, 69, 35, 21, 13]. Several methods have been studied in this domain, including standard collaborative filtering algorithms, MDP-based methods [70], utility theory-based methods [44, 84] and so on. Recommendation for long-tail users and long-tail products is a major challenge with very limited research.

Traditional recommender systems in the e-commerce domain have the following limitations:

- Traditional recommender systems mostly leverage the user's purchase history to make recommendations within a session. In a real-world e-commerce website, there is more useful information to use, such as the user's short-term behavior(search, click, etc.). The user's heterogenous information tells much about a user's purchase intention in the current session, which could help the system to find matching products.

- Traditional recommender systems often select products with the highest predicted ratings to recommend to the user. However, earlier research in economics and marketing indicates that a consumer usually makes purchase decision(s) based on the product's marginal net utility (i.e., the marginal utility minus the product price). Utility is defined as the satisfaction or pleasure a user gets when purchasing the corresponding product. A rational consumer chooses the product to purchase in order to maximize the total net utility. In contrast to the predicted rating, the marginal utility of a product depends on the user's purchase history and changes over time. According to the Law of Diminishing Marginal Utility, many products have the decreasing marginal utility with the increase of purchase count, such as cell phones, computers, and so on. Users are not likely to purchase the same or similar product again in a short time if they already purchased it before. On the other hand, some products, such as pet food, baby diapers, would be purchased again and again.

- Traditional recommender systems aim to recommend the right product to a user. In fact the effectiveness of a recommendation also depends on the time of the recommendation. Let us take a user who just purchased a laptop as example. She may purchase a replacement battery in 2 years (assuming that the laptop's original battery often fails to work around that time) and purchase a new laptop in another 2 years. In this case, it is not likely for the recommender system to get positive user feedback by recommending a new laptop or a replacement battery right after the user purchased the new laptop. It could hurt the user's satisfaction of the recommender system if she receives a potentially right product recommendation at the wrong time.

1.2 The Goal of Session-Aware Recommender System

In this dissertation, the author proposes the **session-aware recommender system** in the e-commerce domain. A e-commerce *session* starts as soon as the user enters the site and ends after the user leaves the site or after the user has been inactive for more than k minutes (k is usually set as 30). A session-aware recommender system has all user behaviors (including search, click, purchase, etc.) in record. Such system makes recommendations based on the user's short-term behavior within a session, as well as the user's long-term behavior across multiple sessions. It focuses on modeling a user's purchase intention in each session. By recommending the right product at the right time, the session-aware recommender system could enhance the users' conversion rate and their satisfaction/utility.

The session-aware recommender system is inspired by theories in computer science, economics and statistics. The advantages of the system include the following:

Identify the user's purchase intention with short-term user behavior within a session:

The first advantage of the session-aware recommender system is its capacity to predict the user's purchase intention within a session, given the user's short-term behavior (search, click, etc.). The system leverages a user's heterogeneous information to predict the user's goal in the session. The purchase intention can be further analyzed in the category level, in the purchase state level and in the product level. The intention in the category level indicates which category the user would make a purchase in. The intention in the purchase state level indicates whether the user would do a re-purchase (i.e., to purchase a product that she purchased before), a variety-seeking purchase(i.e., to purchase a new product in the category that she purchased before), or a new purchase (i.e., to purchase a new product in a new category). Finally the intention in the product level indicates which product the user is likely to purchase given the category intention and purchase state intention. The predictability of a user's purchase intention within a session could help the system to do more targeted recommendation and related marketing campaigns.

Identify the follow-up purchase intention with user's purchase history across sessions:

When there is no user short-term behavior in a session, recommender systems make recommendations based on the user's behavior in previous sessions. The second advantage of the session-aware recommender system is its capacity to predict the user's purchase intention of the follow-up purchase, given the user's long-term purchase history across sessions. Traditional recommender systems leverage the user's purchase history to recommend products that the user likes or products that the user tends to give high ratings. As we described, a rational user doesn't make the purchase decision based on his predicted rating of a product. Instead a rational user makes the purchase decision based on a product's marginal net utility. Thus the session-aware recommender system should identify the user's purchase intention based on each product's marginal net utility for the user in the session. With such capacity to predict, the session-aware recommender system is able to enhance the users' conversion rate.

Identify the right time to make follow-up purchases across sessions: The third advantage of the session-aware recommender system is its capacity to predict of the right time for a user to make follow-up purchases. Traditional recommender systems focus on finding the right product to recommend. Yet another key factor is to find the right time to recommend. It is an important factor in making good recommendations. In the push-based system, the system would know when is the best time to send a campaign email to the user. In the pull-based system with no query, the system recommends products that are most likely to be purchased at the current time. It helps the system to enhance the user's satisfaction of the recommendations.

1.3 Contributions of the Dissertation

The dissertation is a comprehensive study of recommender systems in the e-commerce domain. We propose the research direction of studying the session-aware recommender system in the e-commerce domain. Different from traditional recommender systems, the session-aware recommender systems in the e-commerce domain focus on predicting a user's purchase intention within a session and across sessions. First, it helps the e-commerce website to make better recommendations, which enhances the user's satisfaction of the system. Second, the e-commerce website could design better marketing campaigns by knowing the user's purchase intention. We demonstrate the effect of the system by evaluating it with a real-world e-commerce data from shop.com.

The technical contributions of the dissertation include the following:

Within-session recommendation with heterogeneous information: We first explore how to integrate the complementary information (e.g. the user's purchase information and query information) within a single session to build a unified recommender system. We analyze the available information for the unified model, including the user's history-related information, the search-related information and the product's marketing-related information. We then propose and evaluate three unified models to integrate different pieces of information. The first basic model is based on the multinomial regression. The model is widely used in a multi-class classification task. It predicts the user's product choice given a set of product candidates. The second model is based on the gradient tree boosting algorithm which is more advanced and complex. It is widely used in the search field and learning-to-rank tasks. It predicts the probability of a user purchasing a product. These two models work as black box, which does not provide clear explanations of a user's product purchase decision. In order to better understand the user's purchase intention, we propose a new graphical model as our third unified model. It explicitly models the user's categorical choice, purchase state (repurchase, variety seeking, or new purchase) in addition to the final product choice. Experiments on a data from an e-commerce website (shop.com) show that all unified models work better than the basic search or recommendation systems on average, particularly for the repeated purchase situations. In addition, the graphical model can predict a user's categorical choice and purchase state reasonably well, while the other two models work as black boxes. The insight and predicted purchase state may be useful for implementing the user-state specific marketing and advertising strategies.

Cross-session recommendation of follow-up purchase: To better match users' cross-session purchase decisions in the real world, we study how to recommend products with the highest marginal net utility in e-commerce sites. Inspired by the Cobb-Douglas utility function in consumer behavior theory, we propose a novel utility-based recommendation framework. The framework can be utilized to revamp a family of existing recommendation algorithms. To demonstrate the idea, we use Singular Value Decomposition (SVD) as an example and revamp it with the framework. We evaluate the proposed algorithm on an e-commerce (shop.com) data set. The new algorithm significantly improves the base algorithm, largely due to its ability to recommend both products that are new to the user and products that the user is likely to repurchase.

Cross-session recommendation at the right time: We further study how to make right follow-up purchase recommendation at the right time. We adapt the proportional hazards model in survival analysis and propose the new *opportunity model* in e-commerce. The new model estimates the joint probability of a user making a follow-up purchase of a particular product at a particular time. This joint purchase probability can be leveraged by recommender systems in various scenarios, including a zero-query pull-based scenario or a proactive push-based email promotion scenario. We evaluate the soundness of the opportunity model with multiple metrics. Experimental results with a real-world e-commerce website(shop.com) show that it has decent predictability of the user's follow-up purchase behavior at the particular time. In addition, the opportunity model significantly improves the conversion rate in pull-based systems and the user satisfaction/utility in push-based systems.

1.4 Outline of the Dissertation

The rest of the dissertation is organized as following: Chapter 2 discusses the state-of-art related work, including the traditional approaches in recommender systems, more advanced approaches with hybrid methods and general evaluation methods of recommender systems. It gives readers some general background information of recommender systems. Chapter 3 describes how the unified recommender system leverages heterogeneous user information within a single session. Chapter 4 describes the marginal net utility framework that we propose to make recommendations of follow-up purchases across sessions. Chapter 5 describes the opportunity model that we design to find the right time to recommend the right product. Chapter 6 summarizes contributions of the dissertation, as well as its limitation and future research directions.

If readers are familiar with recommender systems, they can skip the introduction of related work in Chapter 2. If they are interested in the specific model of with-in session recommendation and cross-session recommendation, they can read Chapter 3, Chapter 4, Chapter 5 respectively. Otherwise, they can read Chapter 6 directly to get a general idea.

Parts of the dissertation have been published in conferences. Chapter 3 is based on a paper published in the 8th Asia Information Retrieval Societies Conference (AIRS) in 2012 [86]. Chapter 4 is based on a paper published in the 34th annual ACM Conference on Research and Development in Information Retrieval (SIGIR) in 2011 [84], as well as a paper published in the 5th ACM conference on Recommender systems (RecSys) in 2011 [83]. Chapter 5 is based on a paper published in the 36th annual ACM Conference on Research and Development in Information Retrieval (SIGIR) in 2013 [85].

Chapter 2

Related Work

In this section, we review some background information of recommender systems in the e-commerce domain. The major directions of recommender systems are the content-based approach and the collaborative filtering approach. The former one utilizes the item's content and the latter one focuses on using the user's behavior history. We first describe the content-based methods in Section 2.1, followed by the review of collaborative filtering methods in Section 2.2. We then discuss some related work of recommender systems with temporal factor in Section 2.3, which is related with our dissertation. In the last part, we introduce the basic evaluation metrics of recommender systems in the e-commerce domain in Section 2.4.

2.1 Content-based Recommender Systems

In the content-based approach, the idea is to detect items that are most "similar" to the user's existing profile. A user profile is composed of his/her previous transaction history, such as what he/she viewed or purchased before. After the user profile is set up, how to determine similarity between an item and the profile is the key challenge. Various approaches [41, 55, 56, 96] have been developed, such as cosine similarity with TF.IDF term weight, Bayesian classifiers, clustering, etc.

Our research is in the e-commerce domain, which indicates that content information is limited. Every product is only associated with title, image and possible limited description and user reviews. With such constraints, the effectiveness of content-based recommender systems is bounded. On the other hand, content-based approach can help to make the recommendation list more comprehensive. It is likely that some appropriate products or newly-launched products were never purchased or viewed in the history data. In this case, collaborative filtering approach can never discover such products. Content-based approach is very useful to increase the recommendation coverage.

2.2 Collaborative Filtering Recommender Systems

Collaborative filtering approaches are based on the following assumption: users that have similar tastes on some items may also have similar preferences on other items. Thus the main idea is to utilize behavior history from other like-minded users, in order to find good recommendations for the current active user.

The typical user behavior history is presented as a $m \times n$ user-item matrix R. Accordingly, there are m users and n items in the dataset. Every entry $r_{i,j}$ is the $user_i$'s rating for $item_j$. The rating can be in the form of binary action (view a product or not), or numerical rating (integer between 1 and 5). The rating $r_{i,j}$ can be provided by the user in an explicit way [5]. Or it implicitly reflects user's preference to this product, from his/her transaction history [25]. The user-item matrix is usually incomplete and sparse. For an active user u_a, the goal is to select top N products from the item set $I = \{i_1, i_2, ..., i_n\}$. In e-commerce, the system's recommendation, I_r, usually has no overlap with what the user already purchased, I_p.

The process of rating-oriented collaborative filtering approaches [93, 23, 38] simplifies the problem as predicting the missing data in the user-item matrix, given the observed data. The first step is to make prediction of items that user didn't purchase or viewe before. The second step is to select top N recommendations according to the predicted rating. Besides, recommendation could be made directly from the user's behavior history [48], which skips the prediction part and the possible error in that part. There are several major directions of collaborative filtering systems. We discuss some of them in the following sections.

2.2.1 Vector-Space Model

In the user-item matrix, each row is naturally a user vector and each column is an item vector.

The **user-based** method utilizes the user vector, and belongs to the **memory-based** approach in literature [36, 66, 71, 62, 8, 26]. The first step is to identify most similar users for the active user u_a. Then all these neighbors' preference is combined to compose a set of items. The final step is to select top-N recommendations from the set. Two challenges are how to determine the similarity between users and how to select top-N recommendations from the set. To tackle the first challenge, various different similarity methods have been applied to user's vector, $< u_{a_1}, u_{a_2}, ..., u_{a_n} >$. Similarity methods include cosine similarity, Pearson correlation coefficient functions, so on and so forth [66]. To tackle the second challenge, one direct way is to select top-N items that were purchased most frequently by neighbor users. Another way is to use the weighted sum in aggregation, which means that ratings from more similar users contribute more. In addition, if the user's different rating scale is considered, deviation from the average rating can be utilized instead of the raw rating.

Though the memory-based method has a wide popularity, it has some drawbacks. First of all, the nature sparsity of the user-item matrix might lead to poor recommendation quality. Since each user only has few ratings, it is hard for the system to find similar neighbors with high quality. Another major weakness is its poor scalability. The computation cost grows fast with the number of user and the number of items. In e-commerce systems, if the recommendation is based on the user's real-time actions, all computation has to be taken online. It makes the memory-based method not realistic in real world applications. The clustering-based approach [78, 53, 8] provides a worthwhile trade-off between accuracy and scalability. Users are clustered into different groups according to the different level of participation. For the active user, other users in the same cluster can be treated as candidates of similar neighbors. Thus the number of users to consider is reduced from all population to the size of cluster. Another way is to average the opinion of all users in the same cluster, considering their participation in the cluster.

The **item-based** method utilizes the item vector, and belongs to the **model-based** approach. Instead of finding similar user neighbors, this method directly finds similar item neighbors for items in the user vector. Then top-N recommendations can be selected from all similar item neighbors [65, 34, 17, 46]. Similarly to the user-based approach, two challenges include how to determine the item-item similarity and how to select from all item neighbors. To tackle the first challenge, the similarity can be computed by cosine similarity, correlation-based similarity, conditional probability-based similarity, etc. To tackle the second challenge, the idea is to utilize both active user u_a's existing ratings and the similarity of item neighbors to make the prediction. It can be the weighted sum of the user's existing rating to an item, together with the similarity between this item and its neighbors. The item-based method significantly improved the recommendation quality and reduced the computation cost, compared with the user-based method. When there is enough training data, item-based methods are usually proved to be as good as or better than other collaborative filtering methods.

The **feature-based** method [25] is an extension to the item-based method to solve the "cold-start" new product problem. The new product problem is caused by the data sparsity nature of the user-item matrix. It is likely that some good products were not purchased before. In the traditional item-based method, such products could not be recommended. In the feature-based method, all products are abstracted into feature vectors. The assumption is that "people who purchased products with some features like A also purchased other products with features like B". After top-N items are selected, they are abstracted into the feature level and common features are aggregated. Items with matching features are recommended to the user. In this case, new products with no previous transaction history can also be recommended to the user.

Instead of using only the user vector or the item vector, a unified model is developed to combine these two methods [88]. Each rating is treated as a weighted combination of rating from similar users, similar items, and similar items from similar users. Experiments showed that the unified framework could improve the prediction accuracy and deal with the data sparsity problem.

2.2.2 Probabilistic Model Based Approach

In order to reduce the online computation cost, many model-based approaches have been developed in recent years. Model-based approaches learn a model from the existing user-item matrix, and the model is further utilized in the prediction step. In short, model-based approaches sacrifice some level of precision for better online scalability. The offline model building process could be quite expensive. We discuss several typical methods for dyadic data (i.e.

domain with two finite sets of objects). Similar methods can be found for polyadic data (i.e., data with more elements).

LSI is a widely-used technology to reduce matrix dimension. It is mainly utilized to solve the data sparsity problem in collaborative filtering approaches [67]. The underlying matrix factorization approach is *Singular Value Decomposition(SVD)*. The original user-item matrix is factored as $R = U \times S \times V'$. The dimension of the original R matrix is $m \times n$, and the dimension of the reduced U matrix is $m \times r$ while that of the reduced V matrix is $n \times r$. The matrix S can be reduced to S_k, which contains the k largest diagonal values. Accordingly, U is reduced to U_k and V is reduced to V_k. SVD can help to capture the latent relationship between user and item. After U_k and V_k matrix are formed, the R matrix can be re-constructed as $R_k = U_k \times S_k \times V_k$. We can further perform prediction and recommendation on the hidden representation of the user vector $U_k {S_k}^{1/2}$ and the item vector ${S_k}^{1/2} V_k'$. Though the offline computation is more expensive than the online computation of correlation, it won't affect the speed of online recommendation.

Based on a more solid statistical foundation, PLSA is a statistical latent class model (*Aspect models*) [30, 29, 28]. The key idea is that users and items are independent given the latent variable z. The probability model for $(user, item)$ pair follows the rule $P(u,i) = \sum_{z \in Z} P(z)P(u|z)P(i|z)$. In this equation, $P(u|z)$ is the user distribution given z, and $P(i|z)$ is the item distribution given z. The latent factor z has its own prior probability $p(z)$. Note that users and items share the same latent variable and they do not have their own clusters. Parameter estimation is implemented by the EM algorithm and the rating prediction is to maximize the posterior probability. In this model, each rating for a $(user, item)$ pair is a convex combination of all latent factors. $P(u,i,r) = \sum_{z \in Z} P(z)P(u|z)P(i|z)P(r|z)$. For binary rating, $P(r|z)$ can follow a multinomial distribution [29] while for numerical rating, $P(r|z)$ can follow a gaussian distribution [28].

In addition to using one latent factor model, *Flexible mixture models(FMM)* [74] introduces two separate latent factors, one for the user cluster and another one for the item cluster. Thus the probability model turns out to be

$$P(u,i,r) = \sum_{z \in Z} P(z_u)P(z_i)P(u|z_u)P(i|z_i)P(r|z_u,z_i)$$

The performance is better than the aspect model since FMM separates the user class and the item class.

The further refinement leads to the *Decoupled models* [32]. It takes into consideration of the users' difference in rating behavior. Users with same preference may have different rating habits. In decoupled models, each rating is generated both by the preference factor and the rating behavior latent factor. Experiments in [32] showed that the improvement of decoupled models was not obvious from FMM.

The first weakness of PLSA model is about the *new* item. It is not natural to assign latent topic to a new item, which makes it not a good generative model. Besides, the number of parameters grows linearly with the number of latent topics. It causes a severe over-fitting problem.

To improve the PLSA model, researchers utilized Latent Dirichlet Allocation model(LDA) [7, 49]. In LDA model, the latent variable z is drawn from the multinomial distribution of θ. θ is drawn from the dirichlet distribution. Thus the mixture weight of the latent variable becomes to be a more flexible hidden random variable.

Multiple multiplicative factor model(MMF) [50] introduces the factor expression level. Each rating can be influenced by the whole vector of latent variables in MMF, which is different from PLSA and LDA model. Besides, some latent factors are allowed to have zero influence, which is different from a mixture model. The factor expression level z^n is taken from a Bernoulli distribution. Given a factor expression level z^n, the data likelihood is defined in equation 2.1.

In the equation,$P(v|k)$ follows a basic multinomial distribution.

$$P(v|z^n) = \frac{\prod_{k=1}^{K} P(v|k)^{z_k}}{\sum_{v'}^{V} \prod_{k=1}^{K} P(v'|k)^{z_k}} \tag{2.1}$$

The probability of a latent factor vector is $P(Z = z|\eta) = \prod_{k=1}^{K} P(Z_k = z_k|\eta_k)$. In collaborative filtering, we can treat the latent factor as the "user attitude" [50]. For each rating v_i, some attitudes influence it while some others do not. This is the major contribution of MMF, compared to mixture models.

2.2.3 MDP (Markov Decision Processes) Based Recommender Systems

One promising research direction is MDP-based recommender systems [70]. In this work, the sequential recommendation nature plays an important role, which was negligible in other related work. The MDP-based collaborative filtering treats the problem as a sequential decision problem and utilizes *Markov decision process (MDP)* to solve the problem. The model has four essential factors, a set of states S, a set of actions A, a reward function Rwd and a state-transition function tr. In the domain of recommender systems, S was the most recent k items that a user bought. (k was set as 3 in [70]). Each action a in A represents a recommendation action. The policy π maps from a state to an action, i.e., $a = \pi(s)$. The reward function Rwd can be set as the net profit for a state. The state-transition function tr reflects the probability a user with current state $< x_1, ..., x_k >$ will select x' next, denoted as $tr(< x_1, x_2, ..., x_k >, < x_2, ..., x_k, x' >)$. tr needs to be learnt from the data.

The first step is to construct a predictive model to predict tr, which predicts the item a user will buy next. Since it does not consider the effect of recommendation(action), it is by nature a Markov chain problem. After that, the ultimate goal is to find the optimal policy $\pi(s)$.

Then we can derive the item to recommend according to the user state. The value function of a policy $\pi(s)$ is defined as

$$V^{\pi}(s) = Rwd(s, \pi(s)) + \gamma \sum_{s_j \in S} tr(s, \pi(s), s_j) V^{\pi}(s_j)$$

The optimal policy π^* can provide the maximal value V^*.

$$V^*(s) = max_{a \in A}[Rwd(s, a) + \gamma \sum_{s_j \in S} tr(s, a, s_j) V^{\pi}(s_j)]$$

In [70], they utilized the policy iteration to compute the optimal policy π^*, and the corresponding V^*. In each iteration,

$$V_i(s) = Rwd(s, \pi_i(s)) + \gamma \sum_{s_j \in S} tr(s, \pi_i(s), s_j) V_i(s_j)$$

$$\pi_{i+1}(s) = argmax_{a \in A}[Rwd(s, a) + \gamma \sum_{s_j \in S} tr(s, a, s_j) V_i(s_j)]$$

The iterative process will converge to an optimal policy. There are several essential issues in this approach, such as how to set the initial value of tr, how to update the model online, and how to balance between exploration and exploitation. All these factors need to be designed carefully in the recommender systems.

2.2.4 Methods with Matrix Factorization

Research on collaborative filtering algorithms reached a peak due to the 1 million dollar Netflix movie recommendation competition [5]. Factorization-based collaborative filtering approaches [16, 37, 84, 72], such as the regularized Singular Value Decomposition performed well on this competition. It is possibly better than Netflix's own well-tuned Pearson correlation coefficient algorithm. A common characteristic of these models is the introduction of user latent factors or/and item latent factors to solve the data sparsity issue.

Let's take SVD with matrix factorization as an example. It is the basis of several recommendation algorithms based on latent factors, which have been proven to work well on benchmark recommendation datasets including the Netflix dataset [5]. The basic algorithm operates over a user-product matrix $P_{M \times N}$. It assumes that each entry $P_{u,i}$ in the matrix P can be estimated using the following form:

$$\hat{P}_{u,i} = q_i^T p_u \tag{2.2}$$

where q_i and p_u are vectors, which are the hidden representation of product i and user u. These vectors can be estimated based on all given entries in $P_{M \times N}$.

The value of $P_{u,i}$ in the observed matrix $P_{M \times N}$ is determined by the user purchase history. In the e-commerce domain, $P_{u,i}$ is usually set to be a unary value that indicates whether user u purchased product i or not [27]. Existing work finds that unary value is more suitable than numerical value, i.e., the number of times the user purchased the product [68].

There are many extensions (AsySVD, SVD++, etc.) to the basic SVD in the literature [16, 37]. In addition, Agarwal and Chen [1] proposed a regression-based latent factor model which works in a generalized linear model framework. It generalized the unobserved latent factors from the observable user and item features.

2.2.5 Other Model-based Approaches

Graph-based methods are developed to go beyond nearest neighbor approaches [2]. Users are represented as nodes in the graph. During the recommendation process, all nearby nodes of the active node contribute to the final recommendation list. Even if a user didn't rate the common item with the active user, his/her ratings might have influence on the final recommendation list. It utilizes the transitivity feature in the graph, which produces better predictions than nearest neighbor approaches.

Breese et. al introduced *Bayesian networks* [8] to make the prediction. Each node in the network corresponds to the rating of an item. In addition, each node has a set of parent nodes which are treated as the predictors of its rating. The Bayesian network is learnt from the training data in offline process, which leads to a small model with fast online computation. The result in [8] showed that the performance of Bayesian network is better than the cluster model and is as good as the user-based model. One limitation of Bayesian network is that the rating must be binary. Another limitation is that it performs well only if the user preferences do not change frequently.

