MATRIX AND TENSOR APPROXIMATION FOR INTERNET LATENCY PREDICTION AND ONLINE PURCHASE PREDICTION

by

Bang Liu

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Matrix and tensor approximation have gradually become prominent techniques for data mining. They are currently used for a multitude of applications in different subject areas. There are many real-world applications that need to handle sparse data. Given sparse observable or measured data, how to recover or predict the missing entries is critical for these applications.

In this thesis, I study the application of matrix and tensor approximation techniques to Internet latency prediction and purchase prediction in e-commerce. Pairwise latencies in a network are usually estimated from sampled partial measurements. Traditional approaches based on Euclidean embedding suffer from the limitation of the Euclidean assumptions such as triangle inequality and symmetric latencies. I propose a new scheme that can decompose an incomplete latency matrix into a distance component and a network feature component, and apply low-rank matrix completion to complete the network feature matrix based on the fact that network conditions are correlated. In the second problem, I further propose to use tensor approximation for purchase prediction in e-commerce. I analyze the user behavior records collected from Alibaba group's mobile B2C platform and note that different types of historical actions all play an important role in recommending the next item for purchase. I propose a novel model, named Multifaceted Factorizing Personalized Markov Chains (Multifaceted-FPMC), to jointly factorize the tensor of purchase records into hidden factors for various context information, such as user, item, different types of historical actions, etc. Extensive evaluations based on real-world datasets show that our proposed approaches outperform traditional approaches in both problems.

Acknowledgments

I would like to thank all the people who contributed in some way to the work described in this thesis. First and foremost, I am deeply indebted to my academic advisors, Dr. Di Niu and Dr. Vicky Zhao. Their guidance and supervision help me to learn how to be a good researcher and pursue an academic career. Their willingness to discussion helped me through two important years of my life. Additionally, I would like to thank my committee members Dr. Hao Liang and Dr. Linglong Kong for their interest in my work.

I would also like to thank Dr. Zongpeng Li. He is one of the cooperator in my first research work on network latency prediction. Thanks for the discussions and advices during the project, and I learned a lot through this research work. Dr. Linglong Kong supervised me on the second project about online purchase prediction. I greatly benefited from his keen scientific insight and creativity.

Finally, I would like to acknowledge friends and family who supported me during my time here. First and foremost I would like to thank Mom, Dad and other family members for their constant love and support. Additionally, I am lucky to have met lots of good friends here and have a happy life. Thanks for their friendship and unyielding support!

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Acronyms	**Definition**
ALS	Alternative Least Square
BSFM	Bayesian Sparse Factorization Machines
CF	Collaborative Filtering
FM	Factorizing Machines
FPMC	Factorizing Personalized Markov Chains
MC	Matrix Completion
MCMC	Markov chain Monte Carlo
MF	Matrix Factorization
NCS	Network Coordinate System
NMF	Non-negative Matrix Factorization
PD	Penalty Decomposition
RE	Relative Error
RTT	Round Trip Time
RVP	Relative Varying Percentage
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TIV	Triangle Inequality Violations

Chapter 1

Introduction

1.1 Matrix and Tensor Approximation

Matrix and tensor approximation techniques are being used in different subject areas. In many applications it can be useful to approximate an incomplete matrix with a low-rank matrix. For example, in the famous Netflix prize problem [1], we are given a very large ratings matrix, whose rows are the customers and columns are the movies, and the entries are the rating that each customer would hypothetically assign to each movie. As not every customer has rented every movie, only a small fraction of this matrix is actually known. However, if one makes the assumption that most customers' rating preference is determined by only a small number of characteristics of the movie (e.g. genre, lead actor/actresses, director, etc.), then the matrix should be approximately low rank. Matrix factorization characterizes both customers and movies by vectors of latent factors inferred from movie rating patterns [2]. After learned these vectors, we can estimate a customer's rating to a movie by the inner product of the two corresponding latent vectors. The concept of matrix approximation naturally extends to tensor approximation. For high dimensional data, researchers further propose to utilize tensor approximation techniques to discover the latent factors beneath the data [3]–[5]. For example, in the problem of next-basket recommendation [4], we are aiming to predict the items in users' next shopping basket given his/her previous shopping basket items. In this problem, a special case of Tucker Decomposition is utilized to complete a 3D transition cube, in which each element represents the probability a user will purchase an item given he/she already bought another item in the previous shopping basket [4]. In this thesis, we consider two problems: mobile network latency prediction, and online purchase prediction. Based on the analysis to real-world datasets, we propose new models and algorithms based on matrix and tensor approximation techniques.

1.2 Network Latency Prediction

Recent years have witnessed a dramatic growth of Internet traffic of personal devices, among which a large portion comes from mobile devices such as smartphones and tablets [6]. Due to the increasing popularity of interactive applications including live video chat (e.g., FaceTime, Skype, Google+) and gaming, understanding the latencies between personal devices has become essential to the operation of such real-time and delay-sensitive applications. A common idea to estimate end-to-end Internet latencies in a large network is to measure RTTs for only a subset of all pairs, based on which the missing latencies of other pairs are recovered. Existing solutions to such an estimation problem either relies on network embedding (e.g., Vivaldi [7], GNP [8]), which maps nodes into a space, so that their distances in the space predict their latencies, or applies matrix factorization [9] assuming the latency matrix has a certain low rank.

However, the unique characteristics of personal devices have posed great challenges to latency estimation. *First*, almost all existing approaches perform static network latency prediction, based on one incomplete matrix formed by current, mean or median RTTs, assuming the latencies are stable or unchanged, while in reality, latencies between personal devices could vary dramatically over time due to changing network connectivities. In other words, the prediction based on such 2D sampling fails to utilize the significantly useful structures inherent in the 3D data of delay matrices evolved over time. *Second*, network embedding algorithms such as Vivaldi [7], [10] often attempt to find the network coordinates of nodes in a

Euclidean space. However, it is a widespread belief [9], [11], [12] that the triangle inequality may not hold for latencies among end users at the edge of the Internet. *Third*, matrix factorization schemes [9] assume a certain rank of the delay matrices to decide the dimensions of the factors. However, in reality, ranks of delay matrices of personal devices are either hard to know or unstable.

In this thesis, we conduct an in-depth analysis of latency measurements collected from *Seattle* [13], an educational and research platform of open cloud computing and peer-to-peer computing. Seattle consist of laptops, servers, and phones, donated by users and institutions. Compared with another dataset we collected from the PlanetLab, we observe that the latencies between personal devices present different properties in latency distribution as well as time-varying characteristics.

Based on measurements from Seattle, we propose novel methods for both static and dynamic latency estimation problems. *First*, we propose the so-called "Distance-Feature (D-F) Decomposition" method which can decompose a given incomplete latency matrix into a distance matrix that models the impact of geographical distances on propagation delays, and a low-rank network feature matrix that models correlated network conditions among nodes. We propose an iterative learning process using Euclidean embedding and the Penalty Decomposition (PD) method for matrix completion as subroutines. The proposed decomposition avoids the shortcomings of both Euclidean embedding and matrix completion, while exploiting both of their strengths, since the symmetry and triangle inequality do not have to hold for network features, while the low-rank assumption is not imposed on distances.

More importantly, to predict changing latencies, we propose a novel dynamic recovery process to estimate the current missing latencies based on "frames" of incomplete latency matrices sampled in the past. By jointly applying different matrix transformation schemes, we convert the collected incomplete 3D data into structured 2D matrices, and extend the proposed D-F decomposition to apply to the transformed matrices, leveraging the inherent structures both within each frame and across different frames.