Another type of methods mines the user-item matrix to detect *association rules* [52, 45]. During the recommendation process, association rules are applied to the existing items of active user u_a. Each rule is assigned with a weight, which reflects the similarity between active user's existing items and the antecedent of association rule, as well as the confidence of the rule itself.

2.3 Recommender Systems with Temporal Effect

Factoring time has received much research attention [38, 83, 101, 64, 73, 47, 95, 19, 40, 90, 75].

In [18], the item-item similarity was weighted by the time distance between two items. They compared between the exponential decay function and the logistic function. Then they chose the exponential function in their paper. Exponential function is considered to be more suitable since recent data should be emphasized more. The time function was defined as $f(t) = e^{-\lambda \cdot t}$ where the decay factor $\lambda = \frac{1}{T_0}$. The parameter T_0, "half-life parameter", could also be calculated by $F(T_0) = (1/2)f(0)$. In [18], items for each user were clustered into different groups, and proposed learning algorithm to learn T_0 for every cluster for each user. The experiments showed that it performed better than pure item-based collaborative filtering.

Another focus is about the drift of the user's preference over time [38, 91, 60]. Rendel et. al [60] proposed a factorized personalized model that subsumes both a common Markov chain and the normal matrix factorization model. An important work by Koren [38] thoroughly analyzed the temporal effect in collaborative filtering approaches. Instead of using the traditional time-window or instance-decay method, they tracked the time-changing behavior through the whole data. They added the temporal effect on two models, the matrix factorization and the item-based similarity. They addressed the following temporal factors: $b_u(t)$, the gradual change of user's rating over time and the sudden user rating behavior change; $b_i(t)$, the change of item's popularity over time; the periodic effect on the rating; the change of the user's rating scale. The rating predictor was modified to include these temporal factors and the learning algorithm was designed accordingly. With a unified framework, this work combined various gradual and sudden changing user-dependent concepts, as well as item-dependent concepts. It showed significant improvement over the baseline predictor without considering temporal effect. They reported the best performance on the widely-used Netflix movie data [5].

Some recent work studied modeling the time interval between purchase orders in the e-commerce domain [83, 101]. Wang et al. [83] discovered different post-purchase behavior in different time windows after purchase. Zhao et al. [101] used the purchasing time interval to improve the temporal diversity [60]. The time interval and the corresponding purchase probability is modeled inside the framework of a utility-based recommender system [84]. The hybrid system takes the time interval into consideration when ranking all candidate items.

2.4 Evaluation Metrics of Recommender Systems

There are several metrics to evaluate recommender algorithms in the literature [27]. Considering the usage scenario in a typical e-commerce web site, the ranking of all recommendations is more important than the rating prediction. Instead of using some common rating prediction accuracy measures (Root Mean Square Error, etc.), we evaluate all algorithms in the context of a ranking task [16].

- Each order $D_{u,t}$ corresponds to a testing point, which is uniquely identified by a (user, order time) pair. Let $S_{purchased}$ be the set that contains all products in this purchase order.
- Rank all products according to their predicted marginal net utility for user u at time t. Let $S_{K,recommended}$ be the set that contains the top K products.

Then we have:

$$recall@K = \frac{|S_{purchased} \cap S_{K,recommended}|}{|S_{purchased}|} \tag{2.3}$$

$$precision@K = \frac{|S_{purchased} \cap S_{K,recommended}|}{K} \tag{2.4}$$

Conversion rate, a commonly-used metric in e-commerce, is used as an additional evaluation metric in our experiments. If the user purchased at least one product from the recommended top K list, we consider that the user has converted from a browser into a buyer. The calculation of conversion rate for one testing point is shown in the following equation.

$$conversion\ rate@K = \begin{cases} 1 & S_{purchased} \cap S_{K,recommended} \neq \varnothing \\ 0 & otherwise \end{cases} \tag{2.5}$$

Conversion rate reflects whether a user receives at least one good recommendation.

Chapter 3

Within-Session: Unified Recommender System with Different User Information

As we discussed in Section 1, traditional recommendation approaches learn a user's long-term preferences from the user's purchase or rating history. They work well in domains such as movie and music recommendations. Yet their effectiveness in the e-commerce domain is limited [13]. A consumer often makes purchase decisions based on various short-term or contextual reasons that are unknown to the e-commerce system. It is extremely challenging for recommender systems to predict under this circumstance. Fortunately an e-commerce website is not limited to the user's purchase history. The short-term user behavior within a session includes issuing a query, clicking to view a product, adding a product to cart and so on. All this different information could help the recommender system to better understand the user's purchase intention in the session. Here we propose to build a **unified recommender system** to capture both the long-term preferences and short-term needs of a user within a session.

To demonstrate the effect of incorporating user's short-term behavior (such as issuing a query), we describe the unified model in a search session. A search session is a session in which the user has issued one or more queries before she makes a purchase decision. These queries provide much information about the consumer's current purchase intention, and could benefit the recommender system if used appropriately. To do so, the system incorporates the user's purchase history and her short-term search information within the session. We can roughly break down different pieces of information into three groups: the user's history-related information, the query-related information, and the product's marketing-related information. The user's history-related information, which is typically derived from a user's prior history and utilized by a traditional recommender system, reflects the user's long-term preferences and characteristics. The query-related features capture a product's relevance with the user's queries, which can be generated by the basic text-based retrieval algorithms such as BM25 and language models. This feature group largely reflects the current purchase intention of the user. The product's marketing-related features (popularity, category, price, promotion, etc.) capture the marketing environment and have been found to influence a consumer's purchase decision by marketing researchers.

We propose three unified models to integrate this information and generate recommendation results for a user in a search session. The first basic unified model is the multinomial regression function. The second model uses the gradient boosting tree (GBT) to integrate all three groups of information mentioned above. This model has been proven to be very successful in the web search community. Most of the top performing algorithms in the *Learning To*

Rank Challenge[1] are variations of GBT [9]. Although these two models have decent prediction performance, they work as black boxes. In order to better understand and explain the user's purchase intention, we propose a third model, a new graphical model, to jointly model a user's unobservable categorical choice, purchase state (repurchase, new purchase, or variety seeking) and product choice. The new graphical model provides more insights about a user's purchase intention and decision process.

We compare our proposed models with a variety of search and recommendation algorithms on an e-commerce dataset collected from shop.com. The experimental results demonstrate the value of *a unified recommender system* with better prediction power. It achieves significantly better conversion rate than the basic search and recommendation algorithms. We also analyze the contribution of each feature group and how different models work in different purchase states in details.

The following sections describe our efforts towards evaluating and developing unified recommendation algorithms. Section 3.1 describes the problem and analyzes three groups of information that can be used by the unified recommender system. Section 3.2 describe the proposed unified models in details. Section 3.3 and Section 3.4 describes our experimental methodology and results. In the end, Section 3.5 concludes the contribution of this work.

3.1 Problem Analysis

3.1.1 Problem Definition

If a user issues one or more queries before she purchases some products, we call the user session a *search session*. A search session may contain one or more orders. Table 3.1 shows

[1]http://learningtorankchallenge.yahoo.com/workshop.php

some sample sessions for one user. Session 10 is a search session for this user. Yet session 11 is not a search session since there is no query before the purchase of $p3$.

This work focuses on generating a good recommendation list for a user in a search session. After a user issues one query and before she makes a purchase, our system generates a ranked recommendation list and presents it to the user to choose from. In the rest of this work, the following notations are used:

- u: the user index;
- i or j: the product index;
- c: the category index.
- t: time; Many features are dependent on the time t. For example, the similarity between product i and the user's previous purchases depends on the user's purchase history at time t. For simplicity, we drop t in the notation of most features.
- Q: the user's current query/queries at time t. It contains all queries issued in the current session after the last purchase order. For the example in Table 3.1, Q contains $q1$ and $q2$ at time *2011-11-01 13:33:00.* Yet after $p1$ is purchased, $q1$ and $q2$ are no longer part of the current query, and Q contains only $q3$ at time *2011-11-01 14:25:00.*
- $\mathbf{x_{u,i}}$: the feature vector of the candidate product i for user u at time t. This vector is given and more details are discussed in the next section.
- $y_{u,t}$: user u's purchase decision at time t. If $y_{u,t} = i$, it indicates that user u purchases product i at time t.

Table 3.1: Sample search sessions for one user.

SessionID	Time	Action	Content
10	2011-11-01 13:30:00	query	*q1*
10	2011-11-01 13:32:20	query	*q2*
10	2011-11-01 13:59:02	purchase	*p1*
10	2011-11-01 14:24:00	query	*q3*
10	2011-11-01 14:45:03	purchase	*p2*
11	2011-11-02 09:23:34	purchase	*p3*

3.1.2 Information for the Unified Model

On a high level, we classify all information available for the unified system into three groups: the query-related information, the product's marketing-related information and the user's history-related information. In the unified model, every candidate product's feature vector $\mathbf{x_{u,i}}$ is composed of all three groups of information. $\mathbf{x_{u,i}} = \{\mathbf{x^H}, \mathbf{x_i^Q}, \mathbf{x_i^P}\}$

User's History-related Information

User history is essential in building a successful personalized unified system. The history-related information can be extracted from the user's prior purchase history and the query log. To study the contribution of different information, we consider three types of user's history-related information.

The **first** type of information $\mathbf{x_u^H}$ reflects the user's general characteristics in the current session. It is not related to each candidate product. Features include time difference between the current order and the last order, time difference between the current session and the last session, similarity between the current query Q and the user's previous search queries in history, the relevance between the current query Q and previous purchases and so on. This information might be useful for inferring a consumer's purchase state (repurchase, new purchase or variety seeking). There are other features, such as the number and percentage of repurchase/variety seeking/new purchase in the user prior purchase history, might tell whether the user is a loyal consumer or not. The **second** type of information $\mathbf{x^H_{u,c}}$ captures the user's general characteristics in each candidate category c. For instance, a user might be a loyal consumer in the "food" category yet not in the "clothing" category. We include the number of times a product in category c was purchased by user u, number and percentage of repurchase/variety-seeking/new purchase behavior performed by this user in category c and so on. The **third** type of information $\mathbf{x^H_{u,i}}$ captures the user's interaction with each candidate product i. We include the number of times product i was purchased by user u, the total/average/maximum similarity between product i and user u's previous purchases and so on. For a unified model, the score from other basic recommender solutions such as SVD can be used as the interaction feature as well.

Query/Search-related Information $\mathbf{x^Q_i}$

The second group of information is entirely based on the user's queries Q in the search session. The queries issued before purchasing are highly related to the user's purchase intention in that session. The query-related information includes the basic static characteristics such as the length of the query, as well as the relevance score between the query and the candidate product. In this work, we generate relevance scores by varying the following three key factors: **1) Text-based retrieval algorithm.** IR researchers have proposed several text-based algorithms. In this work, we use two well-known algorithms: BM25 [63] and the language modeling approach [94]. **2) The text content of product *i*.** We consider two variations: using product title terms or using all product information, including the title, the description and the product category terms. **3) Query:** The current queries Q may contain multiple queries issued since the last purchase. A typical search engine usually ranks products based on the relevance score for the most recent query. However, the product to be purchased might be related to any query in Q. We consider the following variations of relevance scores: the score based on all terms in the combined query, the average relevance score for all queries, the maximum relevance score among all queries, the score for the first query, and the score for the most recent query. Thus the total number of relevance score based features is $|M| \times |P| \times |L|$, where $|M|$ is the number of text search algorithm options, $|P|$ is the number of product representation options, and $|L|$ is the number of query-related options. As discussed above, we have $|M| = 2$, $|P| = 2$ and $|L| = 5$ in this work. This leads to up to relevance-related 20 features in this group.

Product Marketing-related Information $\mathbf{x_i^P}$

Another important group of information is the product's marketing-related information. The marketing literature has demonstrated that marketing environments influence the consumer's decision process. Thus we extract the following factors as marketing-related features: the product's category, popularity, price, and the corresponding category's popularity. We further consider the repurchase tendency of the product and the category. Some products and categories are consumable, such as pet food and diapers. They are likely to be purchased again and again. In some categories such as clothing, the same product is unlikely to be purchased again and a user often buys a new product. We capture the difference in the repurchase tendency of the product and category by the average purchase count of a product/category per user.

3.2 Unified Models

In this section, we introduce three unified models to predict a consumer's purchase decision (i.e. which product to buy) in a search session.

3.2.1 Multinomial Regression

We first use multinomial regression as the basic unified model to solve a multi-class classification problem: predicting which product a user will purchase given a candidate set at time t. We consider a user's purchase decision as a process to choose exactly one product to purchase at time t. The approach is realistic since a user usually puts one product in the shopping cart and continues shopping for the next product. The model is widely used in the multi-class classification task with decent performance. The time complexity is relatively low. Yet it works as a black box, without clear explanations of the user's product choice.

The probability for user u to choose product i can be estimated based on all available features in $\mathbf{x_{u,i}}$:

$$P(y_{u,t} = i|\mathbf{x_{u,i}}) = \frac{exp\{\mathbf{w}^T\mathbf{x_{u,i}}\}}{\sum_{j\in\psi_{all}} exp\{\mathbf{w}^T\mathbf{x_{u,j}}\}}$$

where ψ_{all} is the candidate set that contains all products in the ranking pool for user u to choose at time t. Assuming the prior distribution of each model parameter is a Gaussian centered on zero, the optimal parameters ($\mathbf{w}$) can be learned from training data using the *maximum a posteriori probability (MAP)* estimation. To make a recommendation, we rank products based on the purchase probability $P(y_{u,t} = j|\mathbf{x_{u,j}})$.

3.2.2 Gradient Tree Boosting

We use Gradient Boosting Tree algorithm as a more advanced unified model to solve a two-class classification problem: predicting whether a user will purchase a product or not. The Gradient tree boosting learns a weighted additive model of simple *trees* from the labeled data. It is a very general and powerful machine learning algorithm that performs well on learning-to-rank tasks. This method and its variations represent the state-of-the-art approach in the well-studied web search problem. Compared to the multinomial regression model, it is more advanced with higher time complexity. It is expected to achieve better performance. Yet it also works as a black box. It is hard to explain the reason behind a user's purchase decision given the model.

As shown in [22], the loss function for a two-class classification problem is usually set to be the negative binomial log likelihood.

$$L(F, r) = log(1 + exp(-2F(\mathbf{x}) * r))$$

where F is the regression function that maps the input feature vector $\mathbf{x}$ to a binary value r. The gradient boosting tree approach approximates $F(\mathbf{x})$ as follows:

$$F(\mathbf{x}) = \sum_{m=0}^{M} \beta_m h(\mathbf{x}; \mathbf{a_m})$$

where $h(\mathbf{x}; \mathbf{a_m})$ is restricted to be a small fixed-size regression tree. The parameters β_m and $\mathbf{a_m}$ are learned incrementally to minimize the empirical loss on training data. More details can be found in [22]. The generalization error of a machine learning model is due to the bias and variance of the modeling approach. The gradient boosting tree approach reduces the bias incrementally by adding trees in the gradient direction of the loss function. Overfitting could be a problem for the basic gradient boosting trees. Therefore bagging, a variance reducing technique, is often used when training a gradient tree boosting model. We denote the gradient boosting trees trained with the bagging technique as *GBT* in the rest of this work.

Given users' purchase history, we can generate the training data set ($\mathbf{x_{u,i}}$, r), where $\mathbf{x_{u,i}}$ contains all information in Section 3.1.2 and $r = 1$ if the user purchases the product i. To generate recommendations, the learned GBT can predict the purchase likelihood for each product and rank all products accordingly.

3.2.3 Graphical Models with Purchase State

The previous two unified models work as black boxes to predict the user final product purchase. In order to better understand the user's purchase intention, we propose a new graphical model here.

Marketing researchers found that a user's purchase state can be classified into the repeated purchase, the variety-seeking purchase and the new purchase [31, 51, 6]. Firstly, these purchase states could be product specific. For consumable products such as pet food or baby diapers, users might purchase them again and again in an e-commerce website. For products such as clothing, users are more likely to switch to a new product they never purchased before. Secondly, the purchase state is user specific. Some consumers are more loyal than others. Naturally, we can tell a user's consideration set (i.e. candidate products) to be different given a different purchase state. In a repurchase state, the user is more likely to stick to her favorite products. In a variety-seeking purchase state, the user would consider new options in the category that she purchased before. In a new purchase state, the user would purchase some new products in the category that she did not purchase before. Understanding these purchase intentions (i.e. the user purchase state of the current shopping session) will help an e-commerce website provide better recommendations to consumers.

Before a user makes a purchase, e-commerce websites don't observe the user's purchase state a priori. To model this, we introduce a new graphical model to explicitly model the unobservable purchase state as shown in Figure 3.1 . Feature vectors $\mathbf{x_i^Q}, \mathbf{x_i^P}, \mathbf{x_u^H}, \mathbf{x_{u,c}^H}, \mathbf{x_{u,i}^H}$ are described in Section 3.1.2. Two variables $c_{u,t}$ and $s_{u,t}$ are introduced to capture the user's categorical choice and purchase state respectively. Categorical choice includes "digital cameras", "camera filters", "dental/oral care", "office supplies", "puzzle toys" and so on. $s_{u,t}$ can take three possible values: repurchase, variety-seeking or new purchase state. Given the query/queries Q

at time t, $c_t = k$ if Q is targeting at products in category k. If category c_t was never purchased by u before, the purchase state is always a new purchase, $s_{u,t} = 0$. Otherwise, the purchase state $s_{u,t}$ could either be repurchase $s_{u,t} = 1$ or variety-seeking purchase $s_{u,t} = -1$.

Before a user makes a purchase decision, $c_{u,t}$, $s_{u,t}$ and $y_{u,t}$ are unknown and $\mathbf{x_{u,i}} = \{\mathbf{x_i^P}, \mathbf{x_i^Q}, \mathbf{x^H}\}$ are known. After a user makes a purchase (i.e. checks out a product or put a product into the shopping cart), all unknown variables are observed. Thus each purchase decision can be treated as a training/testing data point. The likelihood of each purchase decision is:

$$P(y_{u,t}, s_{u,t}, c_t|\mathbf{x_{u,i}}) = P(y_{u,t}|s_{u,t}, c_t, \mathbf{x_{u,i}})P(s_{u,t}|c_t, \mathbf{x_u^H}, \mathbf{x_{u,c}^H})P(c_t|\mathbf{x_i^P}, \mathbf{x_i^Q})$$

Now we describe each component one by one:

The **first** component is $P(c_t = k|\mathbf{x_i^P}, \mathbf{x_i^Q})$, the probability that query Q is targeting at products in category k. We use the following multinomial logistic regression function:

$$P(c_t = k|\mathbf{x_i^P}, \mathbf{x_i^Q}) = \frac{\sum_{i\in\psi_k} exp\{\mathbf{q}^T(\mathbf{x_i^P}, \mathbf{x_i^Q})\}}{\sum_{i\in\psi_{all}} exp\{\mathbf{q}^T(\mathbf{x_i^P}, \mathbf{x_i^Q})\}}$$

where ψ_k contains all products in category k and ψ_{all} contains all candidate products.

The **second** component is $P(s_{u,t}|c_t, \mathbf{x_u^H}, \mathbf{x_{u,c}^H})$, the probability of the purchase state $s_{u,t}$ given that user u would purchase from category c_t. To estimate it, we need to consider two scenarios. If the predicted category c_t was never purchased by the user before, the purchase state $s_{u,t}$ is always 0 (new purchase). Otherwise, $s_{u,t}$ can be 1(repurchase) or -1(variety-seeking purchase). Thus we have:

$$P(s_{u,t}|c_t, \mathbf{x_u^H}, \mathbf{x_{u,c}^H}) = I_{c_t\notin H_{u,t}} \cdot I_{s_{u,t}=0} + I_{c_t\in H_{u,t}} \cdot [I_{s_{u,t}=-1} + I_{s_{u,t}=1}] \cdot P(s_{u,t}|c_t \in H_{u,t}, \mathbf{x_u^H}, \mathbf{x_{u,c}^H})$$

where $H_{u,t}$ contains all categories that user u purchased before time t. I_f is an indicator function where $I_f = 1$ if f is true. We use the following standard binomial logistic regression to

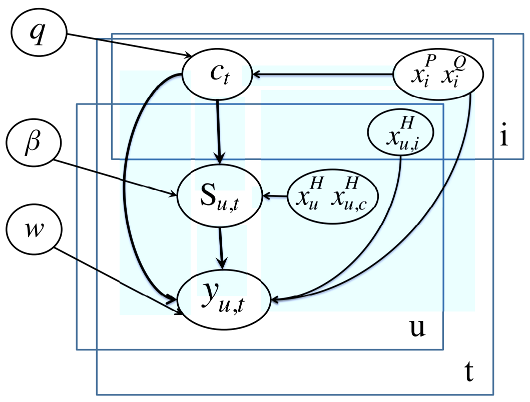

Figure 3.1: Model with the user's unobservable categorical choice, purchase state and the final product choice. q, β, w are parameters to be estimated. x_*^* are observable features. c_t is a consumer's categorical choice. $s_{u,t}$ is a consumer's states, which could be repurchase state (1), variety-seeking state (-1) or new purchase state(0). $\mathbf{w} = \{\mathbf{w_1}, \mathbf{w_{-1}}, \mathbf{w_0}\}$, which correspond to different consumer states. $y_{u,t}$ is a consumer's product choice.

model $P(s_{u,t}|c_t \in H_{u,t}, \mathbf{x_u^H}, \mathbf{x_{u,c}^H})$:

$$P(s_{u,t}|c_t \in H_{u,t}, \mathbf{x_u^H}, \mathbf{x_{u,c}^H}) = \frac{1}{1 + exp\{-s_{u,t}(\beta^{\mathbf{T}}(\mathbf{x_u^H}, \mathbf{x_{u,c}^H}))\}}$$

The **third** component is $P(y_{u,t}|s_{u,t}, c_t, \mathbf{x_{u,i}})$, the probability of user u's purchase decision given that she is in the purchase state $s_{u,t}$ and would choose from category c_t. To estimate it, we use the following multinomial logistic regression to model the user's purchase decision. We model the decision as a process to choose exactly one product to purchase at time t.

$$P(y_{u,t} = i|s_{u,t}, c_t, \mathbf{x_{u,i}}) = \frac{I_{i\in\psi_{s,c,u,t}} exp\{\mathbf{w_{s_{u,t}}}^T \mathbf{x_{u,i}}\}}{\sum_{j\in\psi_{s,c,u,t}} exp\{\mathbf{w_{s_{u,t}}}^T \mathbf{x_{u,j}}\}}$$

The major difference between this function and the one introduced in Section 3.2.1 is the candidate product set $\psi_{s,c,u,t}$. Here we only consider candidate products corresponding to the user's purchase state $s_{u,t}$ and categorical choice c_t. For example, if the category to purchase is k and the unobservable state is repurchase, i.e. $s_{u,t} = 1$, user u is expected to choose one repurchase product from the set $\psi_{1,k,u,t}$. It contains all products that user u has purchased in category k before. Depending on the value of $s_{u,t}$, the model parameter $\mathbf{w_{s_{u,t}}}$ is either $\mathbf{w_1}$, $\mathbf{w_{-1}}$ or $\mathbf{w_0}$. By having three different $\mathbf{w_s}$ parameters, the graphical model learns state-dependent multinomial logistic regression functions for each purchase state.

Assuming the prior distribution of each model parameter is a Gaussian centered on zero, the optimal parameters $(\mathbf{w}, \beta, \mathbf{q})$ can be learned from the training data using the *maximum a posteriori probability (MAP)* estimation. Stochastic gradient descent is used in the optimization process.

$$
\begin{aligned}
&(\mathbf{w_1}, \mathbf{w_{-1}}, \mathbf{w_0}, \mathbf{q}, \beta) \\
&= \arg\max L = \arg min - log(L) \\
&= \arg\min -log\{P(\beta)P(\mathbf{w_1})P(\mathbf{w_{-1}})P(\mathbf{w_0})P(\mathbf{q}) \\
&\prod_{u,t}[P(y_{u,t} = i|s_{u,t}, c_t, \mathbf{x_{u,i}})P(s_{u,t}|c_t, \mathbf{x_u^H}, \mathbf{x_{u,c}^H})P(c_t|\mathbf{x_i^P}, \mathbf{x_i^Q})]\} \\
&= \arg\min - \sum_{u,t}[\mathbf{w_{s_{u,t}}}^T \mathbf{x_{u,i}} - log(\sum_{j\in\psi_{s,c,u,t}} exp\{\mathbf{w_{s_{u,t}}}^T \mathbf{x_{u,j}}\})] \\
&+ \sum_{u,t} log(1 + exp\{-s_{u,t}(\beta^{\mathbf{T}}(\mathbf{x_u^H}, \mathbf{x_{u,c}^H}))\}) I_{c_t \in H_{u,t}} \\
&- \sum_{u,t}[log(\sum_{i\in\psi_k} exp\{\mathbf{q}^T(\mathbf{x_i^P}, \mathbf{x_i^Q})\}) - log(\sum_{i\in\psi_{all}} exp\{\mathbf{q}^T(\mathbf{x_i^P}, \mathbf{x_i^Q})\})] \\
&+ \lambda_1||\mathbf{w_0}|| + \lambda_2||\mathbf{w_1}|| + \lambda_3||\mathbf{w_{-1}}|| + \lambda_4||\beta|| + \lambda_5||\mathbf{q}||
\end{aligned}
$$

In the prediction step, we can estimate the probabilistic distribution of $y_{u,t}$ by summing over all possible values of the unobservable state $s_{u,t}$ and categorical choice c_t:

$$P(y_{u,t} = j|\mathbf{x_{u,j}}) = \sum_{c_t}\sum_{s_{u,t}} P(y_{u,t}, s_{u,t}, c_t|\mathbf{x_{u,j}})$$

To make a recommendation, we can rank products based on the purchase probability $P(y_{u,t} = j)$.

3.3 Experimental Setup

3.3.1 shop.com Dataset

We create an evaluation dataset based on the purchase record that was collected from 2004-01 to 2009-03 on an e-commerce website (shop.com). The Apache log that includes users' search history is collected from 2007-06 to 2008-11. Thus we study the unified search and recommendation model on the period of 2007-06 to 2008-11. In this work, all products that were purchased in the period of 2007-06 to 2008-11 with detailed product information(including title, description, price, category) are used. All users that made purchase(s) from these products and all orders in the period from 2004-01 to 2008-11 are included. In the entire dataset, there are 124,813 products, 244,226 users, 296,145 unique orders, i.e. (user, order time) and 341,598 unique purchases, i.e. (user, product). There are 378 secondary-level categories for all products in total. The categories are manually checked to ensure that different product purchases in one category are indeed the variety-seeking behavior. Since this work focuses on recommendation in a search session, we only evaluate models' performance when a user issued one or more queries before the order. For sessions with the Apache log information, the first 80% is used as the training data to calculate feature values, the following 10% is used as the training data to build models and the last 10% is used for testing. Sessions with no Apache log are also used as the training data to calculate feature values. Our data splitting approach is shown in Figure 3.2. Some basic statistics can be found in Table 3.2. Note that the user's history-related information is only available for users with at least one purchase before the order time. Cross-validation is not feasible in this work since only user's past behavior can be used to predict her current state and choice.

For each unified model, we select top k% products($k = 0.01$ in our experiment) in each basic recommender solution and search solution's own ranked list to put into the ranking candidate pool. In addition, products that the user purchased before the order time are also added into the pool. Each unified model learns the ranking function from the training data and ranks all products in the pool in the testing step.

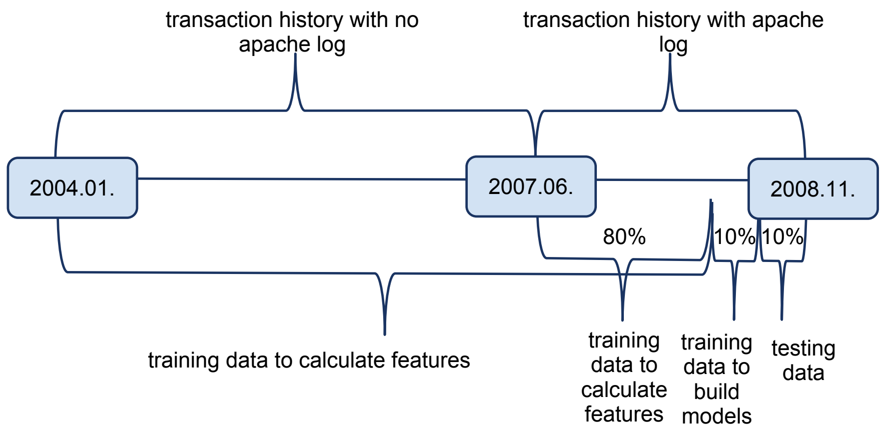
transaction history with no apache log
transaction history with apache log
2004.01.
2007.06.
2008.11.
80%
10%
10%
training data to calculate features
training data to calculate features
training data to build models
testing data

Figure 3.2: Data splitting approach

Table 3.2: Basic statistics in the shop.com dataset. New users are users with no purchase history before the order time.

Session	Instance	Number of unique instances		
		training	validation	testing
All sessions	orders made by all users	252,824	20,748	22,573
	products purchased by all users	288,208	25,522	29,286
Search sessions	orders made by all users	/	14,657	11,848
	orders made by new users	/	9,722	6,966
	orders made by users with history	/	4,935	4,882

3.3.2 Algorithms to be Compared

Our experiments compare the following three groups of system solutions (Table 3.3):

Recommender System Solutions

For basic recommender system solutions, we implement a collaborative filtering model, a content-based filtering model, and two variations of a hybrid filtering model. *PureSVD* [16] is a representative collaborative filtering model with a decent performance on standard datasets. Cremonesi et al. [16] showed that *PureSVD* has the best prediction power when treating the recommendation problem as a ranking problem. The idea is to perform the conventional singular value decomposition on the user-product matrix and estimate the rating with the matrix of a lower dimension. We use the SVD package SVDLIBC by following [16] [2]. *ContentRec* is a content-based filtering model. It estimates the score of each product as its total similarity to the user's previous purchase(s). The similarity of two products is calculated as the cosine similarity between the product titles without stop-words. *GBT.Rec* and *GBT.Rec.P* are hybrid filtering models. We use GBT to integrate the user's history-related features $\mathbf{x^{Hp}}$ that are generated from the user's prior purchase history. This feature set contains information that is typically used by the traditional recommender system, as well as scores generated by *PureSVD* and *ContentRec*. *GBT.Rec.P* further integrates the product's marketing-related information to *GBT.Rec*.

Search-based Solutions

For basic search-based solutions, we implement three methods. The first one is the well-known BM25 ranking algorithm which uses the most recent query and all terms of the candidate product to calculate a relevance score for each query-product pair. The second

[2]More algorithm details can be referred to [16]

Table 3.3: Algorithms studied in the experiments

Type	Model	Feature Set
Recommender	PureSVD	collaborative filtering with $\mathbf{x_{u,i}} = \{\mathbf{x^{H_P}}\}$
	ContentRec	content-based filtering with $\mathbf{x_{u,i}} = \{\mathbf{x^{H_P}}\}$
	GBT.Rec	$\mathbf{x_{u,i}} = \{\mathbf{x^{H_P}}$, score of PureSVD and ContentRec$\}$
	GBT.Rec.P	$\mathbf{x_{u,i}} = \{\mathbf{x^{H_P}}$, score of PureSVD and ContentRec, $\mathbf{x_i^P}\}$
Search	BM25	all terms of the product, the most recent query terms
	GBT.Search	$\mathbf{x_{u,i}} = \{\mathbf{x_i^Q}\}$
	GBT.Search.P	$\mathbf{x_{u,i}} = \{\mathbf{x_i^Q}, \mathbf{x_i^P}\}$
Unified	Multi.U	$\mathbf{x_{u,i}} = \{\mathbf{x^H}, \mathbf{x_i^Q}, \mathbf{x_i^P}\}$.
	GBT.U	$\mathbf{x_{u,i}} = \{\mathbf{x^H}, \mathbf{x_i^Q}, \mathbf{x_i^P}\}$.
	Graphical.U	$\mathbf{x_{u,i}} = \{\mathbf{x^H}$, score of GBT.Search.P$\}$.

one is *GBT.Search* which integrates features of $\mathbf{x_i^Q}$. As discussed in Section 3.1.2, these features include scores from variations of basic single search methods, as well as the query's own features (such as the number of terms in the query). The third one is *GBT.Search.P* which further integrates the product's marketing-related features $\mathbf{x_i^P}$. [3]

Unified Solutions

Three unified models in Section 3.2 are implemented and compared. The standard *Multi.U* and *GBT.U* are implemented. In *Graphical.U*, we focus on evaluating the ability of unified models in incorporating the user history related features x^H with other search/product related features. Thus we synthesize features of $\mathbf{x_i^Q}$ and $\mathbf{x_i^P}$ into a single feature: the output of the *GBT.Search.P* algorithm.

3.3.3 Evaluation Metric

There are several metrics to evaluate recommender algorithms in the literature [27]. In a typical e-commerce website, the ranking of all recommendations is more important than

[3]We found BM25 and GBT perform significantly better than language models on e-commerce in our study, thus we didn't report the performance of language models in this work.

rating predictions. Instead of using some common rating prediction accuracy measures (Root Mean Square Error, etc.), we evaluate all algorithms in the context of a ranking task [16]. In our offline experiment, only search sessions that have follow-up purchase behavior are used in the evaluation.

- Each order corresponds to a testing point, which is uniquely identified by a (user, order time) pair. Let $S_{purchased}$ be the set that contains all products in this order.
- Rank each product according to its predicted score for user u at time t, where t is right before an order time in our experiments. Let $S_{K,recommended}$ be the set that contains top K products.

We use conversion rate as a metric. It is commonly used in e-commerce, which is same as the 1-call metric in literature [11]. If a user purchases at least one product from the recommended top K list, we consider that the user has converted from a browser into a buyer. The calculation of conversion rate for one testing point is shown in the following equation:

$$conversion\ rate@K = \begin{cases} 1 & S_{purchased} \cap S_{K,recommended} \neq \varnothing \\ 0 & otherwise \end{cases}$$

Conversion rate reflects whether a user receives at least one good recommendation. We evaluate the conversion rate for the top 5 recommendations. The average value of all testing points will be used to compare among different algorithms. Significance level of 0.05 with the paired two-tailed t-test are used when comparing two methods.

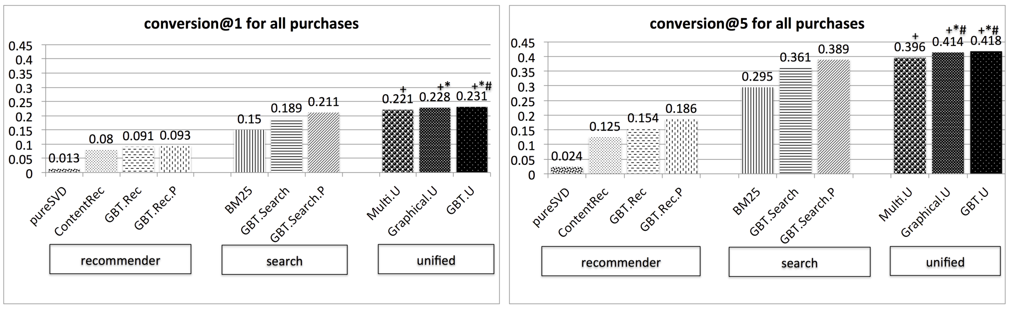

Figure 3.3: Comparison of different system's performance for conversion rate@1 and conversion rate@5. $Value^{+}$ is significantly better than *GBT.Rec.P*. $Value^{*}$ is significantly better than *GBT.Search.P*. $Value^{\#}$ is significantly better than *Multi.U*.

3.4 Experimental Results and Analysis

3.4.1 Overall Performance Analysis

Search solutions, recommendation solutions, and unified solutions are compared for user search sessions with at least one query and prior purchase (Figure 3.3). The recommendation solutions without integrating the query-related information are the least effective. On the other hand, the basic search solutions are more effective. After integrating a user's prior purchase/query history and her current query/queries, unified solutions are better. Based on the paired two-tailed t-test, the basic unified model *Multi.U* is better than the best search solution and statistically significantly better than the best recommendation solution. *GBT.U* and *Graphical.U* are statistically significantly better than the best recommendation solution, the best search solution and the basic unified model *Multi.U*. There is no significant difference between *GBT.U* and *Graphical.U* in all top 5 positions.

Why does the traditional recommender system perform not so well in the e-commerce system? We compare the density of this dataset with the other commonly used datasets of

Table 3.4: Density comparison of the shop.com data with some commonly used datasets in the field of recommender system

Dataset	# of users	# of items	# of ratings	Density
shop.com	244,226	124,813	341,598	0.00112%
movielens	6,040	3,883	1,000,000	4.26%
Netflix	480,189	17,770	100,000,000	1.18%
Yahoo! Music	1,000,990	624,961	262,810,175	0.042%

recommender system research in Table 3.4. The shop.com dataset is significantly more sparse and the huge sparsity leads to the poor performance of the traditional recommender system. This serious data sparsity issue also exists in other e-commerce website [13]. *GBT.Rec* performs as a hybrid recommender system by incorporating scores from *PureSVD*, *ContentRec* and other features generated from the user's prior purchase history. It gives some improvement over the single algorithm, yet still not good enough. In a search session with the query information available, it is essential to incorporate the information into a unified recommender and search system.

We carry some further evaluation of the basic search solutions and find that systems can serve new users with no purchase history better than users with purchase history. The conversion rate for top 1 recommendation of BM25 is 0.289 for new users with no purchase history and 0.150 for returning users with at least one previous purchase. On one hand, it is possible that a new user is directed to the e-commerce website by an external search engine such as Google or Bing product search. The user has a clearer idea about what she wants to buy with very informative queries. Contrary to that, some experienced users may be wandering around the e-commerce website with less focused purchase intention. The queries they issued are less informative which drives down the conversion. Another possible reason is that new users are

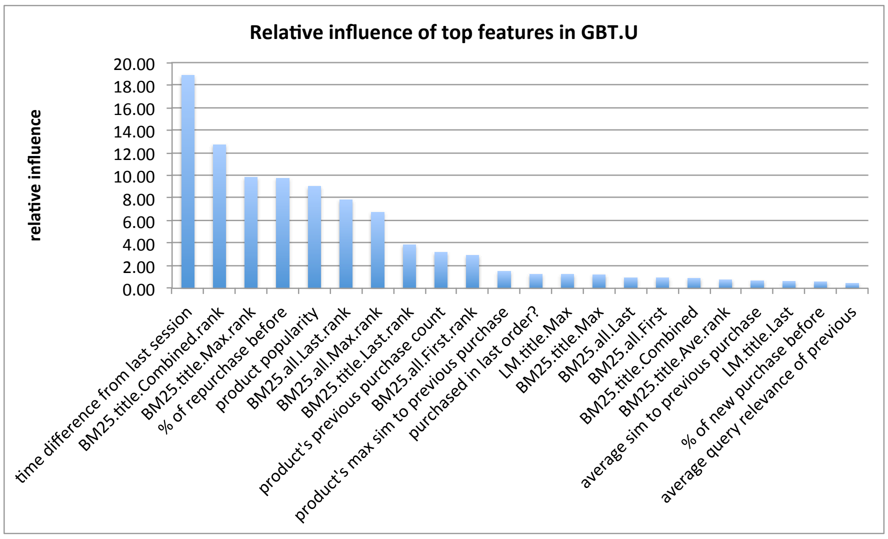

Figure 3.4: Top features with the highest relative influence in the gradient boosting trees *GBT.U*.

still naive and are more likely to purchase from search results returned by the website's search engine. This needs further exploration.

3.4.2 Performance of Different Feature Groups

As mentioned in Section 3.1.2, there are three feature groups: user's history-related features $\mathbf{x^H}$, query-related features $\mathbf{x_i^Q}$, and product's marketing-related features $\mathbf{x_i^P}$. What is the contribution of each feature group in the unified solution? Table 3.3 lists the feature set for each solution and Figure 3.3 compares performance of all solutions at top 1 recommendation and top 5 recommendation. 1)*GBT.Rec.P* outperforms *GBT.Rec* while *GBT.Search.P* outperforms *GBT.Search*. This suggests that the product's marketing-related features help to generate a better recommendation list. 2)*GBT.U* which uses all feature groups is significantly better than *GBT.Rec.P* and *GBT.Search.P*. On the one hand, it is valuable to incorporate a user's query information(both the current query and her prior query history) into the recommendation process. On the other hand, it is also valuable to integrate the user's prior purchase history into the retrieval process of a search engine.

We further look into each feature group in the *GBT.U* model. The relative influence[4] of top features are shown in Figure 3.4. Here are some interesting observations: 1) **Relevance scores** $\mathbf{x_i^Q}$ between the query Q and the product dominate as the most important feature group. The user's purchase intention in a search session is largely determined and reflected by her search queries. 2) **User history-related features** $\mathbf{x^H}$ are important factors in modeling the consumer's final decision. The most important feature is the time difference between the current session and last session. In the data, if the time difference is too high, the user is likely to make a new purchase. Thus this feature can be used as a strong signal in *GBT.U* to make a split in the tree. In addition, other user history features all play as strong signals to determine the user's purchase state, such as the percentage of repurchase and new purchase behavior in history, the similarity between the product and the user's previous purchases, etc. 3) **Product's marketing-related features** $\mathbf{x_i^P}$ both at product level and at category level, affect the user's purchase decision. Top important feature in this group is the product popularity.

We have similar findings for feature weights in the other two unified models *Multi.U* and *GBT.U*. Among user's history-related features, $\mathbf{x^H_{u,i}}$ are more important than the $\mathbf{x^H_{u,c}}$ and $\mathbf{x^H_u}$. It is not surprising since $\mathbf{x^H_{u,c}}$ is the same for all products within the same category while $\mathbf{x^H_u}$ is the same for all products.

3.4.3 Performance of Different Purchase Types

We further examine the performance of several algorithms on different purchase types: repurchase orders(14.03% of all orders), new purchase orders(73.08% of all orders) and variety-

[4]Relative influence is proposed by Friedman [22] for boosted estimates to reflect each feature's contribution of reducing the loss by splitting on the feature. More details can be found in [22].

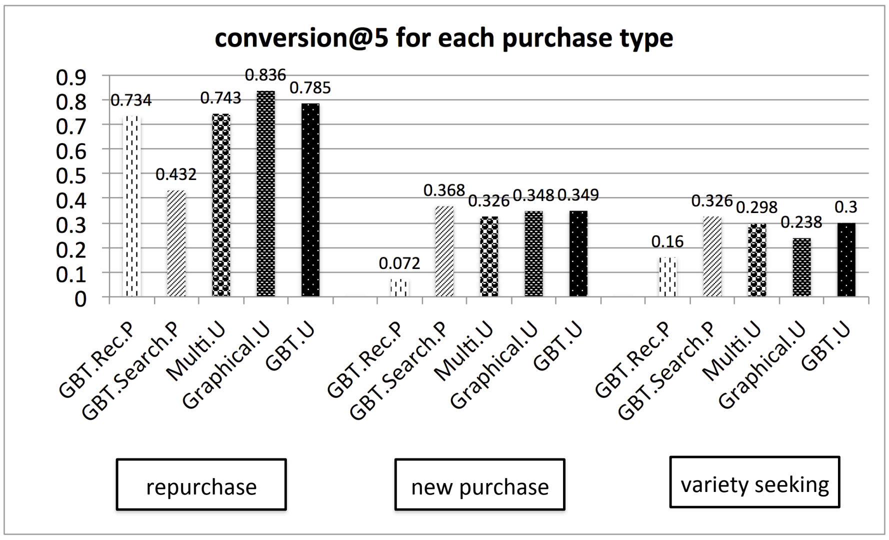

Figure 3.5: Performance of representative algorithms in conversion rate@5 for three purchase types:repurchase, new purchase and variety-seeking purchase.

seeking orders(20.54% of all orders) [5]. The algorithms to compare include the best recommendation solution (*GBT.Rec.P*), the best search solution (*GBT.Search.P*) and three unified models. Conversion rate for top 5 recommendations is shown in Figure 3.5. Here are some observations: 1) The recommendation solution performs better than the search solution for repurchases, but worse for new purchases and variety-seeking purchases. 2) Unified models perform significantly better than the recommendation solution in new purchase and variety-seeking orders. The search queries in the current session largely reflect the consumer's short-term purchase intention. Before incorporating the user's query, it is hard for the recommender system to learn the consumer purchase intention purely from her prior purchase history. 3) Unified models perform significantly better than the best search solution (*GBT.Search.P*) in repurchase orders, but worse in new purchase and variety-seeking orders. The unified models try to integrate the consumer's prior history at the cost of higher model complexity. Improving the performance

[5]A single order might contain more than one purchase type.

Table 3.5: Performance of unobservable variable estimation in the graphical model *Graphical.U*. Performance is reported for the top 1 recommendation.

Stage	# of candidates	Precision@1	Recall@1
category	378	0.652	0.595
category-purchase state	1,134	0.592	0.532
category-purchase state-product	124,813	0.228	0.191

of repurchase orders but hurting the performance of the other two is a trade-off for the unified model. All three unified models are trained to optimize in the overall performance, not in a specific purchase type (repurchase, new purchase, or variety-seeking orders). Given the limited user data in the e-commerce dataset used for this research, the cost of model complexity could hurt the performance in some situations, especially when a user's intention is to find products different from what she has purchased before (variety seeking).

In the new graphical model *Graphical.U*, we build a state dependent product purchase model. The vector $\mathbf{w_s}$ is used to predict the user's purchase decision given the unobservable state $s_{u,t}$. There are three sets of multinomial logistic regression parameter $\mathbf{w_s}$ that correspond to the product selection strategy in that state. In our analysis, we find that 1) The score from *GBT.Search.P* method is not dominant to estimate the product's purchase likelihood in the repurchase state. Yet it plays the dominant role in estimating the product purchase probability in the variety-seeking and new purchase state. 2) Both the variety-seeking state and new purchase state lead to the new product purchase, yet the weight vectors are quite different.

3.4.4 User States in the Graphical Model

Although the two complex unified models *GBT.U* and *Graphical.U* perform similarly to each other, the new graphical model *Graphical.U* has its own contribution. It explicitly models a user's purchase state and may tell us more about the user intention. In order to understand it better, we evaluate the *Graphical.U* prediction for each unknown variable (categorical choice, purchase state, and product choice) using precision and recall, and the results are shown in Table 3.5. For the variable in Figure 3.1, we first rank all candidate categories based on $P(c_t|\mathbf{x_i^P}, \mathbf{x_i^Q})$. We then rank all candidates by the joint probability of the purchase state and the categorical choice, i.e., $P(s_{u,t}, c_t)$. Finally we rank all candidate products by the purchase probability $P(y_{u,t} = i|\mathbf{x_{u,i}})$ estimated by the graphical model. As shown in Table 3.5, the prediction of the product category and the purchase state are relatively accurate. In order to improve the prediction accuracy of the product choice, we need to incorporate more marketing factors. For instance essential marketing factors such as the brand information and the promotion information will help to identify a user's variety-seeking behavior. We may have better prediction accuracy if this information is available, which could be explored in the future research.

Explicit prediction of categorical choices and purchase states allow great opportunities for the e-commerce websites to implement various marketing strategies. Marketing literature has documented that the loyalty and the variety-seeking behavior affect the effectiveness of many marketing strategies. Here are some potential use cases for each purchase state: 1) Marketers could provide related display ads to consumers who are in the repurchase state. Marketing research indicates that the advertising may impact the high loyalty consumer segments while have no impact on low loyalty consumer segments [59]. Besides marketers could provide quantity discounts to loyal consumers as they are more price sensitive in the quantity choice than non-loyal consumers [39]. 2) Marketers could implement different promotion frequencies for prominent brands and less prominent brands for variety-seeking consumers. Research has found less prominent brands may be preferred and the impact on the price promotion frequency needs to be changed accordingly for prominent brands and less prominent brands [33]. 3) If a consumer is likely to purchase in a new category, marketers can better categorize the recommendation list as this would potentially increase the user's satisfaction with their choices [54].