We conduct extensive trace-driven simulations based on a large number of RTT measurements collected from both Seattle and PlanetLab, and show that the D-F decomposition significantly outperforms state-of-the-art latency estimation techniques, including matrix factorization and Vivaldi with a high dimension, especially for the Seattle data. The dynamic recovery based on 3D sampling can further substantially enhance the prediction accuracy of changing latencies between personal devices.

1.3 Online Purchase Prediction

Online B2C purchase platforms (e.g., Alibaba group, Amazon, eBay) can now easily track and monitor customers' past actions other than basic demographic information. While the amount of data collected is staggering, effective approaches to glean value from diverse information are still limited. One specific problem is how to predict customers' next-day-purchases given their historical records of different actions that include *click*, *collect*, *add-to-cart* and *payment*. It is highly valuable and promising if we are able to uncover the deep meaning of historical records data and recommend appropriate commodities for users at the right time.

Traditional approaches for recommendation include content filtering [14], collaborative filtering (CF) [15], matrix factorization (MF) [2], etc. However, our unique problem pose great challenge to these methods. First, with only the users' historical actions records available, measuring the similarities between users and items, a key step for both content filtering and collaborative filtering, is challenging. Second, users are more prefer to buy items different with what they have bought. For example, a person who has already got an iPhone is unlikely to buy another smartphone. However, based on what users have already bought, CF and MF usually tend to recommend similar items to users. Third, the main information contained in users' historical action records is their action sequences and the time of each action, rather than any kind of explicit ratings. New approaches are needed to discover the underlying information from such kinds of historical action records data.

In this thesis, we conduct an in-depth analysis of the users' historical action records provided by the Alibaba group, the world's largest e-commerce company that is currently using big data to understand client behaviors and characteristics and offer responsive financial services. Based on the analysis, we propose multiple models and a unified framework to utilize users' historical action records and predict their next-day-purchases. The main contributions of this work can be summarized as follows:

- We analysis the users' historical action records dataset from Alibaba group and propose a new model, Multifaceted Factorizing Personalized Markov Chains (Multifaceted-FPMC). It incorporates multiple types of historical user actions as well as the time information of predicted day to estimate the purchase probability on the next day between each user-item pair.

- We further observe that users' historical actions' influence on their future purchase actions decays with the time intervals between historical actions and purchase actions, approximately following a power-law distribution. We argue that this phenomenon plays an important role in users' future behavior prediction. Accordingly, we further propose our Time-decayed Multifaceted Factoring Personalized Markov Chains (Time-decayed Multifaceted-FPMC) model, which incorporates the temporal influence decay phenomenon for user-item pairs' next-day-purchase probability estimation.

- Finally, we propose a unified framework, Bayesian Sparse Factorization Machine (BSFM), that subsumes our new models and learn model parameters by Gibbs sampling.

- We conduct extensive evaluations based on the Alibaba dataset, and show that our proposed approaches significantly outperform other state-of-the-art prediction algorithms.

1.4 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces the problem of mobile network latency prediction, and our proposed models and algorithms. Chapter 3 describes the problem of online purchase prediction, and introduces our proposed models and the corresponding algorithm for learning the parameters of models. We summarize our work and discuss future works in chapter 4.

Chapter 2

Network Latency Prediction for Personal Devices

2.1 Introduction

Latency prediction between personal devices including mobile devices becomes an important problem due to an increasing popularity of real-time applications. Traditional approaches recover all-pair latencies in a network from sampled measurements using either Euclidean embedding or matrix factorization. However, these approaches targeting static or mean network latency prediction are insufficient to predict personal device latencies, due to unstable and time-varying network conditions, triangle inequality violation and unknown rank of latency matrices. By analyzing latency measurements from the *Seattle* platform, we argue that either Euclidean embedding or matrix factorization is sufficient to modelling the network latencies between network nodes, and further propose new methods for both static latency estimation as well as the dynamic estimation problem given 3D latency matrices sampled over time. In our model, network latencies are explained by two parts: a distance component and a network feature component. We then propose a distance-feature decomposition algorithm that learns the two components in an iterative manner. What is more, by leveraging the structured pattern inherent in the 3D sampled data, we further increase the estimation accuracy of network latencies. Extensive evaluations driven by real-world traces show that our proposed approaches significantly outperform various state-of-the-art latency prediction techniques.

The remainder of this chapter is organized as follows. Sec. 2.2 reviews the related literature, followed by a comparative study of latency measurements from both Seattle and PlanetLab in Sec. 2.3. We propose the distance-feature decomposition method for latency recovery in Sec. 2.4 and study its performance through trace-driven simulations as compared to state-of-art algorithms. In Sec. 2.5, we propose our dynamic latency estimation scheme based on the 3D data of latency matrices evolved over time and again conduct extensive simulations to evaluate its performance. This chapter is concluded in Sec. 2.6.

2.2 Relationship to Prior Work

Network coordinate systems (NCSs) embed hosts into a coordinate space such as Euclidean space, and predict latencies by the coordinate distances between hosts [16]. In this way, explicit measurements are not required to predict latencies. Most of the existing NCSs, such as Vivaldi [7], GNP [8], rely on the Euclidean embedding model. However, such systems suffer a common drawback that the predicted distances among every three hosts have to satisfy the triangle inequality, which does not always hold in practice. Many studies [10], [17] have reported the wide existence of triangle inequality violations (TIV) on the Internet.

To overcome the TIV problem, matrix factorization is introduced in [18] and has recently drawn an increasing attention in the networking community [9], [19]. The key idea is to assume a network distance matrix is low-rank and complete it by factorizing it into two smaller matrices using methods such as Singular Value Decomposition (SVD) or Non-negative Matrix Factorization (NMF) [20]. The estimated distances via matrix factorization do not have to satisfy the triangle inequality. However, these systems actually do not outperform Euclidean embedding models significantly, due to reported problems such as prediction error propagation [11]. Besides, without considering the geographical distances between hosts that dictate propagation delays, they have missed a major chunk of useful information.

Beyond matrix factorization, the general matrix completion problem, including minimizing the rank of an incomplete matrix subject to limited deviation from known entries [21] and minimizing the deviation from known entries subject to a fixed rank [22], has also been widely studied recently for numerous applications in control, image recovery and data mining. Besides, measurement studies have been conducted for different kinds of networks, such as WiFi networks [23], Cellular networks [24], and 4G LTE networks [25], reporting the latencies and other properties. The latency measurement on Seattle is cross-network in nature, as Seattle involves many different types of nodes from servers to laptops and smartphones.

2.3 *Seattle* vs. *PlanetLab*: Measuring the Latencies

In this section, we characterize the latencies between personal devices according to the measurements we have collected from *Seattle* [13]. *Seattle* is a new open peer-to-peer computing platform that provides access to personal computers worldwide. In contrast to PlanetLab [26], which is a global research network comprised of computers mostly located in stable university networks, the Seattle nodes include many personal devices, such as mobile phones, laptops, and desktop computers, donated by users and institutions. Due to the diversity, mobility and instability of these personal devices, there is significant difference between Seattle and PlanetLab in terms of latency measurements.