3.5 Summary and Contribution

We propose to incorporate the user's prior purchases and queries into a unified recommendation model for e-commerce websites. We identify three groups of features that such an engine could use: user's history-related features, query-related features, and product's marketing-related features. We propose and compare three unified models: the state-of-the-art learning-to-rank algorithm gradient boosting trees, a multinomial regression model, and a new graphical model that explicitly models a user's purchase states. Experimental results on shop.com dataset show that standard recommendation algorithms work poorly in search sessions. A user's prior purchase history indicates the user's general purchase preferences and characteristics, yet hardly reflects the user's current purchase intention. While search solutions predict reasonably well in new purchases and variety-seeking purchases, they are not as good as recommender systems in predicting repeated purchases. Instead, the unified search and recommendation models perform better than the basic search and recommendation models on average. More complex model such as Gradient Boosted Trees or graphical models work better than the simple multinomial models while there is no significant difference between the two complex models. *Graphical.U* is more interesting since it is more transparent. As we have discussed, the better explanatory power and transparency are desirable for various reasons.

Search engine is a major research focus of the information retrieval community [22, 24, 42, 76, 81, 100, 97]. It has been applied in different domains, including web search, multimedia search, chemical search, faceted search, etc. Web search is the most well-studied and heavily-studied problem. In information retrieval field, the state-of-the-art learning algorithms are based on **gradient boosting trees** [22]. It has been widely used as a unified model to combine different feature groups. Compared to the graphical model that we propose in this paper, gradient boosting tree works like a black box. Although search engines in e-commerce are widely used, the research is very limited. Guo and Agichtein [24] presented search models to capture a user's fine-grained interaction with the search results, such as the mouse events, keypress events, scroll events, and so on. Although our experiments only use the user's query and purchase history, these fine-grained user interaction features can be incorporated into our unified models. Recently, Li et al. [42] proposed a utility-based framework for product search based on combined market share data. Unlike them, we observe individual consumer's purchases or product search queries and develop a personalized query-based recommendation system to address the e-commerce product search problem. Another recent work by Li et. al [43] studied the problem of using recommender models to enhance the basic search performance with the linear sum of two models' results. Different from their work, we deeply analyze the user's behavior intention to design a graphical model with different stages. In web search, researchers have studied refinding queries [76, 77], i.e., queries that the user issued in order to find URLs that were previously clicked. The repurchase state studied in our new graphical model can be viewed as a similar behavior of the refinding behavior in the web search. We further analyze and model the repurchase state together with another two types of behaviors/purchase states: the variety-seeking purchase and the new purchase.

In addition to the unified models that we propose, one straightforward approach could be appending search results to recommendation results. It would be interesting to compare the performance of this simple approach with results from the unified models. The contribution of the unified models is mostly about products that both match the user's short-term search intention and the user's long-term purchase preference. Online A/B testing would be valuable to show the contribution of such unified systems in the real world.

The major contributions of this section include the following:

- Demonstrate the value of a *unified recommender system* to predict a user's purchase intention in e-commerce sites. We analyze the available information, including the query-related information, the product's marketing-related information and the user's history-related information.
- Compare several basic search algorithms and recommender systems in a e-commerce website.
- Propose to use the multinomial regression as the first basic unified model. It works significantly better than the basic search or recommender system. Yet it works as a black box without clear explanations of the user's purchase intention.
- Utilize the gradient boosting tree as the second unified model. It has better prediction performance than the first model and works as a black box.
- Design a three-stage graphical model as the third unified model to better understand the user's purchase intention. It jointly models the user's unobservable category choice, purchase state and product purchase. It has better explanatory power, in addition to the good prediction performance.

Chapter 4

Cross-Session: Utilizing Marginal Net Utility to Recommend Follow-up Purchase

In previous section, we discussed how to make recommendations by leveraging a user's heterogenous information within a session. While doing cross-session recommendations, traditional recommender systems predict a user's rating of an item based on the user's previous history and recommend products with highest ratings. Yet users do not make purchase decisions based on the rating. According to consumer behavior theory, a rational consumer chooses the product with the highest marginal net utility, i.e., a product's marginal utility minus the price. In the literature, the satisfaction or pleasure a user gets when purchasing/consuming a product is called the marginal utility. A product's marginal utility is dependent on the user's previous purchase history. A product with the higher marginal net utility is more likely to be purchased.

Some products have diminishing marginal utility. For example, the utility of purchasing a second computer is less than that of purchasing the first computer. For these products, users are not likely to purchase them again and again in a short time period. This is called the Law of Diminishing Marginal Utility in economics. On the other hand, some products (pet food, baby diapers, etc.) are likely to be purchased again and again. We call it the "re-purchase" behavior in this dissertation.

In order to match users' purchase decision(s), the recommender system should choose products with the highest marginal net utilities to recommend. Such a system is able to capture characteristics of both types of products and makes recommendations accordingly. Unfortunately, most of existing recommendation algorithms are not based on the marginal net utility optimization and cannot model the above two different products well. Most existing algorithms select products with the highest predicted ratings to recommend, assuming that the value/utility of a product for a user does not change over time.

This section introduces the concept of marginal net utility to develop recommendation algorithms. Inspired by the Cobb-Douglas function in consumer behavior theory, we propose a utility function for the recommender system in e-commerce sites. The new function contains a factor to control the product's marginal utility diminishing rate. Assuming that a user's purchasing decision depends on the marginal net utility, the rate can be learned from the purchase history of all users. The function can be applied on a family of existing recommender algorithms, which we call base algorithms. In this work, we choose SVD as our base algorithm. Applying our utility function to SVD leads to a new utility function SVD_{util} in this section.

We evaluate our algorithm on the purchase history from an e-commerce website shop.com. The experimental results show that our approach can improve the base algorithm significantly with better precision, recall and conversion rates. If the user purchased what he/she purchased before, the product is denoted as a **re-purchase product**. Otherwise, if the user purchased a product that he/she never purchased before, the product is denoted as a **new product**. The recommended list generated by our proposed algorithm contains both new products and re-purchase products. Or we can add a filter on top of the recommender system to recommend only re-purchase products or only new products for specific recommendation tasks.

The rest of this section is organized as follows: Section 4.1 introduces basic utility functions in economics and marketing research. Then we propose a new utility-based recommendation framework, motivated by these basic utility functions. Section 4.2 applies the new framework to revamp the well-known SVD algorithm. Section 4.3 first introduces the experimental design to evaluate and understand the new algorithm. Then it presents experimental results and further analysis. In the end, Section 4.4 concludes the contribution of this work.

4.1 Algorithm Design

In this section, we first define some basic notations to be used. Then we review two representative consumer utility functions. After proposing a new utility function for our problem, we describe how it can be used in a general framework to revamp some existing recommendation algorithms.

4.1.1 Notations

The following notations are used in the problem definition and analysis.

$u = 1, ..., M$: the index of a user. M is the number of unique users in the system.

i or $j = 1, ..., N$: the index of a product. N is the number of unique products in the system.

t: time.

c_i: the price of product i. We assume that this value is given. Making this value time dependent (i.e., replacing it with $c_{i,t}$) won't affect any analysis in the rest of this work.

$P_{M \times N}$: user-product matrix. Depending on the context, each entry $P_{u,i}$ could be user u's rating of product i, user u's purchase count of product i, or a unary/binary value indicating whether user u has purchased product i or not.

The goal of our system is to provide a ranked list of personalized recommendations to user u. In this work, each user's budget is not taken into consideration. The optimal purchase rule [3] indicates that a rational consumer purchases the product to maximize the marginal net utility. Accordingly the algorithm should choose the product with the maximum marginal net utility at each recommendation time point. Since the product price is given, the core problem of a recommender system is to determine the marginal utility for each product.

4.1.2 Marginal Utility

The utility of a product for a user depends on the user's purchase history. Marginal utility is used in economics and marketing research to represent the additional utility the consumer gets when consuming an additional unit of a product. The **Law Of Diminishing Marginal Utility** states that the marginal utility of a product drops while the consumption of the product increases. For example, for user u, the utility of consuming the first "iPhone 4" might be 10, the utility of consuming the second one might be 5, while the utility of consuming a third one might be only 1.

It is worth mentioning that the standard definition of the **Law Of Diminishing Marginal Utility** assumes continuity. It means that all units of a product are purchased one after another, without time gaps between any two purchases. However this assumption might not hold in an e-commerce site, where there are time gaps and purchases of other products between most re-purchases. For example, the user might purchase some pet food on Friday evening, then some books on the following Monday morning, and some pet food again after two weeks. Although the continuity assumption no longer holds in our e-commerce domain, we can still use the concept of marginal utility. Utility functions in economics can be adapted to our problem. This enables us to make recommendations based on the different diminishing return rate of different products.

Linear Utility Function

Linear utility function is one of the simplest functions in consumer behavior theory, as shown in Equation 4.1.

$$U(X) = \sum_{j} \alpha_j x_j \tag{4.1}$$

where X is the set of products the user consumed, and x_j is the consumption quantity of product j. α_j is the basic utility of product j, indicating the purchasing intention for product j. $U(X)$ is the utility of the entire purchase list X.

We can calculate the marginal utility $\Delta U(X, i)$ of purchasing one additional unit of product i in the following equation.

$$\Delta U(X, i) = U(X, i) - U(X) = \alpha_i$$

where $U(X, i) = \sum_{j:j \neq i} \alpha_j x_j + \alpha_i x_i'$ is the utility of the entire purchase list with one additional product i. $x_i' = x_i + 1$ is the updated quantity of product i.

The linear utility does not capture the diminishing return characteristic. From the deduction, we can see that the previous purchase count x_i of product i does not affect the marginal utility of purchasing one additional unit of product i. If product i's basic utility α_i is high, the system will recommend it regardless of previous purchase(s) of the same product.

In most existing recommender systems, products with the highest predicted values are recommended to the users. If these products include both re-purchase ones and new products, the approach follows the linear utility assumption. If only products that were never purchased are recommended, the following utility function is used:

$$U(X) = \sum_j \alpha_j \tag{4.2}$$

However, both of these two underlying utility functions do not match how users make purchase decisions in the real world.

Cobb-Douglas Utility Function

Another representative utility function is the Cobb-Douglas utility function [15]. This function is widely used due to its attractive mathematical characteristic: the ability of modeling the diminishing marginal return. The functional form is:

$$U(X) = \sum_j \alpha_j log(x_j) \tag{4.3}$$

where the definitions of x_j and α_j are the same as before. The marginal utility of purchasing one additional unit of product i is:

$$\begin{aligned} \Delta U(X,i) = U(X,i) - U(X) &= \alpha_i(log(x_i') - log(x_i)) \\ &= \alpha_i(log(x_i+1) - log(x_i)) \end{aligned} \tag{4.4}$$

The above equation shows that the marginal utility of product i decreases as the consumption quantity of product i increases. The diminishing return rate is $log(x_i+1)-log(x_i)$.

4.1.3 New Marginal Utility Function for E-Commerce Sites

Diminishing marginal utility is a widely recognized consumer behavior which we intend to model in the design of a recommender system for e-commerce sites. This motivates us to utilize the Cobb-Douglas utility function in our algorithm design.

However, Equation 4.4 shows two major drawbacks of the Cobb-Douglas utility function. First, the marginal utility of different products has the same diminishing return rate $log(x_i + 1) - log(x_i)$. It does not differentiate two types of products: products that the user would not purchase many times vs. products that the user would purchase again and again. Second, the basic utility α_i of a product i does not depend on the particular user, which contradicts with the goal of a personalized recommender system.

A real-world e-commerce system could collect a large amount of consumer purchase data. This enables us to handle these drawbacks of the Cobb-Douglas utility function. We can learn the product-specific diminishing return rate and the user-specific basic utility from the data. Now we describe how to modify the Cobb-Douglas utility function to achieve this goal.

We propose a new marginal utility function as follows:

$$\Delta U_{u,t}(X, i) = \alpha_{u,i}((x_{u,i,t} + 1)^{\gamma_i} - x_{u,i,t}^{\gamma_i}) \tag{4.5}$$

where $x_{u,i,t}$ is user u's consumption quantity of product j by time t.

There are four major differences between Equation 4.5 and Equation 4.4. First, we substitute $log(x_j)$ with x_j for the mathematical convenience. This simplification is motivated by the well-known **Constant elasticity of substitution (CES)** utility function in Equation 4.6 [79]:

$$U(X) = \sum_j \alpha_j x_j^{\gamma} \tag{4.6}$$

Second, x_i is substituted by $x_{u,i,t}$, so that it depends on the product, the user and the time. Third, α_i is substituted by $\alpha_{u,i}$, so that it is personalized to each individual user. Fourth, we introduce γ_i as a parameter to capture the diminishing return rate of product i.

One important question is how to determine $x_{u,i,t}$. The original definition of $x_{u,i,t}$ is user u's purchase count of product i by time t. We make two major modifications while calculating $x_{u,i,t}$ in the e-commerce domain. First, we define the purchase count as the number of purchase orders the current user made. Each order is counted once for the same product, regardless of the product quantity in the order. For example, if the user purchases 4 window panels in one order, the purchase count of window panel is 1. Second, we assume that the marginal utility is affected not only by previous purchase(s) of the same product, but also by previous purchase(s) of similar products. For example, the previous purchase of "iPhone 3" has effect on the marginal utility of the current purchase of "iPhone 4". We first find products that are similar to product i based on the metadata, such as the product title. Then let $x_{u,i,t}$ be the total similarity between these similar products and the current product i for user u at time t.

$$x_{u,i,t} = \sum_{j:sim(i,j)\geq\theta} C_{u,j,t} \times sim(i,j)$$

where $C_{u,j,t}$ is user u's purchase count of product j by time t, and $sim(i,j)$ is the similarity between product i and product j. θ is a similarity threshold, which will be estimated by the cross-validation in our experiments.

It is worth mentioning that the utility over a sequence of purchases can be calculated based on the definition of the above marginal utility (Equation 4.5). However, the utility of an unordered set of products is not defined. The only exception is when $\theta = 1$, in which case only the same product will influence the marginal utility. Under such circumstance, the utility over a sequence is independent of the sequential order of products. Then the utility can be used for an unordered set of products. With some derivation, we can get the definition of a user u's utility over an unordered set of products X at time t in this special case as follows:

$$U_{u,t}(X) = \sum_{j} \alpha_{u,j} x_{u,j,t}^{\gamma_j} \tag{4.7}$$

It is similar to the Constant Elasticity of Substitution utility function in Equation 4.6. However, the meaning of γ_i in Equation 4.7 is different from γ in Equation 4.6. γ is the parameter to tune the elasticity of substitution [1] and is the same for different products. In our new utility function (Equation 4.7), γ_i is a parameter to tune the diminishing return rate. It is product-specific and can be learned based on the purchase history of each product. Linear and Cobb-Douglas utilities can be viewed as special cases of our utility function. That is, in the limit as all γ_i approach 1, we get the linear utility; as all γ_i approach 0, we get the Cobb-Douglas utility.

4.1.4 Revamp Existing Algorithms with the New Marginal Utility Function

In most cases, we can view an existing recommendation algorithm as a function $f(u, i)$ to estimate the value of product i for user u without considering the diminishing return or the cost of a product. To reflect a user's true decision behavior in reality, we model $v_{u,i,t}$, the marginal net utility of product i for user u at time t, as follows:

$$v_{u,i,t} = f(u,i)[(x_{u,i,t} + 1)^{\gamma_i} - (x_{u,i,t})^{\gamma_i}] - c_i \tag{4.8}$$

Whether product i is likely to be purchased is dependent on the product's basic utility, its diminishing return rate, as well as the product's price. Comparing Equation 4.8 and Equation

[1]More information of CES utility function can be found in the reference [79].

4.5, we notice that $\alpha_{u,i}$ is replaced by $f(u,i)$. In other words, we propose to use an existing recommendation algorithm $f(u,i)$ to estimate $\alpha_{u,i}$, the basic utility of product i for user u.

At each decision point t, a higher marginal net utility $v_{u,i,t}$ indicates that user u is more likely to purchase product i. The following logistic function can be used to capture this intuition and model the conditional probability of making the purchase.

$$Pr(r_{u,i,t}|v_{u,i,t}) = \frac{1}{1 + e^{-v_{u,i,t} \cdot r_{u,i,t}}} \tag{4.9}$$

where $r_{u,i,t} = 1$ if user u purchases i at time t. Otherwise $r_{u,i,t} = -1$.

Assume that the cost c_i of product i is given, the parameters of the above model include γ_i and parameters of function f. To learn these parameters, we can order the entire user purchase history by the purchase time. At each time point t, user u's purchase decision of product i is considered as a training point. If user u purchased the product, it is a positive training point with $r_{u,i,t} = 1$. Otherwise, it is a negative training point with $r_{u,i,t} = -1$.

When a new recommended list is needed, the system can estimate each product's marginal net utility based on Equation 4.8 and rank them accordingly. For email-based or message-based marketing/recommendation applications, the system can predict how likely a user will purchase an item using Equation 4.9 and decide whether to recommend an item to the user accordingly (as in TREC adaptive filtering tasks).

4.2 Apply New Marginal Utility Function on SVD

In this section, we choose a popular recommendation algorithm SVD as an example. The algorithm in Section 4.1.4 is used to revamp it. This leads to a new recommendation algorithm, which we call SVD_{util}.

4.2.1 SVD

In this work, we choose SVD as an example because it is a popular recommendation algorithm with a decent performance. It is the basis of several recommendation algorithms based on latent factors, which have been proven to work well on benchmark recommendation datasets including the Netflix dataset [5].

Following [16, 37], we represent SVD as a matrix factorization approach. The basic SVD algorithm operates over a user-product matrix $P_{M\times N}$. It assumes that each entry $P_{u,i}$ in the matrix P can be estimated using the following form:

$$\hat{P}_{u,i} = q_i^T p_u \tag{4.10}$$

where q_i and p_u are vectors, which are the hidden representation of product i and user u. These vectors can be estimated based on all given entries in $P_{M\times N}$.

The value of $P_{u,i}$ in the observed matrix $P_{M\times N}$ is determined by the user purchase history. In the e-commerce domain, $P_{u,i}$ is usually set to be a unary value that indicates whether user u purchased product i or not [27]. Existing work finds that unary value is more suitable than numerical value, i.e., the number of times the user purchased the product [68]. Our piloting experiment compares these two types of values and reaches the same conclusion.

When a recommender system ranks all products by their estimated $\hat{P}_{u,i}$ values and selects the top ones to recommend, it is equivalent to maximizing the linear utility.

4.2.2 SVD$_{util}$: Revamp SVD

In this section, we apply the technique in Section 4.1.4 to revamp SVD. To do so, we set the basic utility $\alpha_{u,i} = f(u,i) = q_i^T p_u$. Based on Equation 4.8, the marginal net utility is:

$$v_{u,i,t} = q_i^T p_u[(x_{u,i,t}+1)^{\gamma_i} - (x_{u,i,t})^{\gamma_i}] - c_i \tag{4.11}$$

For simplicity and as commonly done in the literature, we assume that prior distributions of user vector p_u and item vector q_i are Gaussian distributions. $Pr(p_u)$ and $Pr(q_i)$ have mean zero and variance $1/\lambda_1$. We assume that the prior distribution of γ_i is a Gaussian distribution with mean γ_0 and variance $1/\lambda_2$. We treat each purchase order made by a user as a decision point. The purchase history of all users can be viewed as the training data $D = (r_{u,i,t}, c_i, u, t)$. The joint probability (likelihood) of all parameters and the training data is:

$$L = \prod_u Pr(p_u) \prod_i Pr(q_i) \prod_i Pr(\gamma_i) \prod_{u,i,t} Pr(r_{u,i,t}|v_{u,i,t}) \tag{4.12}$$

4.2.3 Parameter Estimation

The model parameters can be found by maximizing the joint probability of all parameters and the training data. According to Equation 4.12 and Equation 4.9, this is equivalent to minimizing the negative log likelihood of the data as follows:

$$\begin{aligned}
&(p_u, q_i, \gamma_i) = argmin\ [-logL] \\
=&argmin\ \frac{1}{2}\lambda_1 \sum_u ||p_u||^2 + \frac{1}{2}\lambda_1 \sum_i ||q_i||^2 + \frac{1}{2}\lambda_2 \sum_i (\gamma_i - \gamma_0)^2 \\
&+ \sum_{u,i,t} log(1 + e^{-v_{u,i,t} \cdot r_{u,i,t}})
\end{aligned}$$

λ_* can also be viewed as regularization factors to avoid the overfitting problem.

The first order derivatives are:

$$
\begin{aligned}
\frac{\partial(-logL)}{\partial q_i} &= \lambda_1 q_i + g_{u,i,t} \cdot [p_u \cdot d_{u,i,t}] \\
\frac{\partial(-logL)}{\partial p_u} &= \lambda_1 p_u + g_{u,i,t} \cdot [q_i \cdot d_{u,i,t}] \\
\frac{\partial(-logL)}{\partial \gamma_i} &= \lambda_2 \gamma_i + g_{u,i,t} \cdot [q_i^T p_u \cdot ((x_{u,i,t}+1)^{\gamma_i} \\
&\cdot log(x_{u,i,t}+1) - (x_{u,i,t})^{\gamma_i} \cdot log(x_{u,i,t}))]
\end{aligned}
$$

where

$$
\begin{aligned}
d_{u,i,t} &= [(x_{u,i,t}+1)^{\gamma_i} - (x_{u,i,t})^{\gamma_i}] \\
v_{u,i,t} &= q_i^T p_u d_{u,i,t} - c_i \\
g_{u,i,t} &= \frac{\partial(log(1+e^{-v_{u,i,t} \cdot r_{u,i,t}}))}{\partial v_{u,i,t}} \\
&= \frac{e^{-v_{u,i,t} \cdot r_{u,i,t}}}{1+e^{-v_{u,i,t} \cdot r_{u,i,t}}} \cdot (-r_{u,i,t})
\end{aligned}
$$

Based on the above derivation, we can use the stochastic gradient descent method to find the optimal parameters. Following the standard stochastic gradient descent method, update rules at each iteration are shown in the following equations. The algorithm stops when the change in an iteration is small enough.

$$
\begin{aligned}
p_u &= p_u - \beta_1 \cdot \frac{\partial(-logL)}{\partial p_u} \\
q_i &= q_i - \beta_1 \cdot \frac{\partial(-logL)}{\partial q_i} \\
\gamma_i &= \gamma_i - \beta_2 \cdot \frac{\partial(-logL)}{\partial \gamma_i}
\end{aligned}
$$

where β_* controls the learning rate at each iteration. β_* and λ_* can be set by the cross-validation.

4.3 Experimental Design and Result

We collect a dataset from a real-world e-commerce website, shop.com, for our experiments. This dataset contains the purchase history from 2004-01-01 to 2009-03-08. We use all purchases that have category information of the product. Tail users that made less than 5 product purchases are filtered out in the training data, which follows the similar pre-processing in related work [101]. In addition, 10 spam users that made more than 200 product purchases are filtered out as well. The remaining data contains 11,351 users and 67,291 products. There are 105,550 unique (user, product) pairs. As we can see, the user-product matrix is quite sparse, with only 0.014% density. All products are kept in the training process.

We sort all purchase history by time. The first 90% is used as the training data (data before 2008-11-24) and the last 10% is used as the testing data (data after 2008-11-24). There are 7,014 testing cases in total. At the product level, there are 1,143 repurchase cases(16.29%) and 6,269 new purchase cases(89.37%). At the categorical level, there are 2,850 repurchase cases(i.e. 40.63% cases with purchase from the same category) and 4,850 new purchase cases(i.e. 69.14% cases with purchase from a new category).

We have $\beta_1 = 0.015, \beta_2 = 0.035, \lambda_1 = 0.05$, $\lambda_2 = 0.01$ for positive training points, $\lambda_2 = 0.001$ for negative training points, $\gamma_0 = 1$, and $\theta = 0.7$. Instead of using all negative training points, we randomly sampled 1% from missing entries as negative training points in SVD_{util}. The sample percentage is determined by cross-validation. Both positive and negative training datas are used to learn model parameters. Finally the model is used to generate the recommendation list in the testing stage. For every decision point t in the testing data, all products are ranked by the marginal net utility $v_{u,i,t}$ (Equation 4.11). The top ranked products are recommended to the user.

We set the dimension of the user vector and the product vector to be 50. Product titles are used to calculate products' similarity. All stop words are removed. [2]

Some researchers have tried to fit SVD model parameters to the entire matrix P by replacing the missing entries with a base value such as 0. They found that such methods perform better for ranking metrics, such as recall and precision [16]. Their experiment results are performed on the movie data (Movielens and Netflix) with matrix density being 4.26% and 1.18% respectively. The matrix density on an e-commerce data set is usually much lower [27] (0.015% in our dataset). To train SVD, we randomly sample 0.1% missing entries and set them to 0, where percentage is determined by the cross-validation. The method is denoted as SVD_{matrix}.

4.3.1 Evaluation Metrics

There are several metrics to evaluate recommender algorithms in the literature [27]. Considering the usage scenario in a typical e-commerce web site, the ranking of all recommendations is more important than the rating prediction. Instead of using some common rating prediction accuracy measures (Root Mean Square Error, etc.), we evaluate all algorithms in the context of a ranking task [16].

- Each order $D_{u,t}$ corresponds to a testing point, which is uniquely identified by a (user, order time) pair. Let $S_{purchased}$ be the set that contains all products in this purchase order.
- Rank all products according to their predicted marginal net utility for user u at time t. Let $S_{K,recommended}$ be the set that contains the top K products.

[2]Other metadata information, such as user's review to the product, product's limited description, can be used in the future.

Then we have:

$$recall@K = \frac{|S_{purchased} \cap S_{K,recommended}|}{|S_{purchased}|} \tag{4.13}$$

$$precision@K = \frac{|S_{purchased} \cap S_{K,recommended}|}{K} \tag{4.14}$$

Conversion rate, a commonly-used metric in e-commerce, is used as an additional evaluation metric in our experiments. If the user purchased at least one product from the recommended top K list, we consider that the user has converted from a browser into a buyer. The calculation of conversion rate for one testing point is shown in the following equation.

$$conversion\ rate@K = \begin{cases} 1 & S_{purchased} \cap S_{K,recommended} \neq \varnothing \\ 0 & otherwise \end{cases} \tag{4.15}$$

Conversion rate reflects whether a user receives at least one good recommendation. The average value of all testing points will be used to compare among different algorithms. Statistical significant tests are used when comparing two methods.

4.3.2 General Analysis

In this section, we intend to answer the following questions with the general analysis:

- **How does SVD_{util} perform?** We compare it with SVD_{matrix} and a naive method *TopPop*. The *TopPop* method recommends the most popular products. It is a simple non-personalized recommendation algorithm that is commonly used in real world applications. Prior research has shown that it outperforms some common recommendation approaches and almost matches the accuracy of sophisticated algorithms in top-N recommendation task [16].

Table 4.1: Conversion rate performance for the general recommendation task. Value* is significantly better than the baseline SVD_{matrix} and TopPop.

Method	K=1	K=2	K=3	K=4	K=5
TopPop	0.0000	0.0003	0.0007	0.0153	0.0155
SVD_{matrix}	0.0134	0.0231	0.0284	0.0311	0.0344
SVD_{util}^{same}	0.0287*	0.0322*	0.0341*	0.0365*	0.0369
$SVD_{util}^{0.7}$	0.0255*	0.0314*	0.0345*	0.0354*	0.0366

- **Does considering similar products help the performance compared to using the exact same product?** We compare SVD_{util} with different similarity thresholds: $\theta = 0.7$ vs. $\theta = 1.0$. When $\theta = 1.0$, only the purchase(s) of the same product influences the diminishing return. When $\theta = 0.7$, the purchase(s) of similar products also influences the diminishing return. In the analysis, we represent the model with $\theta = 0.7$ as $SVD_{util}^{0.7}$, and the model with $\theta = 1.0$ as SVD_{util}^{same}.

The conversion rate performance of all methods is shown in Table 4.1. It is clear that all personalized methods are significantly better than the non-personalized method *TopPop*. Although the precision and recall are not reported here, we have similar observations when evaluating with these two metrics.

Comparing $SVD_{util}^{0.7}$ and SVD_{util}^{same} with SVD_{matrix}, we can see that our proposed marginal utility function helps. Both $SVD_{util}^{0.7}$ and SVD_{util}^{same} are significantly better than SVD_{matrix} in top 4 positions.

Between two methods with the new utility function, the performance of SVD_{util}^{same} is slightly better than $SVD_{util}^{0.7}$. The method SVD_{util}^{same} only utilizes the same product to learn the diminishing return rate and estimate the marginal utility. Thus it might catch re-purchase behavior with the higher accuracy. This will be further analyzed in the next section.

4.3.3 Further Analysis: Re-purchase Product Recommendation and New Product Recommendation

In the previous section, we generate a recommended list by ranking all products, including both re-purchase products and new products. Besides the general recommendation task, there are more specific tasks in e-commerce sites. Now we perform further analysis to compare these recommendation algorithms in two different recommendation tasks.

One task is to recommend products for a user to re-purchase. In the testing data, 16.29% of the purchase orders from returning users contain re-purchase products, i.e., products that were purchased by the user before. To successfully recommend products for the user to re-purchase could save a consumer much time and effort, and might be able to increase sales.[3] The challenge is how to rank products the user has purchased before. For all re-purchase products in the testing data, we plot the the histogram of their previous purchase count (Figure 4.1). We observe that the majority of re-purchase products were purchased only once or twice in the training data. Thus it is hard to use the previous purchase count to rank. In Section 4.3.3, we compare all methods' performance in ranking products that were purchased before. Only re-purchase products in each testing order are used to evaluate all methods' performance.

A second task is to recommend new products that a user has never purchased before. In the testing data, 89.37% in the dataset of all purchase orders from returning users contain new products. The new product recommendation task is much harder and more interesting than the re-purchase recommendation task. The recommendation candidates include all products that were not purchased by the current user before. Only new products in each testing order are used to evaluate all methods' performance.

In this section, we intend to answer the following questions:

[3]This is why amazon.com is providing subscribing service with discount to encourage re-purchase.

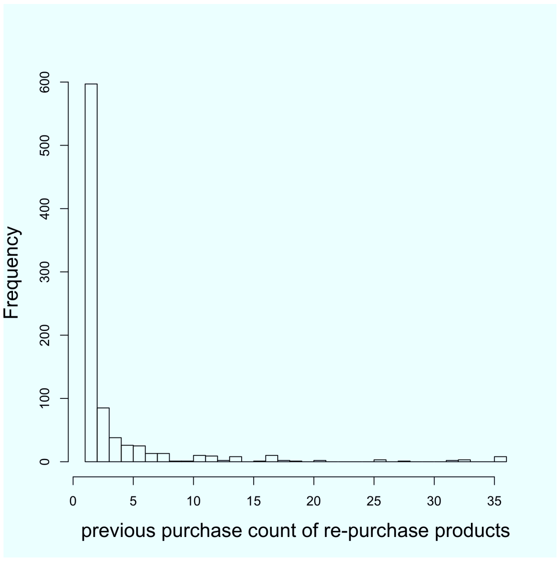

Figure 4.1: Histogram of the previous purchase count in the training data of all re-purchase products. If a product in the testing data was purchased before, it is a re-purchase product.

Table 4.2: Conversion rate@K for the re-purchase recommendation task. Value* is significantly better than the baseline SVD_{matrix} and TopPop.

Method	K=1	K=2	K=3	K=4	K=5
TopPop	0.0	0.0009	0.0035	0.0927	0.0936
SVD_{matrix}	0.0822	0.1409	0.1724	0.1881	0.2073
SVD_{util}^{same}	0.1750*	0.1969*	0.2082*	0.2222*	0.2248*
$SVD_{util}^{0.7}$	0.1540*	0.1872*	0.2056*	0.2108*	0.2170*

- **How does SVD_{util} perform in the re-purchase product recommendation task?**
- **How does SVD_{util} perform in the new product recommendation task?**

Re-purchase Product Recommendation Task

The conversion rate for the re-purchase product recommendation task is shown in Table 4.2.

Table 4.3: Product 2426's purchase history in the training data. The table is used in Section 4.3.3.

userID	order time
u_{1916}	12/14/06 6:44
u_{2694}	12/16/07 19:34
u_{2613}	6/10/08 16:11
u_{2857}	8/6/08 20:11
u_{2863}	8/23/08 8:58

$SVD^{0.7}_{util}$ and SVD^{same}_{util} perform significantly better than baseline methods. This is as expected, since the new algorithm is expected to capture the re-purchase behavior. SVD^{same}_{util} is better than $SVD^{0.7}_{util}$, probably because using similar products introduces some noise into the model. However, it is worth mentioning that $SVD^{0.7}_{util}$ is able to catch some re-purchase behavior ignored by SVD^{same}_{util}. Some product was not purchased again and again in the training data, yet it is similar to the user's previous purchase. If such product is purchased in the testing data, the re-purchase behavior can only be captured by $SVD^{0.7}_{util}$ yet not SVD^{same}_{util}. For methods with filter, $SVD^{0.7}_{util}.Previous$ performs the best.

Here is one example. Table 4.3 shows product 2426's entire order history in the training data. Table 4.4 shows the product title of product 2426 and some of its similar products. 2426 was purchased only once by each of the five users. In this case, SVD^{same}_{util} cannot learn that it is a potential re-purchase product. Yet user u_{2613} purchased two products (2365 and 1329) before, which are similar to product 2426. With this information, $SVD^{0.7}_{util}$ learns that product 2426 is likely to be purchased again after a user purchases a similar product or itself. In the testing data, $SVD^{0.7}_{util}$ recommends product 2426 to user u_{2857} at timestamp 12/1/08,17:52. It was purchased by user u_{2857} at that time, which is a re-purchase behavior.

Table 4.4: Product title of Product 2426 and two similar products. The table is used in Section 4.3.3.

ProductID	Product Title
2426	Obsession by Calvin Klein TESTER for Women Eau de Parfum Spray 3.4 oz
2365	Eternity by Calvin Klein for Women Eau de Parfum Spray 1.7 oz
1329	Eternity by Calvin Klein for Women Eau de Parfum Spray 3.4 oz

Table 4.5: Conversion rate@K for the new product recommendation task. Value* is significantly better than the baseline SVD_{matrix} and TopPop.

Method	K=1	K=2	K=3	K=4	K=5
TopPop	0.0002	0.0002	0.0002	0.0002	0.0003
SVD_{matrix}	0.0000	0.0002	0.0003	0.0008	0.0011
SVD_{util}^{same}	0.0002	0.0002	0.0002	0.0003	0.0003
$SVD_{util}^{0.7}$	0.0005	0.0011	0.0016	0.0022	0.0026

New Product Recommendation Task

Table 4.5 compares the conversion rate of all methods in recommending new products. $SVD_{util}^{0.7}$ achieves better performance than the baseline SVD_{matrix} and TopPop. On the other hand, SVD_{util}^{same} does not help recommending new products. In this task, $SVD_{util}^{0.7}$ performs better than SVD_{util}^{same} since it learns from similar products' purchase behavior with $\theta < 1$. Recommending good and new products makes $SVD_{util}^{0.7}$ more attractive in e-commerce sites since it adds more serendipity to the user.

To better understand how the proposed method works, Table 4.6 shows user u_{3007}'s order and Table 4.7 shows the corresponding recommended lists generated by different methods.

User u_{3007} purchased product 915 in the training data. In the SVD_{util}^{same} method, only the marginal (net) utility of the exact same product 915 will be changed. Since product 915 was purchased by the other user(s) for many times, it is recommended to user u_{3007} by

Table 4.6: user u_{3007}'s purchase order

times	productID	product title
In the training data		
9/28/08 0:20	915	Frontline Top Spot DOG up to 22lb (3 pack)
	1049	Pet Bio Guard Shampoo
	1033	Frontline Top Spot DOG 89-132lb. (3 pack)
In the testing data		
10/23/08 8:37	914	Frontline Top Spot DOG 45-88lb. (3 pack)
	293	Frontline PLUS - Dog (23 to 44 Lbs) 3-Pack

Table 4.7: Top 5 recommendation for user u_{3007} at time 10/23/08 8:37

Method	ProductID	Product title
TopPop	165	Frontline PLUS - Dog (45-88 Lbs)
	166	Frontline PLUS - Cat 3-Pack
	322	Pet Solid Compressed Rawhide Bone - 5"
	1611	Frontline PLUS - Dog up to 88 lbs. (6-Pack)
	1924	9" x 12" Colorations Heavyweight Construction Paper
SVD_{matrix}	165	Frontline PLUS - Dog (45-88 Lbs)
	1611	Frontline PLUS - Dog up to 88 lbs. (6-Pack)
	166	Frontline PLUS - Cat 3-Pack
	1593	Frontline PLUS - For Cats (6-Pack)
	2028	Colorations Simply Washable Tempera Paint
SVD_{util}^{same}	915	Frontline Top Spot DOG up to 22lb (3 pack)
	1033	Frontline Top Spot DOG 89-132lb. (3 pack)
	1049	Pet Bio Guard Shampoo
	2978	Nasal Aspirator
	2678	Brother LC51BK Compatible Black Ink Cartridge
$SVD_{util}^{0.7}$	914	**Frontline Top Spot DOG 45-88lb. (3 pack)**
	1033	Frontline Top Spot DOG 89-132lb. (3 pack)
	1610	Frontline Top Spot DOG up to 22lb (6 pack)
	915	Frontline Top Spot DOG up to 22lb (3 pack)
	30	Frontline Top Spot CAT (3 pack)

SVD_{util}^{same}. In the $SVD_{util}^{0.7}$ method, the marginal (net) utility of similar products will also be changed, including product 914. Its estimation shows that product 914 has a higher marginal net utility. Thus it appears in the top position of $SVD_{util}^{0.7}$ recommended list, which was actually purchased by the user in the testing data. With these examples in the data, we discover that it is worthwhile to model the purchase decision based on the product's marginal net utility. The utility function can capture much information from consumer behavior.

4.4 Summary and Contribution

Inspired by the utility function in consumer behavior theory, we design a general framework to revamp existing recommendation algorithms for e-commerce sites. We apply it to SVD, which leads to a new algorithm SVD_{util}. We evaluate two special cases of the new algorithm: $SVD^{0.7}_{util}$ and SVD^{same}_{util}. The former one utilizes similar products in estimating the marginal utility while the latter one utilizes only the same product. These two methods use each purchase decision point in the data sequentially for the parameter estimation. On shop.com data, the new methods perform significantly better than baselines because they can better capture the re-purchase behavior as well as the diminishing return of the marginal utility. When comparing between these two cases, we found that SVD^{same}_{util} performs better in the re-purchase product recommendation task. $SVD^{0.7}_{util}$ is more useful in recommending new products, which is a harder task and generates more interesting result to the user.

To sum up, major contributions of this work include the following:

- Introduce the idea of utilizing the marginal net utility in the recommender system of e-commerce sites. Adapt the utility function from the Cobb-Douglas function to apply in a general framework.
- Apply the new utility function on SVD algorithm, denoted as SVD_{util}. The new algorithm achieves significant improvement in conversion rates.
- The new algorithm $SVD^{0.7}_{util}$ performs significantly better in the re-purchase product recommendation task and the new product recommendation task.

There are different types of products in general. We list some examples here and demonstrate how the session-aware recommender system deals with different cases:

Consumable product: Such products are easily to be consumed, including pet food, diapers, office supplies, and so on. Due to the product's consumable characteristics, customers tend to purchase them again and again regularly. In the framework of a session-aware recommender system, the marginal utility of such a product increases over time. As a result, the consumable products in the user's purchase history tend to be recommended to the user often. It saves the user's time and effort to search for these products. A missing piece of information is the time interval between a repurchase behavior. For example, it would be disappointing if the system recommends the baby diaper to a mom each time she enters the site, no matter daily or weekly. One approach is to let the user subscribe to the repurchase products by themselves and tell the system how often they make a repurchase. Another approach is to let the system learn each user's purchase time of a consumable product and recommend at the right time. It would be discussed in the next section.

Durable product: Different from consumable products, durable products are not easily wear out, including computers, furnitures, cars, and so on. Due to the product's durable characteristics, customers do not tend to purchase them again and again in a short time. In the framework of a session-aware recommender system, the marginal utility of such a product decreases over time. As a result, the durable products in the user's purchase history will not be recommended to the user after his first purchase. In reality, it is good for a short time period, such as one year or even two years. Yet the user might want to upgrade his durable product, such as a laptop, after a certain time period. Thus the marginal utility for a durable product could drop first and increase gradually over time. More appropriate utility functions could be explored in the future. In addition, the system could learn such behavior given all users' purchase history over a long time. It would be discussed in the next section.

Substitute product: In this case, a product's demand would increase when the price of another similar product increased. For example, coca cola and pepsi cola are substitutes. In the framework of a session-aware recommender system, substitutes would influence each other's marginal utility due to their similarity. This is reasonable in the real application. For example, if the user just purchased "iphone", the recommendation of "blackberry" would be redundant.

Complementary product: In this case, a product's demand would decrease when the price of another similar product decreased. For example, camera and camera lens are complementary products. Such follow-up purchases between complementary products are not captured by the marginal utility framework. It would be discussed in the next section.

Chapter 5

Cross-Session: Opportunity Model to Recommend Right Product at Right Time

As we discussed before, traditional recommender systems focus on finding the right item to recommend. Major approaches include content-based methods and collaborative filtering methods. For example, if a user viewed some cameras in the website, the system learns that the user is interested in cameras and recommends more similar items to the user. In the previous section, we proposed that recommender systems should not only recommend items that the user is interested in. The goal is to recommend items that maximize the users' marginal utility. For example, the marginal utility of a camera decreases immediately after a user purchased a camera. Thus the system should recommend camera accessories instead of similar cameras to the user.

In the real world, the user satisfaction/utility is dependent on both the relevance and the time of the recommendation. While an irrelevant recommendation results in a negative utility, a relevant item could also lead to a negative utility due to the wrong time. It could waste a user's time and effort to receive product recommendation emails that are full of products she does not need to purchase at the time. In the long term, the user may have negative feedback about the company, unsubscribe from the marketing email list, or even label the emails as spam emails. To address these issues, recommender systems need to answer the following question: when is the *right time* for the system to make recommendations of the *right product(s)*? For example, after a user purchased a camera, when should the system recommend related accessories including camera lenses, batteries, digital photo frames, and so on?

Multiple heuristic approaches [83, 101] have been proposed to tackle this problem. In this work, we propose a theoretical model to learn the probability of a user making a follow-up purchase at a particular time. The model is inspired by the hazards model in survival analysis in statistics. The purchase time would be influenced by multiple factors, such as the user's purchase history, the product promotion information, the global environment and so on. Thus we propose to leverage the proportional hazards model which incorporates related factors as covariates(i.e., features). We further extend the model with the hierarchical Bayesian framework to handle the data sparsity issue. The new model is denoted as *Opportunity Model* in this section. It predicts the joint purchase probability, i.e., the probability of a user purchasing a product at a particular time. It helps to promote a right item at the right time which further enhances the user satisfaction. Experimental results are performed with a dataset from a real-world e-commerce website. Detailed analysis shows that the opportunity model could help to improve the conversion rate and the user satisfaction significantly.

5.1 Opportunity Model in E-commerce

To recommend the right product at the right time, we exam each candidate product for each user at a particular decision time. We propose to build an opportunity model to estimate the probability of a user purchasing the product at a particular time interval (i.e. $(y, y + \Delta t]$). That is the joint probability $P(p\_product, T \in (y, y + \Delta t])$, where T is the purchasing time. Let $P(p\_product)$ represent the probability of the user purchasing the product, and $P(T \in (y, y + \Delta t]|p\_product)$ represent the conditional probability of the user purchasing the product at a particular time period conditioned on that the user will purchase the product. Based on the chain rule, we have the joint probability:

$$\begin{aligned}&P(p\_product, T \in (y, y + \Delta t]) \\ &= P(T \in (y, y + \Delta t]|p\_product)P(p\_product) \end{aligned} \tag{5.1}$$

We propose to adapt hazards models in survival analysis to estimate $P(T \in (y, y+\Delta t]|p\_product)$, and adapt existing recommendation algorithms to estimate $P(p\_product)$.

5.1.1 Hazards Model in Survival Analysis

Survival analysis is a heavily studied topic in statistics, which is named as duration modeling in economics or reliability analysis in engineering. One major part of survival analysis is to estimate the time of an event, such as when a machine fails to work or when a patient fails to survive. In the e-commerce domain, we can view the task of conditional opportunity model as predicting the time of the follow-up purchase event of the product. Follow-up purchases may include repurchases that happen regularly or new purchases that are triggered by the previous purchase. Given the similar nature of survival analysis and our e-task, we propose to use the hazards model in survival analysis to estimate $p(y|p\_product)$ in this paper.

Let us review basic hazards models in survival analysis. Let $p(y)$ denote the density function of the time distribution of an event. Let y be a value of time and T be a random variable representing the event time. The cumulative distribution function is denoted as $P(y) = Pr(T \leq y)$ and survival function is denoted as $S(y) = Pr(T > y) = 1 - P(y)$. The hazards function is $h(y) = \frac{p(y)}{S(y)}$. It indicates the instantaneous potential per unit time for the event to occur at time y given that the event has not occurred up to time y.

The survival analysis further extends the model with covariates. Covariates are features that would affect the survival time. For example, if a product is on promotion, it might shorten the time before a user waits to purchase it. The covariates could include different types of features, including time independent variables (user age, gender, household income, product brand etc.), time dependent internal variables (time since last purchase of the same product, recent user search queries or clicks etc.) and time dependent external variables (global economy, seasonal index, day of the week, etc.). There are two common approaches to incorporate covariates $\mathbf{x}$(a vector of features) in a hazards model. The first approach is the *Cox proportional hazards model* [58]. It assumes that covariates are multiplicatively related to the hazards. The second approach is the *Accelerated life model* [89]. It assumes that covariates are multiplicatively related to the survival time. Research in statistics discovered that the Weibull distribution satisfies assumptions in both directions. The density function of the basic Weibull distribution is shown in Equation 5.2 [87].

$$p(y) = \gamma\theta y^{\gamma-1} exp\{-\theta y^{\gamma}\} \tag{5.2}$$

where γ is sometimes called the shape parameter and held fixed. If $\gamma = 1$, the Weibull model reduces to the exponential model and the hazard is constant. If $\gamma > 1$, the hazard increases as time increases. If $\gamma < 1$, the hazard decreases over time. θ is the scale parameter and can be

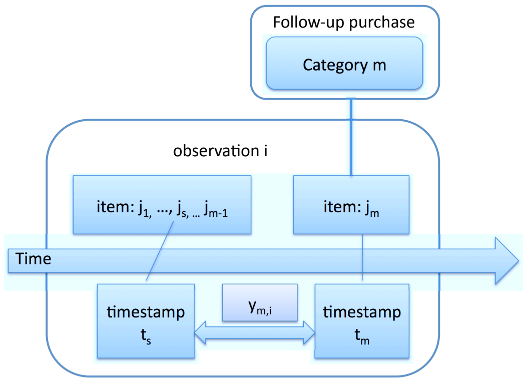

Figure 5.1: Illustration of the relationship of variables. The user first makes a series of product purchases before timestamp t_m. Then the user makes a purchase of item j_m in category m at timestamp t_m. This follow-up purchase in category m is the i^{th} observation in category m. Suppose that the purchase of j_m is triggered by item j_s at timestamp t_s. Purchase time $y_{m,i} = t_m - t_s$ is the time gap between t_m and t_s. An observation i is associated with two major variables: 1) purchase time $y_{m,i}$ and 2) covariates $\mathbf{x_{m,i}}$ (which is not shown in the figure).

re-parameterized based on a regression parameter β and covariates $\mathbf{x}$ as follows:

$$p(y) = \gamma exp\{\beta^T \mathbf{x}\} y^{\gamma-1} exp\{-exp\{\beta^T \mathbf{x}\} y^{\gamma}\} \tag{5.3}$$

where $exp\{\beta^T \mathbf{x}\}$ represents the new scale parameter θ'. This density function represents the basic proportional hazards model that models the event time with associated covariates. The corresponding hazards function is simply $h(y) = \gamma\theta' y^{\gamma-1}$.

5.1.2 Notations

Here we describe notations in the e-commerce domain in this paper. The relationship between different variables is shown in Figure 5.1.

- $u = 1, 2, ..., U$: the index of users.