We have collected the round trip times (RTTs) between 99 nodes in the Seattle network in a 3-hour period commencing at 9 pm on a day in summer 2014. The measurement has resulted in 688 latency matrices containing $6,743,088$ latencies, each of which has a size of 99×99 and represents the pairwise RTTs between 99 nodes collected in a 15.7-second timeframe. In the sequence, we may refer to each matrix as a “frame” in such 3D measurement data. Our data collection on Seattle was limited to 99 nodes because as a new platform that includes both personal computers and servers, Seattle is yet to receive more donations of personal devices.

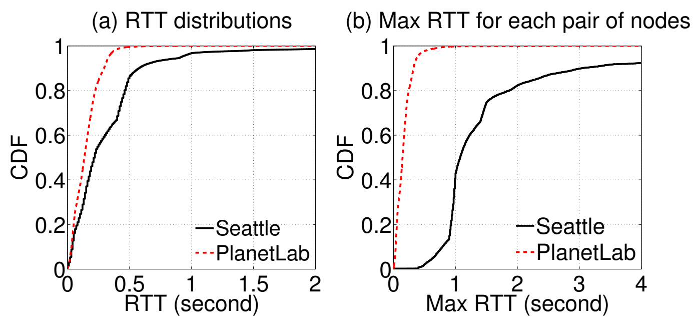


Fig. 2.1. RTT distributions in Seattle and PlanetLab. a) CDFs of all measured RTTs. b) CDFs of the maximum RTT measured for each pair of nodes.

However, it will be clear in Sec. 2.4 and Sec. 2.5 that the collected data is rich enough for the purpose of studying latency prediction algorithms.

As a benchmark dataset, we have also collected the RTTs between 490 PlanetLab nodes in a 9-day period in 2013 and obtained 18 matrices containing $4,321,800$ latencies, each of which has a size of 490×490 and represents the pairwise RTTs collected in a 14.7-hour timeframe. We compare the collected Seattle data and PlanetLab data in terms of inter-node RTTs, rank properties, and time-varying characteristics.

Round Trip Times. Fig. 2.1(a) shows that the Seattle RTTs are greater than those in PlanetLab, with values spread in a wider range. The mean RTT of the two datasets are 0.36 seconds for Seattle and 0.15 seconds for PlanetLab, respectively. While the largest measured RTT in PlanetLab is 7.90 seconds, the maximum RTT measured in Seattle is 90.50 seconds, which maybe because a corresponding node is not online, a frequent case for cellular devices out of the service region. The long tail in Seattle RTTs implies that triangle inequality violation may be prevalent in Seattle.

Rank of Latency Matrices. Fig. 2.2(a) and Fig. 2.2(b) plot the heat maps of a typical frame (one of the 688 latency matrices) in the Seattle data and a typical frame in the PlanetLab data (with white representing large RTT values and black representing small RTTs). We can observe that redundant patterns exist in Fig. 2.2(a) and Fig. 2.2(b) and obtain an intuitive knowledge that the latency matrices in both datasets may be low-rank. We further perform singular value decom-

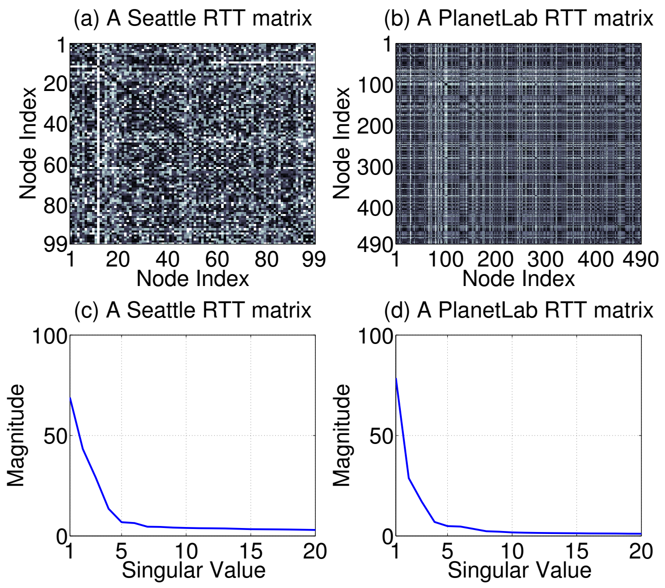


Fig. 2.2. The heat maps and singular values of a Seattle RTT matrix and a PlanetLab RTT matrix.

position (SVD) [27] on both latency matrices, and plot the singular values of both latency matrices in Fig. 2.2(c) and Fig. 2.2(d). We can observe that the singular values of both matrices decrease fast. The 15th singular value of the Seattle latency matrix is 4.9% of its largest one, while the 7th singular value of the PlanetLab latency matrix is 4.7% of its largest one. This confirms the low-rank nature of Internet RTTs reported in previous measurements [28].

Time-Varying Characteristics. Unlike PlanetLab, since Seattle contains personal devices including laptops and mobile phones, the diversity and mobility of these personal devices may greatly affect the stability of latency measurements. Fig. 2.3 plots the RTT measurements evolved over time for 3 typical pairs of nodes in Seattle and PlanetLab, respectively. In contrast to the latencies in PlanetLab which almost remain unchanged over 9 days, the 3 pairs of nodes in Seattle have latencies that vary frequently even in only 30 minutes. The average standard deviation of the latencies between each pair of nodes is 0.36s in Seattle and 0.01s in PlanetLab.

To get a further idea about the evolution of the entire frame of data over time, we denote $M(t)$ the $n \times n$ matrix of RTTs measured at time t, where $M_{ij}(t)$ rep-

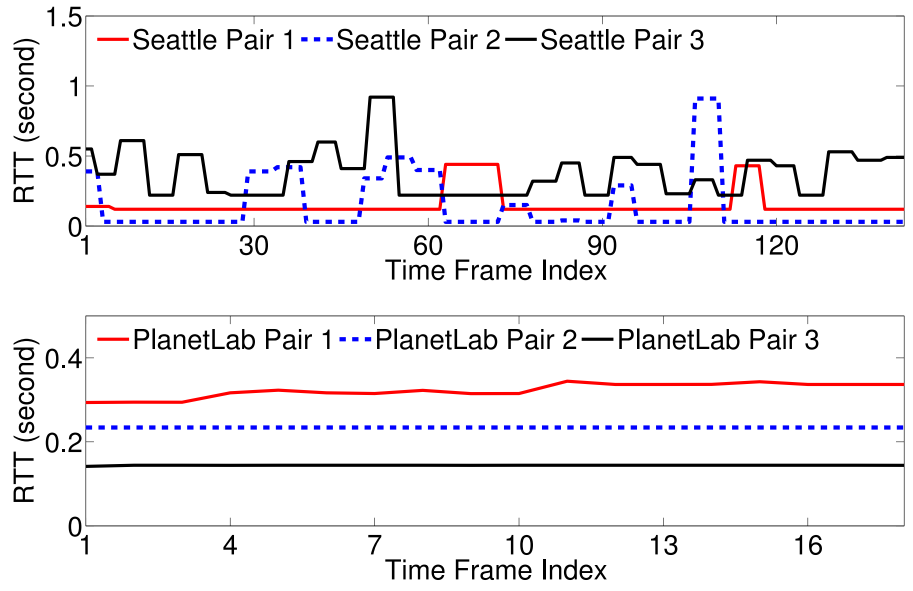


Fig. 2.3. The time-varying latencies of 3 pairs of nodes in Seattle and PlanetLab, respectively.