- $m = 1, 2, ..., M$: the index of item categories. In the e-commerce domain, it is the category of the product, such as *Apparel & Accessories—Swimwear*, *Baby—Car Seats*, *Sports and Fitness—Football*, etc.
- $j_m = 1_m, 2_m, ..., J_m$: the index of items in category m. In this paper, an item is a product.
- t_m: the purchase timestamp of item j_m.
- Δt: the window size of the purchase time in consideration (such as 3 days or 1 hour)
- D: The observed data of all follow-up purchases from all users. Each category m has N_m follow-up purchase observation from all users. Each follow-up purchase observation $i = 1, ..., N_m$ in category m is associated with the purchase time $y_{m,i}$ and covariates $\mathbf{x_{m,i}}$.
- $y_{m,i}$: the purchase time of the i^{th} observation in category m. $y_{m,i} = t_m - t_s$ is the time distance between the user's purchase timestamp t_m of item j_m and the user's purchase timestamp t_s of the triggering item j_s.
- $\mathbf{x_{m,i}}$: the k-dimensional vector of covariates associated with the i^{th} observation in category m. Covariates could be associated with user u who makes the purchase, the user's purchase history $j_1, ..., j_{m-1}$, the purchase item j_m, the global environment, etc.
- $P(p\_product = yes)$: the probability of a user purchasing the product. The probability is calculated for each user-product pair.
- *Opportunity model*: density function/model to predict the joint purchase probability $p(y, p\_product)$, i.e., the probability of a user purchasing of the product at the particular time.

- *Conditional opportunity model*: density function/model to predict the conditional time probability $p(y|p\_product)$, i.e., the probability of a user purchasing the product at the particular time, given that the user will purchase the product.

5.1.3 Conditional Opportunity Model

In this paper we propose to adapt the Cox proportional hazards model to the e-commerce domain and use it to estimate $P(T \in (y, y + \Delta t]|p\_product)$. To do that, we learn the *conditional opportunity model* $p(y|p\_product)$, which is the density function/model of the purchasing time. We can either learn one model per product or one model per product category. Without loss of generality, we describe the model by assuming one model per category in this paper.

In the real world, a small number of categories are often purchased while most categories have few purchases. To solve the data sparsity issue, we follow a common practice and extend the conditional opportunity model with a hierarchical Bayesian framework as illustrated in Figure 5.2. This framework helps the category with few observations by borrowing information from other categories through a common prior for parameters of all proportional hazards models.

For each category m, β_m is sampled from a Gaussian distribution: $\beta_m \sim N(\mu_\beta, \mathbf{\Sigma}_\beta)$ and γ_m is sampled from the Gamma distribution: $\gamma_m \sim Gamma(a_\gamma, b_\gamma)$. We denote $\phi = (\mu_\beta, \mathbf{\Sigma}_\beta, \mathbf{a}_\gamma, \mathbf{b}_\gamma)$. For each i^{th} observation in category m with its observed covariates $\mathbf{x_{m,i}}$, its purchase time $y_{m,i}$ is sampled from the conditional opportunity model

$$\begin{aligned} & p(y_{m,i}|\beta_m, \gamma_m) \qquad (5.4) \\ & = \gamma_m exp\{\beta_m^T \mathbf{x_{m,i}}\} y_{m,i}^{\gamma_m - 1} exp\{-exp\{\beta_m^T \mathbf{x_{m,i}}\} y_{m,i}^{\gamma_m}\} \end{aligned}$$

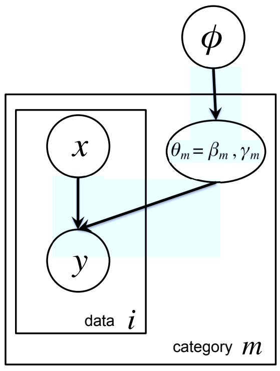

Figure 5.2: Illustration of dependencies of variables in the hierarchical conditional opportunity model. It shows the i^{th} observation of category m. $y_{m,i}$ is the purchase time which is dependent on the conditional opportunity model $\theta_m = \{\beta_m, \gamma_m\}$ of category m, as wells the observed covariates $\mathbf{x_{m,i}}$ of this purchase. Each category m has its own parameters of the conditional opportunity model β_m, γ_m. Models of each category share information through the prior, $\phi = (\mu_\beta, \mathbf{\Sigma}_\beta, \mathbf{a}_\gamma, \mathbf{b}_\gamma)$.

Consider that data D consists of a series of observations from all categories. Purchase times in the observations are generated by using a set of hidden variables $\theta = \{\theta_1, \theta_2 ..., \theta_M\}$ ($\theta_m = \{\beta_m, \gamma_m\}$). The likelihood can be written as a function of $\phi = (\mu_\beta, \mathbf{\Sigma}_\beta, \mathbf{a}_\gamma, \mathbf{b}_\gamma)$. We use y_m to represent $\{y_{m,1},$

$..., y_{m,i}, ..., y_{m,N_m}\}$, i.e., observation times in category m.

$$p(D|\phi) = \prod_{m=1}^{M} p(y_m|\phi) = \prod_{m=1}^{M} \int p(\theta_m, y_m|\phi) d\theta_m \tag{5.5}$$

5.1.4 Parameter Inference with Variational Bayesian

There is no closed-form solution for the estimation of the model parameters. We follow the variational Bayesian method[4] for constrained (approximate) optimization to derive an iterative process to find the approximate solution. Maximizing the likelihood in Equation 5.5 is equivalent to maximizing the log likelihood $L(\phi)$.

$$L(\phi) = \ln p(D|\phi) = \sum_{m=1}^{M} \ln p(y_m|\phi) = \sum_{m=1}^{M} \ln \int p(\theta_m, y_m|\phi) \mathrm{d}\theta_m$$

We can simplify the problem by introducing an auxiliary distribution $q(\theta_m)$ for each hidden variable θ_m [4]. In the variational approach, we constrain $q(\theta_m)$ to be a particular tractable form for computational efficiency. In particular, we assume that $q(\beta_m) = N(\mu_{\beta_m}, \Sigma_{\beta_m})$ and $q(\gamma_m) = Gamma(a_{\gamma_m}, b_{\gamma_m})$. The process to infer parameters is to iterate between the following E-step and M-step until convergence.

E-Step

In the **E-step**, we infer the posterior distributions over hidden variables θ_m given the current parameter setting ϕ. There is no closed-form solution. Instead, we find a tractable approximation of the posterior distribution of θ_m given ϕ (i.e, $q(\theta_m)$ that maximizes $L(\phi)$).

$$\begin{aligned}
L(\phi) &= \sum_{m=1}^{M} \ln \int q(\theta_m) \frac{p(\theta_m, y_m|\phi)}{q(\theta_m)} \mathrm{d}\theta_m \\
&\geq \sum_{m=1}^{M} \int q(\theta_m) \ln \frac{p(\theta_m, y_m|\phi)}{q(\theta_m)} \mathrm{d}\theta_m \\
&= \sum_{m=1}^{M} \int q(\theta_m) \ln p(y_m|\phi) \mathrm{d}\theta_m - \sum_{m=1}^{M} \int q(\theta_m) \ln \frac{q(\theta_m)}{p(\theta_m|y_m, \phi)} \mathrm{d}\theta_m \\
&\equiv \boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi)
\end{aligned} \tag{5.6}$$

To maximize $L(\phi)$, it is the same as minimize the following equation to find each distribution $q(\theta_m)$:

$$q(\theta_m)^{(t)} = \arg\min_{q(\theta_m)} \int q(\theta_m) \ln \frac{q(\theta_m)}{p(\theta_m|y_m, \phi)} \mathrm{d}\theta_m \tag{5.7}$$

Given that $p(\theta_m|y_m, \phi) = \frac{p(y_m|\theta_m)p(\theta_m|\phi)}{p(y_m|\phi)}$, we have:

$$q(\theta_m) = \arg\min_{q(\theta_m)} KL[q(\theta_m)||p(\theta_m|\phi)] - \int q(\theta_m) \ln p(y_m|\theta_m) \mathrm{d}\theta_m \tag{5.8}$$

The first part in Equation A.3 is the KL-divergence between the posterior distribution $q(\theta_m)$ and the prior distribution $p(\theta_m|\phi)$.

$$KL[q(\theta_m)||p(\theta_m|\phi)] = KL[q(\beta_m)||p(\beta_m|\phi)] + KL[q(\gamma_m)||p(\gamma_m|\phi)]$$

The KL-divergence between two Gaussian distributions is

$$\begin{aligned} & KL[q(\beta_m)||p(\beta_m|\phi)] \\ & = \frac{1}{2}[tr(\Sigma_\beta^{-1}\Sigma_{\beta_m}) + (\mu_\beta - \mu_{\beta_m})^T \Sigma_\beta^{-1}(\mu_\beta - \mu_{\beta_m}) \\ & - \ln(\frac{\det \Sigma_{\beta_m}}{\det \Sigma_\beta}) - k] \end{aligned}$$

The KL-divergence between two gamma distributions is

$$\begin{aligned} & KL[q(\gamma_m)||p(\gamma_m|\phi)] \\ & = (a_{\gamma_m} - a_\gamma)\Psi(a_{\gamma_m}) - \log\Gamma(a_{\gamma_m}) + \log\Gamma(a_\gamma) \\ & + a_\gamma(\log b_{\gamma_m} - \log b_\gamma) + a_{\gamma_m}\frac{b_\gamma - b_{\gamma_m}}{b_{\gamma_m}} \end{aligned}$$

where $\Psi(*)$ is the digamma function.

The second part in Equation A.3 is to maximize the data likelihood of $y_m = \{y_{m,1}, ..., y_{m,i}, ..., y_{m,N_m}\}$ with the current $\theta_m = \{\beta_m, \gamma_m\}$.

$$\begin{aligned}
&\sum_{i=1}^{N_m} \int q(\theta_m) \ln p(y_{m,i}|\theta_m) \mathrm{d}\theta_m \\
=&\sum_{i=1}^{N_m} \int \int q(\beta_m) q(\gamma_m) \ln p(y_{m,i}|\beta_m, \gamma_m) \mathrm{d}\beta_m \mathrm{d}\gamma_m \\
=&\sum_{i=1}^{N_m} [E(\ln \gamma_m) + E(\beta_m)^T \mathbf{x_{m,i}} + (E(\gamma_m) - 1) \ln y_{m,i} \\
&- E(exp(\beta_m^T \mathbf{x_{m,i}})) E(y_{m,i}^{\gamma_m})]
\end{aligned} \tag{5.9}$$

The expectations in the above equation are

$$\begin{aligned}
E(\ln \gamma_m) =& \Psi(a_{\gamma_m}) - \ln(b_{\gamma_m}) \\
E(\beta_m) =& \mu_{\beta_m} \\
E(\gamma_m) =& \frac{a_{\gamma_m}}{b_{\gamma_m}} \\
E(exp(\beta_m^T \mathbf{x_{m,i}})) =& exp\{\mu_{\beta_m}^T \mathbf{x_{m,i}} + \frac{1}{2} \mathbf{x_{m,i}}^T \Sigma_{\beta_m} \mathbf{x_{m,i}}\} \\
E(y_{m,i}^{\gamma_m}) =& \frac{b_{\gamma_m}^{a_{\gamma_m}}}{(b_{\gamma_m} - \ln y_{m,i})^{a_{\gamma_m}}}
\end{aligned}$$

We can combine the above derivations with Equation A.3, then find $q(\theta_m)$ using the conjugate gradient descent method.

M-Step

In the **M-step**, the goal is to maximize $\boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi)$ in Equation A.1 with respect to ϕ given all θ_m. It is same as to maximize the following quantity:

$$\phi^{(t+1)} \leftarrow \arg\max_{\phi} \sum_{m=1}^{M} \int q(\theta_m) \ln p(y_m|\phi) \mathrm{d}\theta_m \tag{5.10}$$

The optimal ϕ at this step can be estimated with the following closed form:

$$\begin{aligned}
\mu_\beta &= \frac{\sum_m^M \mu_{\beta_m}}{M} \\
\Sigma_\beta &= \frac{\sum_m^M [\Sigma_{\beta_m} + (\mu_{\beta_m} - \mu_\beta)(\mu_{\beta_m} - \mu_\beta)^T]}{M} \\
a_\gamma &= \Psi^{-1}(\ln b_\gamma + \frac{\sum_m^M \Psi(a_{\gamma_m}) - \ln(b_{\gamma_m})}{M}) \\
b_\gamma &= \frac{M a_\gamma}{\sum_m^M \frac{a_{\gamma m}}{b_{\gamma m}}}
\end{aligned}$$

where $\Psi^{-1}(*)$ is the inverse digamma function.

5.1.5 Joint Purchase Probability

At a time y, we can decide whether to recommendation a particular product in category m to a particular user or not based on the following estimation: whether a user is likely to purchase the product in the near future (i.e. between time y and time $y + \Delta t$), given that the user has not purchased the product since a triggering time point. To do so, we need to estimate $P(p\_product = yes, T \in (y, y+\Delta t]) = P(p\_product = yes)P(y < T \leq y+\Delta t | T > y, p\_product = yes)$, where

$$\begin{aligned}
&P(y < T \leq y + \Delta t | T > y, p\_product = yes) \\
&= \frac{P(T \leq y + \Delta t) - P(T \leq y)}{1 - P(T \leq y)}
\end{aligned} \tag{5.11}$$

where:

$$\begin{aligned}
&P(T \leq y) \qquad (5.12)\\
&= 1 - E_{\beta_m,\gamma_m}[exp\{-y^{\gamma_m} exp\{\beta_m^T \mathbf{x_i}\}\}]\\
&\approx 1 - exp\{E_{\beta_m,\gamma_m}[-y^{\gamma_m} exp\{\beta_m^T \mathbf{x_i}\}]\}\\
&= 1 - exp\{E_{\gamma_m}[-y^{\gamma_m}]E_{\beta_m}[exp\{\beta_m^T \mathbf{x_i}\}]\}\\
&= 1 - exp\{-\frac{b_{\gamma_m}^{a_{\gamma_m}}}{(b_{\gamma_m} - \ln y)^{a_{\gamma_m}}} exp\{\mu_{\beta_m}^T x_i + \frac{1}{2} x_i^T \Sigma_{\beta_m} \mathbf{x_i}\}\}
\end{aligned}$$

The approximation is used because there is no closed-form solution of the integration for calculating the expectation.

To estimate $P(p\_product = yes)$, i.e., the probability of the user purchasing the product, we can use any existing recommender systems through the logistic regression model:

$$P(p\_product = yes) = \frac{1}{1 + e^{-\mathbf{f}^T \mathbf{x}}}$$

where $\mathbf{x}$ is a vector of features that are associated with the purchasing, which includes the score/output of the existing recommender system(such as SVD), as well as other features that might help to predict the user's purchase probability. $\mathbf{f}$ is a vector of coefficients that can be learnt by maximizing the likelihood of the training data.

5.1.6 Implementation Details

The purchase time y_m is determined by the purchase timestamp of product j_m and that of the triggering product j_s. The triggering product is not necessarily the product in the most recent purchase. There are multiple heuristic approaches to find the triggering item j_s in the user's purchase history. Here we leverage the transition probability $P(j_s, j_m)$ in Equation 5.13.

$$P(j_a, j_b) = \frac{\#\ (j_a, j_b) + \lambda}{\sum \#\ (j_a, *) + J \cdot \lambda} \qquad (5.13)$$

where $\# \left(j_a, j_b\right)$ is the number of follow-up purchases of j_b that happened after j_a. $\sum \# \left(j_a, *\right)$ is the number of follow-up purchases after j_a. J is the number of products and λ is the smoothing factor, which is set as 0.1.

We first rank all items $\{j_1, ..., j_s, ..., j_{m-1}\}$ that happened less than k_t days before j_m and have $P(j_s, j_m) > Thres_{tran}$. Then we use the top one as the triggering item. Other approaches can be explored in the future work. Although other purchases $\{j_1, ..., j_{s-1}, j_{s+1}, ..., j_{m-1}\}$ in the user history are not treated as the triggering item, they are incorporated into the opportunity model as covariates $\mathbf{x_{m,i}}$.

In this paper, we use the following covariates for each purchase of product j_m made by user u at timestamp t_m: whether user u purchased any product in category m in time bin $t_{b1}, ..., t_{bk}$; how many times the user u purchased any product in category m in time bin $t_{b1}, ..., t_{bk}$; whether the user purchased the product j_m in time bin $t_{b1}, ..., t_{bk}$; how many times the user u purchased the product j_m in time bin $t_{b1}, ..., t_{bk}$; which season t_m is in, whether t_m is in the holiday season, etc. Time bins $t_{b1}, ..., t_{bk}$ are set as one day, one week, one month, two months, three months, six months, one year, etc. We choose these covariates to show the effect of incorporating covariates in the conditional opportunity model. These covariates capture the change of the time distribution with the user's purchase history, the seasonal change, the cyclic pattern, etc. All covariates are normalized in the scale of $[0, 1]$.

5.2 Evaluation Methodology

As we are studying a new problem of recommending the right product at the right time, there is no standard evaluation methodology. We design various experiments to evaluate the performance of the opportunity model. Major research questions that we aim to answer are:

Predictability of the conditional opportunity model How accurate is the conditional time probability that is predicted by the conditional opportunity model? How accurate is the predicted purchase time, compared to the actual purchase time? Are covariates useful?

Predictability of the opportunity model Is the joint purchase probability a good signal of making recommendations? Does it generate a better ranking in traditional zero-query pull-based recommendation systems? Can it help to improve the user satisfaction/utility in proactive push-based recommendation systems?

5.2.1 Evaluation of Conditional Opportunity Model

Metrics

It is a relatively new research topic to predict the purchase time and evaluate the performance of such model. We introduce the following two metrics to evaluate the performance of the conditional opportunity model.

The first metric is the perplexity of the model. It is motivated by the perplexity metric used to evaluate language models and speech recognition [12]. The perplexity measures how well a model predicts the testing data. It is defined as follows:

$$\begin{aligned} perplexity &= \left(\prod_{m=1}^{M} \prod_{i=1}^{N_m} \frac{1}{P(T \in (y, y+\Delta t] | p\_product)} \right)^{\frac{1}{\sum_{m=1}^{M} N_m}} \\ &= 2^{-\sum_{m=1}^{M} \sum_{i=1}^{N_m} \frac{1}{\sum_{m=1}^{M} N_m} log_2 P(T \in (y, y+\Delta t] | p\_product)} \end{aligned} \tag{5.14}$$

where $P(T \in (y, y + \Delta t]|p\_product)$ is defined in Equation 5.11, and i is the index for a testing data point. A better model tend to give a higher data likelihood to the actual follow-up purchase time in the testing data, thus they have lower *perplexity*, which means they are less surprised by the testing data.

The second metric focuses on the difference between the estimated time $y_{m,i}$ and the actual time $\hat{y}_{m,i}$. After a model predicts the distribution $p(y_{m,i}|p\_purchase)$ of the purchase time, we use the median of the distribution as the estimated purchase time $\hat{y}_{m,i}$. The median of the distribution is $exp(\beta_m \mathbf{x_{m,i}})^{-\frac{1}{r_m}} (\ln 2)^{\frac{1}{r_m}}$. The error across all testing data can then be used to analyze and compare. The smaller the error, the better the model. Here we use three types of errors: mean absolute error(MAE), mean squared error(MSE) and mean absolute percentage error(MAPE $= \frac{1}{\sum_{m=1}^{M} N_m} \sum_{m=1}^{M} \sum_{i=1}^{N_m} \frac{|\hat{y}_{m,i} - y_{m,i}|}{y_{m,i}}$).

Conditional Opportunity Models to Compare

In our experiments, we compare the following four conditional opportunity models.

Uniform assigns a uniform distribution to all time. $p(y_{m,i}$

$|p\_purchase) = \frac{1}{T_{uniform}}$ where $T_{uniform}$ is the maximum time in consideration. We set $T_{uniform} = 500$, i.e., 500 days, arbitrarily in this paper.

O-One is a conditional opportunity model that fits a **single** set of parameters with no covariates (i.e., a basic Weibull distribution) to the purchase data. All follow-up purchases in different category m use the same model.

O-Dest is a hierarchical conditional opportunity model that fits one Weibull distribution per product category, with no covariates.

O-DestCov further incorporates covariates into *O-Dest*. In this case, the probability density function changes for each user and product as values of the associated covariates change over time.

All models smooth the conditional probability estimation $P(T \in (y, y+\Delta t]|p\_product) = \max(P(\text{minThres}, T \in (y,$
$y + \Delta t]|p\_product))$ to avoid having a probability that is too low (such as when there is no triggering item before the purchase). minThres is set as 0.001. Δt in Equation 5.11 is set as 7 (i.e., 7 days) for all models during the prediction step. The threshold for the transition probability to consider the an item j_s as the triggering item is set as 0.01.

5.2.2 Evaluation of Opportunity Model

Metrics

Here we evaluate the predictability of the opportunity model with the joint purchase probability in two scenarios.

1) [**Zero-query pull-based scenario**] assumes the user comes to the site to look for products to purchase without issuing any search query. In this scenario, the goal is to discover products that the user would purchase and recommend them to the user proactively. The system ranks all products by their joint purchase probability and recommends top K products to the user.

The evaluation metric in this scenario is the conversion rate which reflects whether a user receives at least one good recommendation. Each testing time corresponds to the time when a user comes to the site and makes purchases. Let $S_{purchased}$ contain all unique products in the order. Let $C_{purchased}$ contain all unique categories in the order. Let $S_{K,recommended}$ contain top K unique product recommendations. Let $C_{K,recommended}$ top K unique category recommendations. $CR_{product}$ is the conversion rate at the product level and $CR_{category}$ is the conversion rate at the category level. Significance level of 0.05 with the paired two-tailed t-test is used to compare two models.

$$CR_{product}@K = \begin{cases} 1 & S_{purchased} \cap S_{K,recommended} \neq \varnothing \\ 0 & otherwise \end{cases}$$

$$CR_{category}@K = \begin{cases} 1 & C_{purchased} \cap C_{K,recommended} \neq \varnothing \\ 0 & otherwise \end{cases}$$

Table 5.1: The utility set of the recommender system. There are four types of utilities, depending on whether the system recommends the item to the user and whether the user purchases the item.

	show:Y	show:N
accept:Y	u_{TP}	u_{FN}
accept:N	u_{FP}	u_{TN}

2) **[Push-based email promotion scenario]** assumes that recommender systems send email/message proactively to a user regularly regardless of whether the user comes to the site or not.

The evaluation metric in this scenario is the average utility/user satisfaction. The utility for each type of recommendations is shown in Table 5.1. The $utility_g$ for each email of recommendations is calculated by Equation 5.15.

$$\begin{aligned} utility_g = \sum(&u_{TP} I_{show,accept} + u_{FP} I_{show,\bar{accept}} \\ &+ u_{FN} I_{\bar{show},accept} + u_{TN} I_{\bar{show},\bar{accept}}) \end{aligned} \tag{5.15}$$

I_* is the indicator function where $I_* = 1$ if $*$ is true. Unlike traditional metrics such as conversion rate@K, the utility metric considers both the positive effect for *good* recommendations and the negative effect for *bad* recommendations. The higher the utility, the better the model.

To achieve a better utility, the system with a filtering component should send emails only if the expected utility of adding the product is higher than zero (i.e. the joint probability is higher than the threshold). The recommendation threshold is automatically determined by the following equation [20]: $\frac{u_{FP}-u_{TN}}{u_{FP}-u_{TN}+u_{FN}-u_{TP}}$.

We don't have a real push-based email promotion system for a user study. Instead, we create an evaluation dataset with the purchase data from a pull-based e-commerce website.

The goal is to evaluate each model's predictability of discovering the "real opportunity" and avoiding the "fake opportunity" in the email marketing. For each purchase at time t in the testing dataset, we let each model consider two opportunities: sending a recommendation email the weekend right before t and sending an email on some other random weekend before t. Since existing models can not tell whether to send an email or not, they always send an email with top K recommendations for each opportunity. The opportunity model with a filtering component will add a product to the email if the expected utility of adding the product is higher than zero. If no product is added to the email, the opportunity model would skip this opportunity and do not send an email. Recommendations in the email are compared with actual products that the user purchases in the week after the email is sent. Assume that there are G opportunities to send recommendation emails in the testing period. The average utility/user satisfaction $utility$ can be calculated as following: $utility = \frac{\sum_{g=1}^{G} utility_g}{G}$.

Recommendation Models to Compare

We choose the following recommendation models to compare:

TopPop recommends most popular products to the user.

SVD is a widely-used recommendation algorithm with a decent performance on the well-known Netflix competition. It is the basis of several recommendation algorithms based on latent factors.

SVD.util is the state-of-art recommendation algorithm in the e-commerce domain without time consideration. It modifies SVD with the marginal net utility framework [84].

Regression.Model is an alternative new approach to predict the probability of a user purchasing a product at a particular time using a logistic regression model, with various important time-dependent features. This model contains features such as the SVD score of the product, whether the product/category has been purchased t days ago, how many time the product/category has been purchased t days ago, etc.