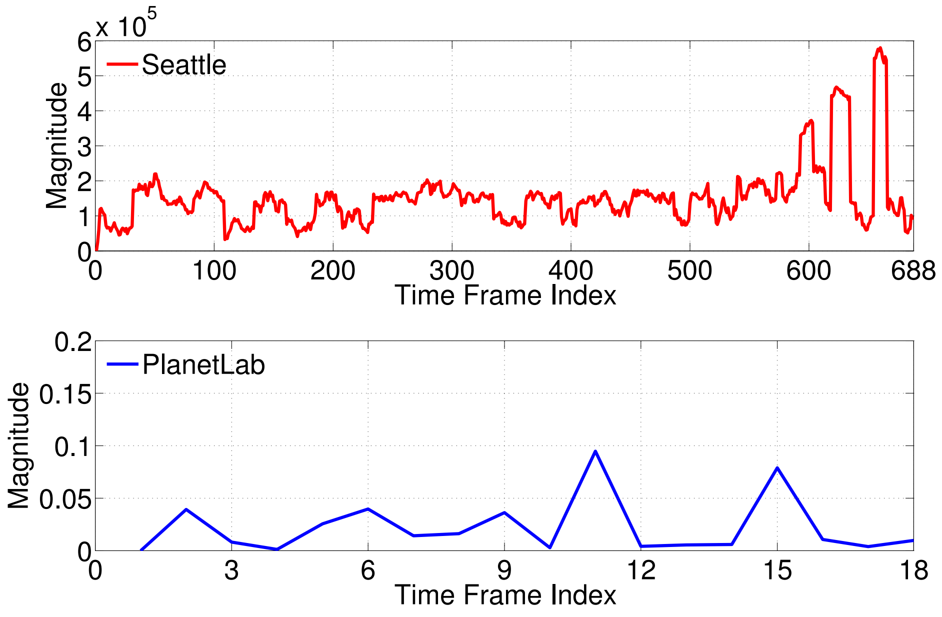


Fig. 2.4. The relative varying percentage of each measured latency matrix relative to the first latency matrix in Seattle and PlanetLab, respectively.

resents the RTT between node i and node j. Then, we define the Relative Varying Percentage (RVP) of $M(t)$ relative to the first matrix $M(1)$ as

$$\text{RVP}(t,1) = \tfrac{1}{n^2-n}\textstyle\sum_{i,j=1,i\neq j}^{n}[M_{ij}(t) - M_{ij}(1)]/M_{ij}(1).$$

We compare the RVPs of the Seattle RTTs over time with those of the PlanetLab RTTs, by plotting the RVP of every frame at time t relative to the first frame of data in Fig. 2.4, which shows a huge difference between the two datasets. While the largest RVP of the PlanetLab frames over 9 days is only 0.09, the RVPs of the Seattle frames measured for 3 hours vary from 0 to 5.8×10^5 with a mean of 1.5×10^5. This demonstrates the time-varying nature of Seattle latencies, which makes it hard to predict the latency between two Seattle nodes. Traditional network coordinate embedding is not suitable to model the latencies in personal device networks. For example, if a Seattle node is a cellphone, whenever the phone user moves, its coordinate will change greatly according to the changes in surrounding network environments.

2.4 Static Latency Estimation via Distance-Feature Decomposition

In this section, we propose a new algorithm for the static network latency recovery problem, given a frame of latency matrix with missing values. Our new algorithm exploits both the underlying geographical distances and the low-rank structure of the RTT matrix at hand. Traditional Euclidean embedding [7], [8] assumes symmetry and triangle inequalities for pairwise latencies, which may not be true in reality, especially for mobile devices with poor connectivity. On the other hand, matrix factorization approaches [9] rely on the assumption of a fixed low rank in the latency matrix. However, it is hard to know such a rank *a priori*. And this method may ignore the true Euclidean component in latencies dictated by geographical distances.

Our algorithm combines the strengths of both methods by modeling the pairwise latencies with two components: a *distance component*, representing geographic information that dictates propagation delay, and a *network feature component*, representing correlated network connectivity. The essence of our algorithm is a learning process that iteratively decomposes both components.

2.4.1 The Static Network Latency Prediction Problem

Let $\mathbb{R}^n$ denote the n-dimensional Euclidean space. The set of all $m \times n$ matrices is denoted by $\mathbb{R}^{m\times n}$. Assume a network contains n nodes, and the latency matrix measured between these nodes is $M \in \mathbb{R}^{n\times n}$, with M_{ij} representing the RTT between node i and node j. We use Θ to denote the set of index pairs $(i,\ j)$ where the measurements M_{ij} are missing. For missing entries $(i,\ j) \in \Theta$, we denote their values as $M_{ij} = \mathsf{unknown}$. We define the sample rate R as the percentage of known entries in M.

The static prediction problem is—given an RTT matrix M with missing entries, recover the values of the missing entries. We let $\hat{M} \in \mathbb{R}^{n\times n}$ denote the recovered RTT matrix.

2.4.2 Iterative Distance-Feature Decomposition

We model the RTT matrix M as the *Hadamard product* (or entry-wise product) of a symmetric distance matrix $D \in \mathbb{R}^{n\times n}$ and an asymmetric network feature matrix $F \in \mathbb{R}^{n\times n}$, i.e.,

$$M = D \circ F, \tag{2.1}$$

where $M_{ij} = D_{ij}F_{ij},\ 1 \le i,\ j \le n$, D_{ij} represents the distance between nodes i and j in a Euclidean space, and F_{ij} represents the "network connectivity" from node i to node j: a smaller F_{ij} indicates a better connectivity. The rationale is that while the geographical distance of two nodes on the earth dictates the propagation delay between them, other factors such as network congestions and node status can also affect the RTT values.

We assume that *only* the network feature matrix F is low-rank. This is because

Algorithm 2.1 Iterative Distance-Feature Decomposition

```
1: $D^0 := M$
2: for $k = 1$ to maxIter do
3:     Perform Euclidean Embedding on $D^{k-1}$ to get the complete matrix of distance estimates $\hat{D}^k$
4:     $F^k_{ij} := \begin{cases} \frac{M_{ij}}{\hat{D}^k_{ij}} & \forall(i,\, j) \notin \Theta \\ \text{unknown} & \forall(i,\, j) \in \Theta \end{cases}$
5:     Perform Matrix Completion (2.4) on $F^k$ to get the complete matrix of network feature estimates $\hat{F}^k$
6:     $D^k_{ij} := \begin{cases} \frac{M_{ij}}{\hat{F}^k_{ij}} & \forall(i,\, j) \notin \Theta \\ \text{unknown} & \forall(i,\, j) \in \Theta \end{cases}$
7: $\hat{M}_{ij} := \hat{D}^{\mathsf{maxIter}}_{ij} \hat{F}^{\mathsf{maxIter}}_{ij},\ 1 \le i,\, j \le n$
```

there exists correlation between network connectivities on all incoming (or outgoing) links of each node. Another interpretation is through feature vectors. If the rank of F is r, F can be represented by

$$F = F_l^{\mathsf{T}} F_r,\ F_l \in \mathbb{R}^{r \times n},\ F_r \in \mathbb{R}^{r \times n}. \tag{2.2}$$

We call the i^{th} column of F_l, denoted by f_l^i, the *left feature vector* of node i, which represents the network feature from node i to other nodes. Similarly, we call the i^{th} column of F_r, denoted by f_r^i, the *right feature vector* of node i, which represents the network feature from other nodes to node i. Hence, the network connectivity from node i to node j can be determined by the feature vectors, i.e., $F_{ij} = {f_l^i}^{\mathsf{T}} f_r^j$.