Conditional.Opportunity.Model recommends products based on $P(T \in (y, y + \Delta t])$, which is estimated by the hierarchical conditional opportunity model with covariates.

Opportunity.Model recommends products based on the joint probability of a user making a purchase of a product in the near future($P(p\_product, T \in (y, y + \Delta t]))$.

Opportunity.Model.Filtering adds a filtering component to *Opportunity.Model*. In the push-based email scenario, it adds a product to the promotion email only if the expected utility of adding the product is higher than zero.

5.2.3 Dataset

The purchase history from 2004-01-01 to 2009-03-08 collected on a real-world e-commerce website, shop.com, is used for our experiments. We use all purchases that have category information of the product. Tail users that made less than 5 product purchases are filtered out in the training data, which follows the similar pre-processing in related work [84, 101]. In addition, 10 possible spam users that made more than 200 product purchases are filtered out as well. The remaining data contains 11,351 users and 67,291 products. There are 105,550 unique (user, product) pairs. This user-product matrix is quite sparse, with only 0.014% density. There are 380 categories in total.

To evaluate the conditional opportunity model as in Section 5.2.1, 10-fold cross validation is used. To evaluate the opportunity model with the joint purchase probability as in Section 5.2.2, we sort all purchase history by time. The first 90% is used as the training data (data before 2008-11-24) and the last 10% is used as the testing data (data after 2008-11-24). There are 7,014 testing cases in total. At the product level, there are 1,143 repurchase cases(16.29%) and 6,269 new purchase cases(89.37%). At the categorical level, there are 2,850 repurchase cases(i.e. 40.63% cases with purchase from the same category) and 4,850 new purchase cases(i.e. 69.14% cases with purchase from a new category).

To train the regression model with SVD as one of the features, the first half training data is used to train the SVD model and the second half training data is used to learn coefficients in the regression model. The number of latent factors for all SVD-related models is set to 50. In the recommendation step, all models recommend top K (K=5) products to the user. To save the computation time, $Regression.Model$, $Conditional.Opportunity.Model$ and $Opportunity.Model$ ra-nk among top N recommendations from SVD and selects top K to recommend, where $N = 100$ and $K = 5$.

The time density plots of some common transitions from products in the *Baby—Feeding* category to the target item j_m are shown in Figure 5.3. This supports our motivation of identifying triggering items in the user history. For example, users who purchased from *Baby—Feeding* would purchase items in other *Baby* related categories in the future. The purchase time of the follow-up purchase does follow different distributions for different products or different covariates (take the triggering product as a covariate). For example, users who purchased from the *Baby—Feeding* category would purchase products from this category again in one month. Later on when the baby grows up, they would purchase products from *Toys—Board, Card&Dice Games.*

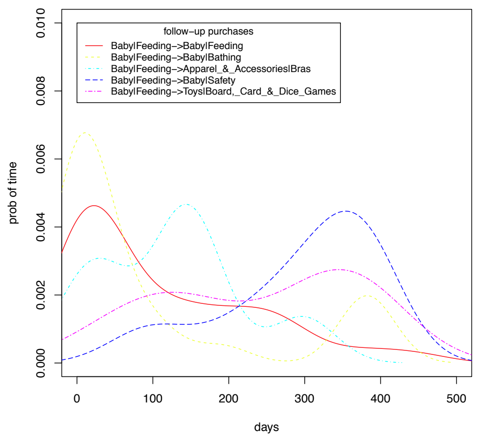

Figure 5.3: Density plot of the purchase time between different follow-up purchases from Baby—Feeding

Table 5.2: Perplexity of different conditional opportunity models in 10-fold cross validation.

Data	Uniform	O-One	O-Dest	O-DestCov
All purchases	46.95	23.75	22.45	**18.95**
repurchases	58.99	16.68	14.80	**9.75**
new purchases	44.25	26.04	25.01	**22.48**

5.3 Experimental Results

5.3.1 Analysis of Conditional Opportunity Model

In this section, we first evaluate the performance of the conditional time probability that is predicted by the conditional opportunity model. The perplexity of all conditional opportunity models in Section 5.2.1 are compared in Table 5.2. All conditional opportunity models have lower perplexity than the baseline model *Uniform*. This demonstrates that conditional opportunity models have better predictability of the time-based purchase probability in the data. Among all opportunity models, *O-DestCov* achieves the lowest perplexity, followed by *O-Dest*, and *O-One*. *O-DestCov* incorporates related covariates in addition to fitting parameters of a conditional

Table 5.3: Error between the estimated purchase time and the actual purchase time. MAE stands for mean absolute error. MSE stands for mean squared error. MAPE stands for mean absolute percentage error.

Data	Error rate	Uniform	O-One	O-Dest	O-DestCov
All Purchases	MAE	159.74	105.279	102.617	**77.695**
	MSE	30859.346	23322.373	21920.462	**14335.505**
	MAPE	16.592	3.796	3.763	**2.434**
Repurchases	MAE	183.337	63.7	62.119	**34.536**
	MSE	37728.853	9406.591	8799.832	**4456.393**
	MAPE	27.895	6.165	5.631	**0.658**
New Purchases	MAE	154.244	114.956	112.042	**87.736**
	MSE	29260.204	26560.746	24973.753	**16634.239**
	MAPE	13.964	3.246	3.329	**2.847**

opportunity model for each category m. It shows the importance of considering covariates when modeling the purchase time of a follow-up purchase.

To further analyze the effect of covariates, we compare the perplexity of all models in the repurchase data and the new purchase data in Table 5.2. As expected, the major gain is in reducing the perplexity of repurchases. More feature exploration would be useful to improve the prediction of new purchases in the future.

Now we compare the estimated purchase time with the actual time of a follow-up purchase in Table 5.3. All opportunity models perform better than the baseline model $Uniform$. *O-DestCov* gives the most accurate estimation of the purchase time, followed by *O-Dest*, and *O-One*. According to MAE, the estimated purchase time from *O-DestCov* is 77 days away(higher or lower) from the actual time. It is a decent performance given that the total time range is 500 days. Further analysis shows that *O-DestCov* predicts more accurately in both the repurchase and the new purchase data.

Table 5.4: Conversion rate of recommendation models in the zero-query pull-based scenario. Numbers in bold are significantly better than the corresponding value in baseline models.

Product level						
	All purchases		Repurchases		New purchases	
Model	CR@1	CR@5	CR@1	CR@5	CR@1	CR@5
TopPop	0.0	0.0155	0.0	0.0936	0.0	0.0003
SVD	0.0134	0.0344	0.0822	0.2073	0.0	0.0011
SVD.util	0.0287	0.0369	0.1750	0.2248	0.0002	0.0003
Regression.Model	**0.0339**	**0.0525**	**0.2082**	**0.3202**	0.0	0.0003
Conditional.Opportunity.Model	0.0160	0.0282	0.0962	0.1706	0.0003	0.0006
Opportunity.Model	0.0289	**0.0452**	**0.1750**	**0.2712**	0.0005	0.0011

Category level						
	All purchases		Repurchases		New purchases	
Model	CR@1	CR@5	CR@1	CR@5	CR@1	CR@5
TopPop	0.0204	0.0848	0.0130	0.1382	0.0219	0.0435
SVD	0.0577	0.1688	0.1309	0.3267	0.0066	0.0540
SVD.util	0.0751	0.1344	0.1779	0.2453	0.0041	0.0511
Regression.Model	**0.0999**	**0.1792**	**0.2330**	**0.3646**	0.0076	0.0472
Conditional.Opportunity.Model	**0.1102**	**0.2154**	**0.2467**	**0.3856**	0.0144	**0.0907**
Opportunity.Model	**0.1313**	**0.2345**	**0.3091**	**0.4495**	0.0082	**0.0816**

5.3.2 Analysis of Opportunity Model

Pull-based Scenario

Now we evaluate the performance of the opportunity model with the joint purchase probability. The conversion rate of different recommendation models in the zero-query pull-based scenario is compared in Table 5.4.

The new opportunity model does achieve higher conversion rate at both the product level and the category level. We did some further analysis to see the contribution of the regression model ($P(product)$) and the conditional opportunity model ($P(time|product)$). We found that the contribution of the conditional opportunity model is mainly at the category level. It is not surprising because the conditional opportunity model is designed at the category level. In the

Table 5.5: Analysis of how frequently different models recommend repurchase products at top 1 place.

Model	total number of repurchase recommendations	number of failures	number of success	success rate
TopPop	1	1	0	0
SVD	2093	1999	94	0.04
SVD.util	2179	1979	200	0.09
Regression.Model	4134	3896	238	0.05
Conditional.Opportunity.Model	1166	1056	110	0.09
Opportunity.Model	2833	2633	200	0.07

future, the product-level model can be further explored. The key is to solve the sparsity and scalability issue.

We further analyze the performance of the conversion rate in different purchase scenarios. In Table 5.4, we show the conversion rate of repurchases and new purchases. It is clear that the major contribution of the opportunity model is in the repurchase scenario. The observation is consistent with the evaluation of the conditional opportunity model in Section 5.3.1: the conditional opportunity model predicts more accurately in repurchases.

We also analyze how frequently different models recommend repurchase products in Table 5.5. We first count the number of times that a model recommends a repurchase product at the top 1 product recommendation. Among all these repurchase recommendations, we analyze how many times the recommendation works and how many times it fails. As we can see, $SVD.Util$ has higher success rate than $Regression.Model$ while $Regression.Model$ is more aggressive in recommending repurchase products. $Conditional.Opportunity.Model$ itself has the highest success rate. By combining it with $Regression.Model$, $Opportunity.Model$ significantly reduces the cases of inappropriate recommendation of repurchase products.

Push-Based Scenario

Now we evaluate all models in the push-based scenario. In the *Opportunity.Model. Filtering* model, a product is recommended only if its joint purchase probability is higher than the threshold. We evaluate the model with three sets of utility at both the product level (i.e. true positive means that the user purchased the exact product) and the category level (i.e. a true positive means that the user purchased a product in same category as the recommendation). The results are shown in Table 5.6. There are 3,014 email opportunities in the testing period. 55% of them have follow-up purchases in the following week. Among those with follow-up purchases, the average number of purchases is 2.227.

In the *first* utility set, the utility u_{TP} for a good recommendation that the user purchases is set to 3. When the system shows a product and the user does not purchase it, the utility u_{FP} is -1. The resulting filtering threshold is 0.25. In this case, *Opportunity.Model.Filtering* has a better utility than other models by filtering out many false alarms. The average utility at the category level is higher, which is expected. It rewards a recommendation if it matches the user's purchase at the category level, which is an easier task. In general, the model with higher conversion rate in the pull-based scenario has high utility in the push-based scenario.

In the *second* utility set, the utility u_{TP} is set to 10, indicating that the user is more tolerant to **bad** recommendations. Thus the threshold is lower, being 0.09. The coverage of *Opportunity.Model.Filtering* increases while the average utility drops. There is a tradeoff between the utility and the recommendation coverage, which can be tuned in a real-world application by a user or the system designer.

In the *third* utility set, the utility u_{FN} is set to -1. It is the penalty for missing to recommend a product that would be purchased by the user in the following week. Thus

Table 5.6: Average utility of email recommendations in the push-based scenario. *Coverage* is the percentage of emails that are sent among all possible opportunities. *Rec count* is the average number of unique recommendations in a email when the email is sent.

utility set 1
u_{TP} = 3, u_{FN} = 0
u_{FP} = -1, u_{TN} = 0
Threshold = 0.25

Product level

Model	TopPop	SVD	SVD.util	Opportunity.Model	Opportunity.Model Filtering
Utility	-4.994	-4.862	-4.768	-4.810	**-0.048**
Coverage	1	1	1	1	0.023
Rec count	5	5	5	5	2.432

category level

Model	TopPop	SVD	SVD.util	Opportunity.Model	Opportunity.Model Filtering
Utility	-4.823	-4.515	-4.570	-4.255	**0.006**
Coverage	1	1	1	1	0.023
Rec count	5	5	5	5	1.081

utility set 2
u_{TP} = 10, u_{FN} = 0
u_{FP} = -1, u_{TN} = 0
Threshold = 0.09

Product level

Model	TopPop	SVD	SVD.util	Opportunity.Model	Opportunity.Model Filtering
Utility	-4.983	-4.621	-4.363	-4.477	**-0.399**
Coverage	1	1	1	1	0.180
Rec count	5	5	5	5	3.007

Category level

Model	TopPop	SVD	SVD.util	Opportunity.Model	Opportunity.Model Filtering
Utility	-4.514	-3.667	-3.819	-2.951	**0.282**
Coverage	1	1	1	1	0.180
Rec count	5	5	5	5	1.880

utility set 3
u_{TP} = 3, u_{FN} = -1
u_{FP} = -1, u_{TN} = 0
Threshold = 0.20

Product level

Model	TopPop	SVD	SVD.util	Opportunity.Model	Opportunity.Model Filtering
Utility	-4.992	-4.828	-4.710	-4.762	**-0.066**
Coverage	1	1	1	1	0.033
Rec count	5	5	5	5	2.356

Category level

Model	TopPop	SVD	SVD.util	Opportunity.Model	Opportunity.Model Filtering
Utility	-4.779	-4.394	-4.463	-4.069	**0.015**
Coverage	1	1	1	1	0.033
Rec count	5	5	5	5	1.115

the coverage of $Opportunity.Model.Filtering$ is higher to avoid missing good recommendations with the threshold being 0.020. It still achieves the highest utility among all models.

5.4 Summary and Contribution

In this work, we propose to develop the *opportunity model* to predict the probability of a user purchasing a product at a particular time. This is achieved by modeling the joint probability of time and product, which can be calculated by combining a proportional hazards model and a logistic regression model. The former component estimates the density function of time while the latter one estimates the probability of a user purchasing a product. The Weibull distribution with various covariates can capture various effects that affect the purchase time, which are not captured by prior work. The hierarchical Bayesian modeling framework enables us to handle the data sparsity issue and the variational Bayesian inference algorithm helps to make the parameter estimation more computationally efficient. This joint probability will guide the system to make a recommendation to the user at the right time. It helps to improve the recommender system's conversion rate. More importantly, if there is any negative penalty associated with recommending a product at the wrong time, the new approach can help to improve the overall utility/user satisfaction. The opportunity model captures the true opportunity when the joint probability is high. It can be leveraged to develop a push-based recommender systems (such as email-based marketing) with a solid statistical foundation for utility optimization.

The major contribution of this work includes the following:

- Propose the research problem in the community of recommender systems: When is the right time to make recommendation of the right product in the e-commerce domain?

- Design the **opportunity model** to predict the joint purchase probability of the recommendation product and the recommendation time. Extend the proportional hazards model in statistics with the hierarchical Bayesian framework. Derive detailed inference steps with the variational Bayesian algorithm.
- Leverage the joint purchase probability in both the zero-query pull-based scenario and the proactive push-based email promotion scenario.Experimental results show that the opportunity model significantly improves the user satisfaction and the conversion rate of the system.

To further explore this topic, researchers could compare the opportunity model with other methods. Straightforward approaches include using the average of the purchase time in each category. More advanced baselines could be autoregressive moving average (ARMA). Online A/B testing with real push-based recommender system would be very valuable to show the contribution of the opportunity model.

Chapter 6

Conclusion and Future Work

In this chapter, we conclude contributions of the session-aware recommender systems in e-commerce, followed by the limitation of the work and future research directions.

6.1 Contribution

6.1.1 Contribution in the Field of Recommender Systems

First, we propose a unified recommendation and search model in Section 3. It is useful in both the field of recommender system and the field of search engine. Recommender systems and search engines help consumers locate products to purchase, songs to listen, movies to watch, etc. Although they have similar goals, they are usually developed and used separately in a site. In the research community, they are also studied as two separate problems. As shown in Section 3, it is worthwhile to study two problems together and leverage the unified system to make recommendations. The unified system could identify both the user's long-term preference and the user's short-term interest. Such capacity to predict the user's purchase intention could help to get a better ranking, which is the key task of search engines. It could be leveraged in other domains in addition to the e-commerce domain.

Second, we introduce the idea of utility into the field of recommender system, both in Section 4 and Section 5. Utility indicates the user's satisfaction or pleasure by purchasing an item, or listening to a song, watching a movie, etc. It contains more information than the pure rating, which is widely used by traditional recommender systems. In the design of recommender systems in different domains, one key analysis should be about the user's utility. For example, the motivation questions include "how does a user feel satisfied in the system?", "how should the system improve the user's utility?", etc.

Third, we tackle an important question in the field of recommender systems in Section 5: When to make the right recommendation? We propose the opportunity model to determine the probability of purchasing at a certain time, which is inspired by the survival analysis in statistics. This model can be leveraged in recommender systems in other domains, such as the job recommendation, the movie recommendation and so on.

6.1.2 Contribution beyond the Field of Recommender Systems

The major contribution beyond the field of recommender system is about identifying the user's purchase intention. The model in Section 3 focuses on modeling the purchase intention within a single session. The models in Section 4 and Section 5 focus on modeling the purchase intention across sessions. In addition to leveraging the purchase intention in making recommendations, the system could make better marketing campaigns to attract potential users. For example, if the user is likely to make a repurchase in the session, the system could recommend products that were purchased by the user. If the user is likely to make a new purchase, the system could make some promotions in the category that is likely to be purchased by the user.

In addition, we derive the inference steps of each model, which can be further leveraged in the future. This is especially true for the opportunity model with Bayesian framework in Section 5. We derive the detailed inference step with variational Bayesian method, which could benefit the research of such model in statistics. The complete derivation is in the Appendix.

6.2 Future Research Directions

6.2.1 Computational Efficiency with Online Updating

In this dissertation, we mostly focus on building the recommendation model to have better predictability of a user's purchase intention. Yet more advanced models might require higher computational power. In real world, the recommendation model in Section 4 and Section 5 might require online updating daily or weekly with the updated user history. There is some existing work of online updating of matrix factorization [61]. It is an important research question to explore in the future.

In Section 5, we only learn the category-level model for conditional opportunity model due to the limited number of research data available and our own computational limits. The model clearly works well at the category level prediction. Another follow-up work is to evaluate this approach on a large-scale system to learn the product-level model, which might lead to better prediction at the product level.

The scalability issue could be explored with a larger dataset in e-commerce, as well as datasets in other domains [80, 82, 87] including job recommendation, tag recommendation, etc.

6.2.2 New Purchase Recommendation

The repurchase product recommendation has been deeply analyzed in all models with decent performance. Yet the performance of new product recommendation is still not satisfying. It is worthwhile to explore the underlying factors that would affect a user's choice of a new product, including the product promotion information, etc.

As discussed in Chapter 4, there are consumable products and durable products. The recommendation of the former type of products is analyzed and improved significantly with the session-aware recommender system. The recommendation of the durable products is a harder task. If it is a follow-up purchase that is related to the user's purchase history, the opportunity model might discover such behavior. Otherwise it would be new product purchase which needs more research in the future.

6.2.3 Feature Engineering with Machine Learning Techniques

As shown in Section 3 and Section 5, features/covariates are important factors to determine the recommendation performance. In this dissertation, we explore features in the following aspects: time-dependent internal features, time-independent internal features, time-dependent external features, and time-independent internal features. More extensive feature engineering would help the system to achieve higher conversion rate, especially in the new purchase recommendation. Existing advanced machine learning techniques could be leveraged here, such as the deep learning.

6.2.4 Leverage in the Marketing Campaign

The user purchase intention could be further leveraged in the marketing campaign design. Some directions have been discussed in Section 3.4.4. The unified model with three levels, *Graphical.U*, has a better explanatory power and can predict user's purchase state (repurchase, new purchase, or variety seeking), which might be valuable for marketing and advertising. The transparency may make it easier for e-commerce websites to provide better explanations to users, which is essential for a successful recommender system [57]. Future research with an online system that can interact with users directly will be useful to study these problems. It would be helpful to carry out experiments with a real push-based system to evaluate the effectiveness of the opportunity model in a real user study.

Appendix A

Appendix

A.1 Parameter Inference of Conditional Opportunity Model with Variational Bayesian

There is no closed-form solution for the estimation of the model parameters. We follow the variational Bayesian method[4] for constrained (approximate) optimization to derive an iterative process to find the approximate solution. Maximizing the likelihood in Equation 5.5 is equivalent to maximizing the log likelihood $L(\phi)$.

$$L(\phi) = \ln p(D|\phi) = \sum_{m=1}^{M} \ln p(y_m|\phi) = \sum_{m=1}^{M} \ln \int p(\theta_m, y_m|\phi) \mathrm{d}\theta_m$$

We can simplify the problem by introducing an auxiliary distribution $q(\theta_m)$ for each hidden variable θ_m [4]. In the variational approach, we constrain $q(\theta_m)$ to be a particular tractable form for computational efficiency. In particular, we assume that $q(\beta_m) = N(\mu_{\beta_m}, \Sigma_{\beta_m})$ and $q(\gamma_m) = Gamma(a_{\gamma_m}, b_{\gamma_m})$. The process to infer parameters is to iterate between the following

E-step and M-step until convergence.

E-Step

In the **E-step**, we infer the posterior distributions over hidden variables θ_m given the current parameter setting ϕ. There is no closed-form solution. Instead, we find a tractable approximation of the posterior distribution of θ_m given ϕ (i.e, $q(\theta_m)$ that maximizes $L(\phi)$).

$$\begin{aligned}
L(\phi) &= \sum_{m=1}^{M} \ln p(y_m|\phi) \\
&= \sum_{m=1}^{M} \ln \int q(\theta_m) \frac{p(\theta_m, y_m|\phi)}{q(\theta_m)} \mathrm{d}\theta_m \\
&\geq \sum_{m=1}^{M} \int q(\theta_m) \ln \frac{p(\theta_m, y_m|\phi)}{q(\theta_m)} \mathrm{d}\theta_m \\
&= \sum_{m=1}^{M} \int q(\theta_m) \ln p(y_m|\phi) \mathrm{d}\theta_m + \sum_{m=1}^{M} \int q(\theta_m) \ln \frac{p(\theta_m|y_m, \phi)}{q(\theta_m)} \mathrm{d}\theta_m \\
&= \sum_{m=1}^{M} \int q(\theta_m) \ln p(y_m|\phi) \mathrm{d}\theta_m - \sum_{m=1}^{M} \int q(\theta_m) \ln \frac{q(\theta_m)}{p(\theta_m|y_m, \phi)} \mathrm{d}\theta_m \\
&\equiv \boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi)
\end{aligned} \tag{A.1}$$

To maximize $L(\phi)$, it is the same as minimize the following equation to find each distribution $q(\theta_m)$:

$$q(\theta_m)^{(t)} = \arg\min_{q(\theta_m)} \int q(\theta_m) \ln \frac{q(\theta_m)}{p(\theta_m|y_m, \phi)} \mathrm{d}\theta_m \tag{A.2}$$

Given that $p(\theta_m|y_m,\phi) = \frac{p(y_m|\theta_m)p(\theta_m|\phi)}{p(y_m|\phi)}$, we have:

$$\begin{aligned}
q(\beta_m,\gamma_m) &= \arg\min_{q(\beta_m,\gamma_m)} \int \frac{q(\beta_m,\gamma_m)}{P(\beta_m,\gamma_m|y_m,\phi)} \mathrm{d}\beta_m \mathrm{d}\gamma_m \\
&= \arg\min_{q(\beta_m,\gamma_m)} \int \frac{q(\beta_m,\gamma_m)}{\frac{P(\beta_m,\gamma_m,y_m|\phi)}{P(y_m|\phi)}} \mathrm{d}\beta_m \mathrm{d}\gamma_m \\
&\propto \arg\min_{q(\beta_m,\gamma_m)} \int \frac{q(\beta_m,\gamma_m)}{P(\beta_m,\gamma_m,y_m|\phi)} \mathrm{d}\beta_m \mathrm{d}\gamma_m \\
&= \arg\min_{q(\beta_m,\gamma_m)} \int [\frac{q(\beta_m,\gamma_m)}{P(\beta_m,\gamma_m|\phi)} \frac{1}{P(y_m|\beta_m,\gamma_m)}] \mathrm{d}\beta_m \mathrm{d}\gamma_m \\
&= \arg\min_{q(\beta_m,\gamma_m)} \int \frac{q(\beta_m,\gamma_m)}{P(\beta_m,\gamma_m|\phi)} \mathrm{d}\beta_m \mathrm{d}\gamma_m - \int q(\beta_m,\gamma_m) P(y_m|\beta_m,\gamma_m) \mathrm{d}\beta_m \mathrm{d}\gamma_m \\
&= \arg\min_{q(\beta_m,\gamma_m)} KL[q(\beta_m,\gamma_m)||p(\beta_m,\gamma_m|\phi)] - \int q(\beta_m,\gamma_m) \ln p(y_m|\beta_m,\gamma_m) \mathrm{d}\beta_m \mathrm{d}\gamma_m
\end{aligned} \tag{A.3}$$

The first part in Equation A.3 is the KL-divergence between the posterior distribution $q(\theta_m)$ and the prior distribution $p(\theta_m|\phi)$.

$$KL[q(\beta_m,\gamma_m)||p(\beta_m,\gamma_m|\phi)] = KL[q(\beta_m)||p(\beta_m|\phi)] + KL[q(\gamma_m)||p(\gamma_m|\phi)]$$

The KL-divergence between two Gaussian distributions is

$$\begin{aligned}
&KL[q(\beta_m)||p(\beta_m|\phi)] \\
&= \frac{1}{2}[tr(\Sigma_\beta^{-1}\Sigma_{\beta_m}) + (\mu_\beta - \mu_{\beta_m})^T \Sigma_\beta^{-1} (\mu_\beta - \mu_{\beta_m}) \\
&- \ln(\frac{\det \Sigma_{\beta_m}}{\det \Sigma_\beta}) - k]
\end{aligned}$$

The KL-divergence between two gamma distributions is

$$\begin{aligned}
&KL[q(\gamma_m)||p(\gamma_m|\phi)] \\
&= (a_{\gamma_m} - a_\gamma)\Psi(a_{\gamma_m}) - \log\Gamma(a_{\gamma_m}) + \log\Gamma(a_\gamma) \\
&+ a_\gamma(\log b_{\gamma_m} - \log b_\gamma) + a_{\gamma_m}\frac{b_\gamma - b_{\gamma_m}}{b_{\gamma_m}}
\end{aligned}$$

where $\Psi(*)$ is the digamma function.