Our model overcomes the weaknesses of both Euclidean embedding and low-rank matrix completion, since symmetry and triangle inequalities only need to hold for the distance matrix D but not F, and the low-rank property is only assumed for network connectivity F.

To learn both the distance matrix D and network feature matrix F from a latency matrix M with missing entries, we propose an iterative algorithm, described in Algorithm 2.1, that incorporates both Euclidean embedding and low-rank matrix completion as subroutines. Denote the estimated matrix D and F at iteration k as $\hat{D}^k$ and $\hat{F}^k$. First, we initialize D^0 to be the original latency matrix M with missing entries. In each iteration, we estimate the distance matrix $\hat{D}^k$ with Euclidean embedding. We then obtain the remaining ratio matrix between M and $\hat{D}^k$, which is the incomplete network feature matrix F^k. By applying low-rank matrix completion on F^k, we get the estimated complete network feature matrix $\hat{F}^k$ at iteration k. We then divide M by $\hat{F}^k$ to get D^k, which is the input for Euclidean embedding in the next iteration, and so on. After a few iterations, $\hat{D}$ and $\hat{F}$ will approach the real geographical distance component and the network factor, respectively. Finally, the predicted latency between nodes i and j is given by (2.1).

The two critical subroutines in our algorithm are Euclidean embedding on D^k and low-rank matrix completion on F^k. There are various algorithms available for these two tasks. We apply the Vivaldi algorithm [7] for Euclidean embedding, and the Penalty Decomposition (PD) method [22] for low-rank matrix completion.

2.4.2.1 Euclidean Embedding

Given the input matrix $M \in \mathbb{R}^{n\times n}$, Vivaldi predicts network latencies by assigning every node a coordinate and estimating the latency between two nodes by their Euclidean distance, i.e., the estimated latency $\hat{M}_{ij}$ between nodes i and j is given by

$$\hat{M}_{ij} = \|x_i - x_j\|, \tag{2.3}$$

where x_i is a d-dimensional coordinate vector assigned to node i.

2.4.2.2 Low-Rank Matrix Completion

Given an input matrix $X \in \mathbb{R}^{m\times n}$ with missing entries, the problem of low-rank matrix completion is to find a complete matrix $\hat{X}$ by solving

$$\begin{aligned} \underset{\hat{X}\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad & \text{rank}(\hat{X}) \\ \text{subject to} \quad & |\hat{X}_{ij} - X_{ij}| \le \tau,\ (i,\ j) \notin \Theta, \end{aligned} \tag{2.4}$$

where τ is a parameter to control the error tolerance on known entries of X [22].

We utilize the Penalty Decomposition (PD) method [22] to solve the low-rank matrix completion problem in Algorithm 2.1. The PD method can solve general rank minimization problems like the following:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f(X) + \nu \,\text{rank}(X) \\ \text{subject to} \quad & g(X) \leq 0,\ h(X) = 0,\ X \in \Phi \cap \Psi, \end{aligned} \tag{2.5}$$

for $\nu \geqslant 0$, where Φ is a closed convex set and Ψ is a closed unitarily invariant convex set in $\mathbb{R}^{m\times n}$, and $f : \mathbb{R}^{m\times n} \to \mathbb{R},\ g : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ and $h : \mathbb{R}^{m\times n} \to \mathbb{R}^q$ are continuously differentiable functions.

The PD method solves problem (2.5) by reformulating it as

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f(X) + \nu \,\text{rank}(Y) \\ \text{subject to} \quad & g(X) \leq 0,\ h(X) = 0,\ X \in \Phi,\ Y \in \Psi, \end{aligned} \tag{2.6}$$

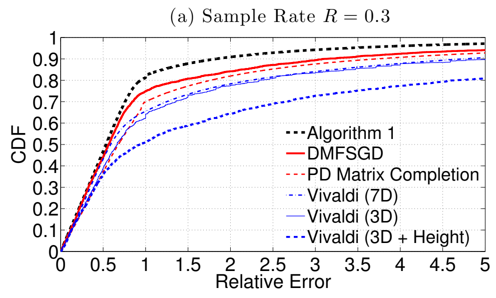
and defining a corresponding quadratic penalty function as

$$\begin{aligned} P_\varrho(X, Y) = {} & f(X) + \nu \,\text{rank}(Y) \\ & + \frac{\varrho}{2}(\|[g(X)]^+\|_2^2 + \|h(X)\|_2^2 + \|X - Y\|_F^2), \end{aligned} \tag{2.7}$$

where $\varrho > 0$ is a penalty parameter, $[\cdot]^+$ denotes the nonnegative part of a vector that $x^+ = \max(x, 0)$ given a vector $x \in \mathbb{R}^n$, and $\|\cdot\|_F$ is the Frobenius norm of a real matrix $X \in \mathbb{R}^{n\times n}$, i.e., $\|X\|_F = \text{Tr}(XY^T)$, with $\text{Tr}(\cdot)$ denoting the trace of a matrix. Then the PD method minimizes (2.7) by alternately solving two subproblems: minimizing over X with Y fixed and minimizing over Y with X fixed, each of which can be approximately solved by a block coordinate descent (BCD) method, which is widely used to solve large-scale optimization problems [29].

It is easy to see that problem (2.4) is a special case of problem (2.5) with $f(X) \equiv 0,\ p = q = 0,\ \nu = 1,\ \Psi = \mathbb{R}^{m\times n}$ and

$$\Phi = \{X \in \mathbb{R}^{m\times n} :\ |X_{ij} - M_{ij}| \leq \tau,\ (i, j) \in \Theta\}. \tag{2.8}$$



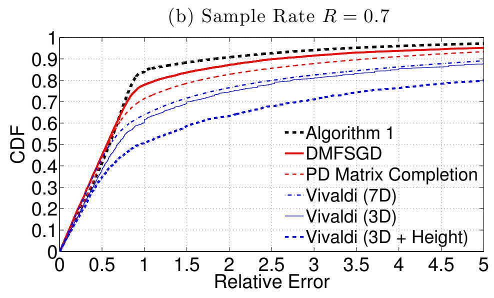
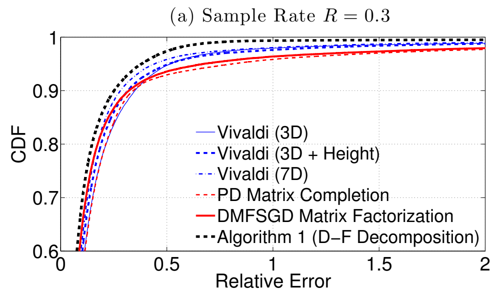


Fig. 2.5. The CDFs of relative estimation errors on missing values for the Seattle dataset, under sample rate $R = 0.3$ and $R = 0.7$. The legends follow the same order as the curves at relative error $= 1.0$



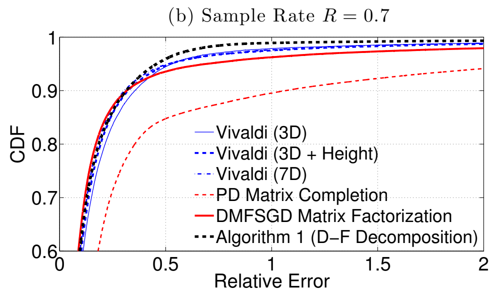


Fig. 2.6. The CDFs of relative estimation errors on missing values for the PlanetLab dataset, under sample rate $R = 0.3$ and $R = 0.7$. The legends follow the same order as the curves at relative error $= 0.5$.

Thus, the two subproblems to be alternately solved are

$$\begin{aligned} &\underset{X}{\text{minimize}} \quad \{\|X - A(Y)\|_F^2 : \; X \in \Phi\}, \\ &\underset{Y}{\text{minimize}} \quad \{\text{rank}(Y) + \varrho\|Y - B(X)\|_F^2 : \; Y \in \mathbb{R}^{m\times n}\} \end{aligned} \tag{2.9}$$

for some $\varrho > 0,\ A,\ B \in \mathbb{R}^{m\times n}$, respectively. Thus, the PD method can be suitably applied to solve (2.4). Please refer to [22] for more details about the PD completion method.

2.4.3 Performance Evaluation

We evaluate our algorithm based on both the Seattle data and PlanetLab data, in comparison with various state-of-the-art approaches. We define the relative estimation error (RE) on missing entries as $|\hat{M}_{ij} - M_{ij}|/M_{ij}$, for $(i,\ j) \in \Theta$, which will be used to evaluate prediction accuracy.

2.4.3.1 Comparison with Other Algorithms

We compare our algorithm with the following approaches:

- **Vivaldi** with dimension $d = 3$, $d = 7$, and $d = 3$ plus a height parameter;
- **PD matrix completion (MC)** directly applied to the latency matrix M;
- **DMFSGD Matrix Factorization** [9] that attempts to approximate M by the product of two smaller matrices $U \in \mathbb{R}^{r\times n}$ and $V \in \mathbb{R}^{r\times n}$, i.e., $\hat{M} = U^T V$, such that a loss function based on $M - \hat{M}$ is minimized, where r is the assumed rank of $\hat{M}$.

For our method, the Euclidean embedding part on D is done using Vivaldi with a low dimension of $d = 3$ without heights.

We randomly choose 100 frames from the 688 frames in the Seattle data. For PlanetLab data, as differences among the 18 frames are small, we randomly choose one frame to test the methods. Recall that the sample rate R is defined as the percentage of known entries. Each chosen frame is independently sampled at a low rate $R = 0.3$ (70% latencies are missing) and at a high rate $R = 0.7$, respectively.

For DMFSGD, we set the rank of $\hat{M}$ to $r = 20$ for Seattle data and $r = 10$ for PlanetLab data, respectively, since the 20^{th} (or 10^{th}) singular value of M is less than 5% of the largest singular value in Seattle (or PlanetLab). In fact, $r = 10$ is adopted by the original DMFSGD work [9] based on PlanetLab data. We have tried other ranks between 10-30 and observed similar performance. We plot the relative estimation errors on missing latencies in Fig. 2.5 and Fig. 2.6, for the Seattle data and PlanetLab data, respectively, under 6 methods.

For the Seattle results in Fig. 2.5, we can see that the D-F decomposition outperforms all other algorithms by a substantial margin. We first compare with Vivaldi. Even if Vivaldi Euclidean embedding is performed in a 7D space, it only improves over 3D space slightly, due to the fundamental limitation of Euclidean assumption.

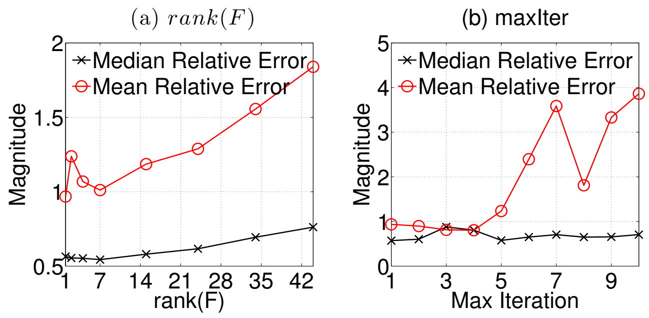


Fig. 2.7. Influence of rank(F) and maxIter for the Seattle dataset.

Furthermore, the 3D Vivaldi with a height parameter, which models the "last-mile latency" to the Internet core [7], is even worse than the 3D Vivaldi without heights in Seattle. This implies that latencies between personal devices are better modeled by their pairwise core distances multiplied by the network conditions, rather than by pairwise core distances plus a "last-mile latency". Thus, we adopt 3D Vivaldi without heights as the Euclidean embedding algorithm in our D-F decomposition.

We now look at the matrix completion algorithms in Fig. 2.5. Both PD matrix completion and DMFSGD are inferior to our algorithm because they solely rely on the low-rank assumption, which may not hold for pairwise core distances. As has been pointed out in Sec. 2.4, ignoring the underlying Euclidean part which does model geographical distances will not yield the best performance, especially for unstable latencies in a mobile network.

For the PlanetLab results in Fig. 2.6, our algorithm is only slightly better than other algorithms, which again implies the much different behavior of Seattle and PlanetLab latencies. The improvement in PlanetLab is not as great as in Seattle, because network conditions in PlanetLab are more stable, which makes the network feature matrix F less useful. This fact again shows the unique strength of our algorithm to cope with unstable personal device networks.

2.4.3.2 Impact of Parameters

We investigate the impact of three parameters to our algorithm: the sample rate R, the rank of network feature matrix rank(F) and the number of iterations maxIter.

Fig. 2.5(a) and Fig. 2.5(b) reveal the robustness of our algorithm at both high ($R = 0.7$) and low ($R = 0.3$) sample rates. We have also tested other sample rates and observed similar results.

We then study the impact of the achieved rank(F) from the PD matrix completion part in our algorithm by tuning τ in (2.4) to indirectly control the produced rank(F). Recall that rank(F) also represents the dimension of node left/right feature vectors. Fig. 2.7(a) shows how the median and mean of relative estimation errors change as rank(F) varies. Our experimental experience suggests that the best results are usually achieved when $1 \leq$ rank$(F) \leq 10$ (the best result is achieved at 7 in this figure).

Finally, Fig. 2.7(b) evaluates the impact of the number of iterations, which shows the best accuracy is often achieved in just the 2^{nd} or 3^{rd} iteration. The performance degrades when more iterations are performed due to the overfitting effect.

2.5 Dynamic Latency Estimation via 3D Sampling

Traditional network latency estimation [7], [9],[18] all attempt to predict static (median/mean) network latencies. However, as shown in Sec. 2.3, the latencies between personal devices may change over time. This motivates us to study the dynamic latency estimation problem to fill the missing entries in the current frame T based on a window of frames measured from $t = 1$ to $t = T$. We call such an approach dynamic latency estimation from "3D sampling", since the network latencies are sampled over time, where each frame of data contains only partial measurements of RTTs.

Although the latency between a pair of Seattle nodes changes frequently, it may stay in a state for a while before hopping to a new state, as shown in Fig. 2.3. Therefore, if we utilize the autocorrelation between frames at different times in addition to the inter-node correlation in the network feature matrix, we may improve the prediction accuracy.