The second part in Equation A.3 is to maximize the data likelihood of $y_m = \{y_{m,1}, ..., y_{m,i}, ..., y_{m,N_m}\}$ with the current $\theta_m = \{\beta_m, \gamma_m\}$.

$$\begin{aligned}
&\sum_{i=1}^{N_m} \int q(\theta_m) \ln p(y_{m,i}|\theta_m) \mathrm{d}\theta_m \qquad \text{(A.4)}\\
=&\sum_{i=1}^{N_m} \int\int q(\beta_m) q(\gamma_m) \ln p(y_{m,i}|\beta_m, \gamma_m) \mathrm{d}\beta_m \mathrm{d}\gamma_m\\
=&\sum_{i=1}^{N_m} \int\int q(\beta_m) q(\gamma_m) [\ln \gamma_m + (\beta_m)^T \mathbf{x_{m,i}} + (\gamma_m - 1) \ln y_{m,i} - exp(\beta_m^T \mathbf{x_{m,i}}) y_{m,i}^{\gamma_m}] \mathrm{d}\beta_m \mathrm{d}\gamma_m\\
=&\sum_{i=1}^{N_m} [\int q(\gamma_m) \ln \gamma_m \mathrm{d}\gamma_m + \int q(\beta_m)(\beta_m)^T \mathbf{x_{m,i}} \mathrm{d}\beta_m + \int q(\gamma_m)(\gamma_m - 1) \ln y_{m,i} \mathrm{d}\gamma_m -\\
&\int\int q(\beta_m) q(\gamma_m) exp(\beta_m^T \mathbf{x_{m,i}}) y_{m,i}^{\gamma_m} \mathrm{d}\beta_m \mathrm{d}\gamma_m]\\
=&\sum_{i=1}^{N_m} [E(\ln \gamma_m) + E(\beta_m)^T \mathbf{x_{m,i}} + (E(\gamma_m) - 1) \ln y_{m,i} - E(exp(\beta_m^T \mathbf{x_{m,i}})) E(y_{m,i}^{\gamma_m})]
\end{aligned}$$

The expectations in the above equation are

$$\begin{aligned}
E(\ln \gamma_m) =& \Psi(a_{\gamma_m}) - \ln(b_{\gamma_m})\\
E(\beta_m) =& \mu_{\beta_m}\\
E(\gamma_m) =& \frac{a_{\gamma_m}}{b_{\gamma_m}}\\
E(exp(\beta_m^T \mathbf{x_{m,i}})) =& exp\{\mu_{\beta_m}^T \mathbf{x_{m,i}} + \frac{1}{2} \mathbf{x_{m,i}}^T \Sigma_{\beta_m} \mathbf{x_{m,i}}\}\\
E(y_{m,i}^{\gamma_m}) =& \int \frac{b_{\gamma_m}^{a_{\gamma_m}}}{\Gamma(a_{\gamma_m})} \gamma_m^{a_{\gamma_m}-1} exp(-b_{\gamma_m}\gamma_m) y_{m,i}^{\gamma_m} \mathrm{d}\gamma_m\\
=& \int \frac{b_{\gamma_m}^{a_{\gamma_m}}}{\Gamma(a_{\gamma_m})} \gamma_m^{a_{\gamma_m}-1} exp(-b_{\gamma_m}\gamma_m) exp(\ln y_{m,i}\gamma_m) \mathrm{d}\gamma_m\\
=& \int \frac{b_{\gamma_m}^{a_{\gamma_m}}}{\Gamma(a_{\gamma_m})} \gamma_m^{a_{\gamma_m}-1} exp((\ln y_{m,i} - b_{\gamma_m})\gamma_m) \mathrm{d}\gamma_m\\
=& \int \frac{(b_{\gamma_m} - y_{m,i})^{a_{\gamma_m}}}{\Gamma(a_{\gamma_m})} \gamma_m^{a_{\gamma_m}-1} exp(-(b_{\gamma_m} - \ln y_{m,i})\gamma_m) \frac{b_{\gamma_m}^{a_{\gamma_m}}}{(b_{\gamma_m} - y_{m,i})^{a_{\gamma_m}}} \mathrm{d}\gamma_m\\
=& \frac{b_{\gamma_m}^{a_{\gamma_m}}}{(b_{\gamma_m} - \ln y_{m,i})^{a_{\gamma_m}}}
\end{aligned}$$

We can combine the above derivations with Equation A.3, then find $q(\theta_m)$ using the conjugate gradient descent method.

M-Step

In the **M-step**, the goal is to maximize $\boldsymbol{F}(q(\theta_1), ..., q(\theta_M), \phi)$ in Equation A.1 with respect to ϕ given all θ_m. It is same as to maximize the following quantity:

$$\phi^{(t+1)} \leftarrow \arg\max_{\phi} \sum_{m=1}^{M} \int q(\theta_m) \ln p(y_m|\phi) \mathrm{d}\theta_m \tag{A.5}$$

We first derive the closed-form solution for μ_β and Σ_β:

$$\begin{aligned}
(\mu_\beta, \Sigma_\beta) &\leftarrow \arg\max_{\mu_\beta, \Sigma_\beta} \sum_{m=1}^{M} \int q(\beta_m) \ln p(y_m|\beta_m) p(\beta_m|\mu_\beta, \Sigma_\beta) \mathrm{d}\beta_m \\
&\propto \arg\max_{\mu_\beta, \Sigma_\beta} \sum_{m=1}^{M} \int q(\beta_m) \ln p(\beta_m|\mu_\beta, \Sigma_\beta) \mathrm{d}\beta_m \\
&= \arg\max_{\mu_\beta, \Sigma_\beta} \sum_{m=1}^{M} \int q(\beta_m) \ln \frac{1}{\sqrt{(2\pi)^k |\Sigma_\beta|}} exp(-\frac{1}{2}(\beta_m - \mu_\beta)^T \Sigma_\beta^{-1} (\beta_m - \mu_\beta)) \mathrm{d}\beta_m \\
&= \arg\max_{\mu_\beta, \Sigma_\beta} \sum_{m=1}^{M} \ln \frac{1}{\sqrt{(2\pi)^k |\Sigma_\beta|}} - \frac{1}{2} E[(\beta_m - \mu_\beta)^T \Sigma_\beta^{-1} (\beta_m - \mu_\beta)]
\end{aligned} \tag{A.6}$$

By having the first derivative as zero, we have the following closed form:

$$\begin{aligned}
\mu_\beta &= \frac{\sum_m^M \mu_{\beta_m}}{M} \\
\Sigma_\beta &= \frac{\sum_m^M [\Sigma_{\beta_m} + (\mu_{\beta_m} - \mu_\beta)(\mu_{\beta_m} - \mu_\beta)^T]}{M}
\end{aligned} \tag{A.7}$$

We then derive the closed-form solution for a_γ and b_γ:

$$\begin{aligned}
(a_\gamma, b_\gamma) &\leftarrow \arg\max_{a_\gamma, b_\gamma} \sum_{m=1}^{M} \int q(\gamma_m) \ln p(y_m|\gamma_m) p(\gamma_m|a_\gamma, b_\gamma) \mathrm{d}\gamma_m \\
&\propto \arg\max_{a_\gamma, b_\gamma} \sum_{m=1}^{M} \int q(\gamma_m) \ln p(\gamma_m|a_\gamma, b_\gamma) \mathrm{d}\gamma_m \\
&= \arg\max_{a_\gamma, b_\gamma} \sum_{m=1}^{M} \int q(\gamma_m) \ln (b_\gamma^{a_\gamma} \frac{1}{\Gamma(a_\gamma)} \gamma_m^(a_\gamma - 1) exp(-b_\gamma \gamma_m)) \mathrm{d}\gamma_m \\
&= \arg\max_{a_\gamma, b_\gamma} \sum_{m=1}^{M} (\ln b_\gamma^{a_\gamma} - \ln \Gamma(a_\gamma) + (a_\gamma - 1) \int q(\gamma_m) \ln \gamma_m \mathrm{d}\gamma_m - b_\gamma \int q(\gamma_m) \gamma_m \mathrm{d}\gamma_m)) \\
&= \arg\max_{a_\gamma, b_\gamma} \sum_{m=1}^{M} (\ln b_\gamma^{a_\gamma} - \ln \Gamma(a_\gamma) + (a_\gamma - 1) E[\ln \gamma_m] - b_\gamma E[\gamma_m])) \\
&= \arg\max_{a_\gamma, b_\gamma} \sum_{m=1}^{M} (\ln b_\gamma^{a_\gamma} - \ln \Gamma(a_\gamma) + (a_\gamma - 1)(\psi(a_{\gamma_m} - \ln b_{\gamma_m}) - b_\gamma \frac{a_{\gamma_m}}{b_{\gamma_m}}))
\end{aligned} \tag{A.8}$$

By having the first derivative as zero, we have the following closed form:

$$\begin{aligned}
a_\gamma &= \Psi^{-1}(\ln b_\gamma + \frac{\sum_m^M \Psi(a_{\gamma_m}) - \ln(b_{\gamma_m})}{M}) \\
b_\gamma &= \frac{M a_\gamma}{\sum_m^M \frac{a_{\gamma_m}}{b_{\gamma_m}}}
\end{aligned}$$

where $\Psi^{-1}(*)$ is the inverse digamma function.

Bibliography

[1] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 19–28, New York, NY, USA, 2009. ACM.

[2] Charu C. Aggarwal, Joel L. Wolf, Kun lung Wu, and Philip S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the Fifth ACM SIGKDD*, 1999.

[3] William J. Baumol and Alan S. Blinder. *Microeconomics: Principles and Policy.* South-Western College Pub, 11 edition, July 2008.

[4] M.J. Beal. *Variational algorithms for approximate Bayesian inference.* PhD thesis, University of London, 2003.

[5] J. Bennett and S. Lanning. The netflix prize. *Proceedings of KDD Cup and Workshop*, 2007.

[6] Carol A. Kohn Berning and Jacob Jacoby. Patterns of information acquisition in new product purchases. *The Journal of Consumer Research*, 1(2):18–22, Sep 1974.

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[8] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. pages 43–52. Morgan Kaufmann, 1998.

[9] Christopher J. C. Burges, Krysta Marie Svore, Paul N. Bennett, Andrzej Pastusiak, and Qiang Wu. Learning to rank using an ensemble of lambda-gradient models. *Journal of Machine Learning Research - Proceedings Track*, 14:25–35, 2011.

[10] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, November 2002.

[11] Harr Chen and David R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th ACM SIGIR'06*, pages 429–436. ACM, 2006.

[12] Stanley Chen, Douglas Beeferman, and Ronald Rosenfeld. Evaluation metrics for language models. In *DARPA Broadcast News Transcription and Understanding Workshop (BNTUW)*, Lansdowne, Virginia, USA, February 1998.

[13] Ye Chen and John F. Canny. Recommending ephemeral items at web scale. In *Proceedings of the 34th international ACM SIGIR'11*, pages 1013–1022. ACM, 2011.

[14] Yunfei Chen, Lanbo Zhang, Aaron Michelony, and Yi Zhang. 4is of social bully filtering: identity, inference, influence, and intervention. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2677–2679. ACM, 2012.

[15] C.W. Cobb and P.H. Douglas. A theory of production. *American Economic Review*, 18:139 – 165, 1928.

[16] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM Recommender systems*, pages 39–46, New York, NY, USA, 2010. ACM.

[17] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[18] Yi Ding and Xue Li. Time weight collaborative filtering. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492, New York, NY, USA, 2005. ACM.

[19] Micah Dubinko, Ravi Kumar, Joseph Magnani, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. Visualizing tags over time. *ACM Trans. Web*, 1(2), August 2007.

[20] Charles Elkan. The foundations of cost-sensitive learning. In *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2*, IJCAI'01, pages 973–978, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[21] Daniel Fleder and Kartik Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Manage. Sci.*, 55:697–712, May 2009.

[22] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

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

[24] Qi Guo and Eugene Agichtein. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *Proceeding of the 33rd international ACM SIGIR'10*, pages 130–137. ACM, 2010.

[25] Eui-Hong (Sam) Han and George Karypis. Feature-based recommendation system. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 446–452, New York, NY, USA, 2005. ACM.

[26] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR*, pages 230–237, New York, NY, USA, 1999. ACM.

[27] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22:5–53, January 2004.

[28] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th SIGIR*, pages 259–266, New York, NY, USA, 2003. ACM.

[29] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[30] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, 1999.

[31] Wayne D Hoyer. An examination of consumer decision making for a common repeat purchase product. *Journal of Consumer Research: An Interdisciplinary Quarterly*, 11(3):822–29, December 1984.

[32] Rong Jin, Luo Si, ChengXiang Zhai, and Jamie Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of the twelfth CIKM*, pages 309–316, New York, NY, USA, 2003. ACM.

[33] Barbara E. Kahn and Jagmohan S. Raju. Effects of price promotions on variety-seeking and reinforcement behavior. *Marketing Science*, 10(4):316–337, 1991.

[34] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254, New York, NY, USA, 2001. ACM.

[35] Yong Soo Kim, Bong-Jin Yum, Junehwa Song, and Su Myeon Kim. Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. *Expert Syst. Appl.*, 28:381–393, February 2005.

[36] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, John Riedl, and High Volume. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40:77–87, 1997.

[37] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD KDD'08*, pages 426–434, New York, NY, USA, 2008. ACM.

[38] Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proceedings of the 15th ACM*, pages 447–456, New York, NY, USA, 2009. ACM.

[39] Lakshman Krishnamurthi and S. P. Raj. An empirical analysis of the relationship between brand loyalty and consumer price elasticity. *Marketing Science*, 10(2):172–183, 1991.

[40] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Recommendation systems: A probabilistic analysis. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS '98, pages 664–, Washington, DC, USA, 1998. IEEE Computer Society.

[41] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Machine Learning Conference (ML95)*, 1995.

[42] Beibei Li, Anindya Ghose, and Panagiotis G. Ipeirotis. Towards a theory model for product search. In *Proceedings of the 20th ACM WWW'11*, pages 327–336. ACM, 2011.

[43] Wei Li, Debasis Ganguly, and Gareth J. F. Jones. Enhanced information retrieval using domain-specific recommender models. In *Proceedings of the Third international conference on Advances in information retrieval theory*, ICTIR'11, pages 201–212, Berlin, Heidelberg, 2011. Springer-Verlag.

[44] Shiu li Huang. Designing utility-based recommender systems for e-commerce: Evaluation of preference-elicitation methods. *Electronic Commerce Research and Applications*, In Press, Corrected Proof:–, 2010.

[45] Weiyang Lin, Sergio A. Alvarez, and Carolina Ruiz. Collaborative recommendation via adaptive association rule mining. In *Data Mining and Knowledge Discovery*, 2000.

[46] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[47] Chao Liu, Ryen W. White, and Susan Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proceedings of the 33rd international ACM*

SIGIR conference on Research and development in information retrieval, SIGIR '10, pages 379–386, New York, NY, USA, 2010. ACM.

[48] Nathan N. Liu and Qiang Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90, New York, NY, USA, 2008. ACM.

[49] Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In *In NIPS*17*. MIT Press, 2003.

[50] Benjamin Marlin and Richard S. Zemel. The multiple multiplicative factor model for collaborative filtering. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 73, New York, NY, USA, 2004. ACM.

[51] Leigh McAlister and Edgar Pessemier. Variety seeking behavior: An interdisciplinary review. *Journal of Consumer Research*, 9(3):311–22, December 1982.

[52] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. *Commun. ACM*, 43(8):142–151, 2000.

[53] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6:61–82, 2002.

[54] Cassie Mogilner, Tamar Rudnick, and Sheena S. Iyengar. The mere categorization effect: How the presence of categories increases choosers' perceptions of assortment variety and outcome satisfaction. *Journal of Consumer Research*, 35(2):202–215, 2008.

[55] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *DL '00*, pages 195–204, New York, NY, USA, 2000. ACM.

[56] Michael Pazzani, Daniel Billsus, S. Michalski, and Janusz Wnek. Learning and revising user profiles: The identification of interesting web sites. In *Machine Learning*, pages 313–331, 1997.

[57] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM RecSys'11*, pages 157–164. ACM, 2011.

[58] Cox D. R. Regression models and life tables. *Journal of the Royal Statistic Society*, B(34):187–202, 1972.

[59] S P Raj. The effects of advertising on high and low loyalty consumer segments. *Journal of Consumer Research*, 9(1):77–89, June 1982.

[60] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th WWW*, pages 811–820, New York, NY, USA, 2010. ACM.

[61] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 251–258, New York, NY, USA, 2008. ACM.

[62] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.

[63] S. E Robertson. Overview of the OKAPI projects. *Journal of Documentation*, 53(1):3–7, 1997.

[64] Eduardo J. Ruiz, Vagelis Hristidis, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Correlating financial time series with micro-blogging activity. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 513–522, New York, NY, USA, 2012. ACM.

[65] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th WWW conference*, pages 285–295, New York, NY, USA, 2001. ACM.

[66] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167, New York, NY, USA, 2000. ACM.

[67] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *ACM WEBKDD WORKSHOP*, 2000.

[68] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.

[69] J. Ben Schafer, Joseph A. Konstan, and John Riedl. E-commerce recommendation applications. *Data Min. Knowl. Discov.*, 5:115–153, January 2001.

[70] Guy Shani, David Heckerman, and Ronen I. Brafman. An mdp-based recommender system. *J. Mach. Learn. Res.*, 6:1265–1295, 2005.

[71] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". pages 210–217. ACM Press, 1995.

[72] Erez Shmueli, Amit Kagian, Yehuda Koren, and Ronny Lempel. Care to comment?: recommendations for commenting on news stories. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 429–438, New York, NY, USA, 2012. ACM.

[73] Milad Shokouhi and Kira Radinsky. Time-sensitive query auto-completion. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 601–610, New York, NY, USA, 2012. ACM.

[74] Luo Si and Rong Jin. Flexible mixture model for collaborative filtering. In *Proc. of ICML*, pages 704–711. AAAI Press, 2003.

[75] Mark D. Smucker and Charles L. A. Clarke. Modeling user variance in time-biased gain. In *Proceedings of the Symposium on Human-Computer Interaction and Information Retrieval*, HCIR '12, pages 3:1–3:10, New York, NY, USA, 2012. ACM.

[76] Sarah K. Tyler and Jaime Teevan. Large scale query log analysis of re-finding. In *Proceedings of the third ACM WSDM'10*, pages 191–200. ACM, 2010.

[77] Sarah K Tyler, Jian Wang, and Yi Zhang. Utilizing re-finding for personalized information retrieval. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1469–1472. ACM, 2010.

[78] Lyle Ungar, Dean Foster, Ellen Andre, Star Wars, Fred Star Wars, Dean Star Wars, and Jason Hiver Whispers. Clustering methods for collaborative filtering. AAAI Press, 1998.

[79] Hirofumi Uzawa. Production functions with constant elasticities of substituion. *Review of Economic Studies*, 29(4):291–299, Oct 1962.

[80] Jian Wang and Brian D Davison. Explorations in tag suggestion and query expansion. In *Proceeding of the 2008 ACM workshop on Search in social media*, pages 43–50. ACM, 2008.

[81] Jian Wang and Brian D Davison. Counting ancestors to estimate authority. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 658–659. ACM, 2009.

[82] Jian Wang, Liangjie Hong, and Brian D Davison. Rsdc09: Tag recommendation using keywords and association rules. *ECML PKDD Discovery Challenge 2009 (DC09)*, page 261, 2010.

[83] Jian Wang, Badrul Sarwar, and Neel Sundaresan. Utilizing related products for post-purchase recommendation in e-commerce. In *Proceedings of the fifth ACM RecSys'11*, pages 329–332. ACM, 2011.

[84] Jian Wang and Yi Zhang. Utilizing marginal net utility for recommendation in e-commerce. In *Proceedings of the 34th ACM SIGIR'11*, pages 1003–1012. ACM, 2011.

[85] Jian Wang and Yi Zhang. Opportunity model for e-commerce recommendation: right product, right time. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in Information.* ACM, 2013.

[86] Jian Wang, Yi Zhang, and Tao Chen. Unified recommendation and search in e-commerce. In *Information Retrieval Technology*, pages 296–305. Springer Berlin Heidelberg, 2012.

[87] Jian Wang, Yi Zhang, Chrisitian Posse, and Anmol Bhasin. Is it time for a career switch. In *Proceeding of the 23rd International World-Wide Web Conference*. ACM, 2013.

[88] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, USA, 2006. ACM.

[89] L. J. Wei. The accelerated failure time model: A useful alternative to the cox regression model in survival analysis. *Statistics in Medicine*, 11(14-15):1871–1879, 1992.

[90] Ryen W. White, Peter Bailey, and Liwei Chen. Predicting user interests from contextual information. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 363–370, New York, NY, USA, 2009. ACM.

[91] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD KDD'10*, pages 723–732, New York, NY, USA, 2010. ACM.

[92] Qianli Xing, Yi Zhang, and Lanbo Zhang. On bias problem in relevance feedback. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1965–1968. ACM, 2011.

[93] Kai Yu, Shenghuo Zhu, John Lafferty, and Yihong Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 211–218, New York, NY, USA, 2009. ACM.

[94] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR'01*, pages 334–342. ACM, 2001.

[95] Dell Zhang, Jinsong Lu, Robert Mao, and Jian-Yun Nie. Time-sensitive language modelling for online term recurrence prediction. In *Proceedings of the 2nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory*, ICTIR '09, pages 128–138, Berlin, Heidelberg, 2009. Springer-Verlag.

[96] Lanbo Zhang and Yi Zhang. Discriminative factored prior models for personalized content-based recommendation. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1569–1572. ACM, 2010.

[97] Lanbo Zhang and Yi Zhang. Interactive retrieval based on faceted feedback. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 363–370. ACM, 2010.

[98] Lanbo Zhang, Yi Zhang, and Yunfei Chen. Summarizing highly structured documents for effective search interaction. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 145–154. ACM, 2012.

[99] Lanbo Zhang, Yi Zhang, Jadiel de Arma, and Kai Yu. Ucsc at relevance feedback track. Technical report, DTIC Document, 2009.

[100] Lanbo Zhang, Yi Zhang, and Qianli Xing. Filtering semi-structured documents based on faceted feedback. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 645–654. ACM, 2011.

[101] Gang Zhao, Mong Li Lee, Wynne Hsu, and Wei Chen. Increasing temporal diversity with purchase intervals. In *Proceedings of the 35th ACM SIGIR*, pages 165–174, New York, NY, USA, 2012. ACM.