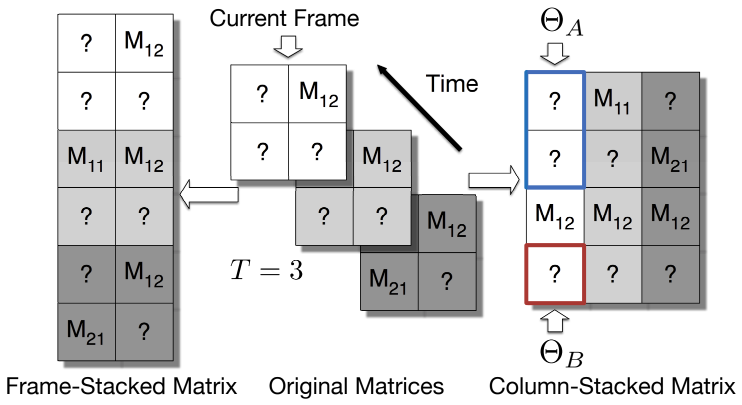


Fig. 2.8. An illustration for frame-stacking and column-stacking operations.

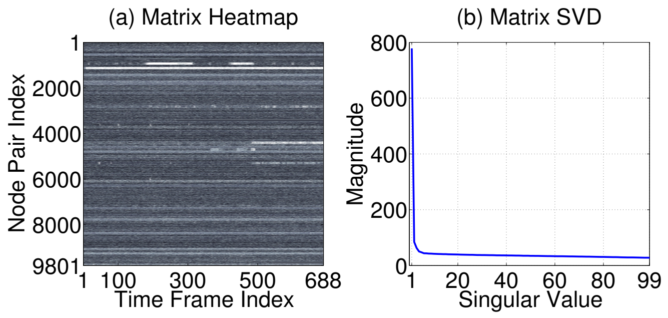


Fig. 2.9. The heat map and singular values of the column-stacked Seattle dataset. The size of the compound matrix is 9801×688, and every column of the matrix contains all the latencies measured in one frame.

Suppose we have measured the latency matrix of n nodes for T time frames, where T is called the prediction window. Denote the incomplete matrix measured at time t by $M(t)$, then our objective is to predict the missing entries in the current latency matrix $M(T)$ based on $M(1), \ldots, M(T)$.

2.5.1 D-F Decomposition from 3D Sampled Data

The main idea of our algorithm is to stack the latency matrices measured at different times in different ways to obtain 2D compound matrices, and then exploit the low-rank nature of the compound matrices. We use two kinds of stack operations: frame-stacking and column-stacking, as illustrated in Fig. 2.8. In column-stacking, every latency matrix $M(t) \in \mathbb{R}^{n\times n}$ is transformed into a column vector $V(t) \in \mathbb{R}^{n^2}$

Algorithm 2.2 D-F Decomposition for 3D Sampled Data

```
1: **Predicting Missing Pairs in $\Theta_A$:**
2: Column-stack $M(t)$ for $1 \leq t \leq T$ to get $\Omega$. Perform matrix completion (2.4) on $\Omega$ to get the complete matrix $\hat{\Omega}$. Unstack $\hat{\Omega}$ to get $\hat{M}_c(t)$ as an estimate of each $M(t)$.
3: **Predicting Missing Pairs in $\Theta_B$:**
4: Initially, let $D^0(t) := M(t)$
5: **for** $k = 1$ to maxIter **do**
6:     Perform Euclidean embedding on $D^{k-1}(t)$ to get the complete estimated matrix $\hat{D}^k(t)$ for each $t$.
7:     $F_{ij}^k(t) := \begin{cases} \frac{M_{ij}(t)}{\hat{D}_{ij}^k(t)} & \forall (i,\ j) \notin \Theta \\ \text{unknown} & \forall (i,\ j) \in \Theta \end{cases}$
8:     Frame-stack $F^k(t)$ for $1 \leq t \leq T$ to get $\mho_F^k$. Perform matrix completion (2.4) on $\mho_F^k$ to get the complete estimated matrix $\hat{\mho}_F^k$. Unstack $\hat{\mho}_F^k$ to get $\hat{F}^k(t) \in \mathbb{R}^{n \times n}$.
9:     $D_{ij}^k(t) := \begin{cases} \frac{M_{ij}(t)}{\hat{F}_{ij}^k(t)} & \forall (i,\ j) \notin \Theta \\ \text{unknown} & \forall (i,\ j) \in \Theta \end{cases}$
10: $\hat{M}_{f,ij}(T) := \hat{D}_{ij}^{\mathsf{maxIter}}(T)\hat{F}_{ij}^{\mathsf{maxIter}}(T),\ 1 \leq i,\ j \leq n.$
11: $\hat{M}_{ij}(T) := \begin{cases} \hat{M}_{c,ij}(T) & \forall (i,\ j) \in \Theta_A \\ \hat{M}_{f,ij}(T) & \forall (i,\ j) \in \Theta_B \\ M_{ij}(T) & \forall (i,\ j) \notin \Theta \end{cases}$
```

containing the latencies of all pairs of nodes. Then, the *column-stacked matrix* $\Omega \in \mathbb{R}^{n^2 \times T}$ consists of vectors $V(t)$ ordered by their measured time. In frame-stacking, we directly concatenate all the measured latency matrices sorted by time to form the *frame-stacked matrix* $\mho \in \mathbb{R}^{nT \times n}$.

Fig. 2.9 shows the heat map and singular values of the column-stacked matrix that consists of 688 frames of the Seattle data. As we can see, the heat map reveals the low-rank nature of the compound matrix: even though the column-stacked matrix has a size of 9801×688, the 22^{nd} singular value of the matrix is only 5% of the largest one. This implies that the latency matrices measured at different times are highly correlated while evolving with time.

Given a prediction window of T frames of latency matrices, we use Θ to denote the set of index pairs (i, j) for which $M_{ij}(T)$ are missing at the current time T. We can further divide Θ into two subsets:

$$\Theta_A = \{(i,j)|M_{ij}(t) \text{ is known for at least one } t \in \{1,\ldots,T-1\}\}$$
$$\Theta_B = \{(i,j)|M_{ij}(t) \text{ is missing for all } t \in \{1,\ldots,T-1\}\}$$

We use two different procedures to recover the missing pairs in Θ_A and Θ_B. For Θ_A, we simply apply PD matrix completion to the *column-stacked* matrix Ω to recover the missing values. The predicted values of $M_{ij}(T)$ are denoted by $\hat{M}_{c,ij}(T)$ for $(i,j) \in \Theta_A$. The intuition is that when some past values $M_{ij}(t)$ are measured, we could directly take advantage of the low-rank property of Ω, i.e., the auto-correlation of measurements across times as shown in Fig. 2.9, to estimate the current missing values.

For Θ_B, we assume each $M(t)$ can be decomposed into a distance matrix $D(t)$ and network feature matrix $F(t)$, and apply a variation of the proposed Algorithm 2.1 (D-F Decomposition) to the *frame-stacked* matrix $\mho$ to recover the missing values. This requires us to iteratively apply Euclidean Embedding for every frame of the distance component $D(t)$ in the prediction window $t \in \{1,\ldots,T\}$, and apply PD matrix completion to the entire frame-stacked matrix $\mho_F$ formed by all the network feature frames $F(t)$ for $t \in \{1,\ldots,T\}$. Therefore, Algorithm 2.1 must be extended to handle the frame-stacking (before PD matrix completion) and the *unstacking* (before Euclidean embedding).

We treat Θ_B differently, since for $(i,j) \in \Theta_B$ where no past value of $M_{ij}(t)$ is measured, the column-stacked matrix is not useful to predict $M_{ij}(T)$, since the entire row in Ω composed of $M_{ij}(t)$ for $1 \leq t \leq T$ is missing. In this case, the completed values of $M_{ij}(T)$ will be meaningless, because the rank of Ω will not change if we scale up or down the entire estimated row. Therefore, we exploit the low-rank nature of $\mho_F$, containing frame-stacked feature matrices $F(t)$ for $1 \leq t \leq T$. Algorithm 2.2 describes the detailed steps of our sparse recovery process based on 3D data.

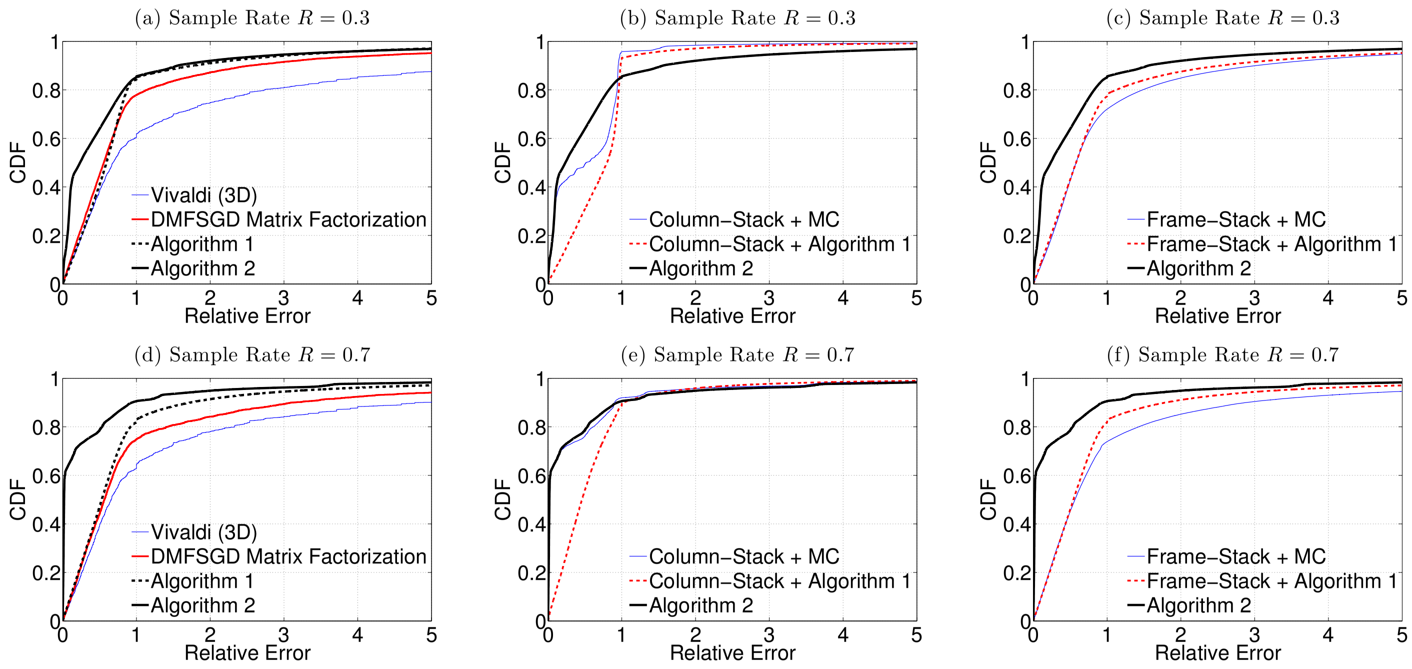


Fig. 2.10. The CDFs of relative estimation errors on the missing values in the current frame with sample rate $R = 0.3$ and $R = 0.7$ for the Seattle dataset.

2.5.2 Performance Evaluation

2.5.2.1 Comparison with Other Algorithms

We test our dynamic prediction algorithm on 50 3D matrices, each randomly selected from the Seattle dataset. Every 3D matrix contains $T = 3$ latency frames. The objective is to recover all the missing values in the last frame. We compare our algorithm with the static prediction methods described in the previous section. Besides, we also compare to four other methods:

- **Column-Stack+MC:** column-stack the latency matrices and perform PD matrix completion on Ω;
- **Column-Stack+Algorithm 1:** column-stack the latency matrices and perform D-F decomposition on Ω;
- **Frame-Stack+MC:** frame-stack the latency matrices and perform PD matrix completion on $\mho$;
- **Frame-Stack+Algorithm 1:** frame-stack the latency matrices and perform D-F decomposition on $\mho$;

Notice that when we perform D-F decomposition on Ω or $\mho$, we perform Euclidean embedding for each unstacked latency frame individually and perform matrix completion on the big stacked network feature matrix in each iteration.

Fig. 2.10(a) and Fig. 2.10(d) compare Algorithm 2.2 with the static prediction algorithms. For both low and high sample rates $R = 0.3$ or $R = 0.7$, Algorithm 2.2 that exploits 3D sampled frames significantly outperforms the static latency prediction methods. It verifies the significant benefit of utilizing historical information, and reveals the strong correlation between different latency frames over timeline. By exploiting the low-rank structure of the column-stacked latency matrix Ω and the frame-stacked network feature matrix $\mho_F$, Algorithm 2.2 takes full advantage of the implicit information in the 3D data.

Fig. 2.10(b) and Fig. 2.10(e) compare Algorithm 2.2 with two other methods based on the column-stack operation: perform PD matrix completion on Ω, and perform D-F decomposition on Ω. Compared with the method that perform PD matrix completion on Ω, our Algorithm 2.2 outperforms it a lot when the sample rate is low ($R = 0.3$). The improvement is due to the different treatment to latencies for node pairs $(i,\ j) \in \Theta_B$ in our algorithm. When the sample rate is high ($R = 0.7$), the difference between Algorithm 2.2 and the method that performs PD matrix completion on Ω is tiny. Because the proportion of node pairs $(i,\ j) \in \Theta_B$ will be small if sample rate is high. For the method that performs D-F decomposition on Ω, it is even worse than performing PD matrix completion directly on Ω. This reveals the fact that we can benefit more from historical values of M_{ij} when they are available rather than using network condition correlations between different nodes for estimation.

Fig. 2.10(c) and Fig. 2.10(f) compare Algorithm 2.2 with two other methods based on the frame-stack operation: performing PD matrix completion on $\mho$, and performing D-F decomposition on $\mho$. As we can see, our algorithm outperforms both of them at both high ($R = 0.7$) and low ($R = 0.3$) sample rates. Furthermore, compared with performing PD matrix completion on $\mho$, the effect of performing D-F decomposition on $\mho$ is better, which again implies that utilizing the low-rank structure of the network feature matrix is more reasonable than utilizing the low-rank property of the original latency matrices.

Through all the comparisons above, we show the benefits of incorporating historical latency frames and prove the necessity of different treatments to unknown node pairs $(i,\ j) \in \Theta_A$ and $(i,\ j) \in \Theta_B$, i.e., the column-stack operation is suitable for node pairs $(i,\ j) \in \Theta_A$ and the frame-stack operation is better for node pairs $(i,\ j) \in \Theta_B$. It is shown that the combined strategy in our hybrid Algorithm 2.2 is optimal.

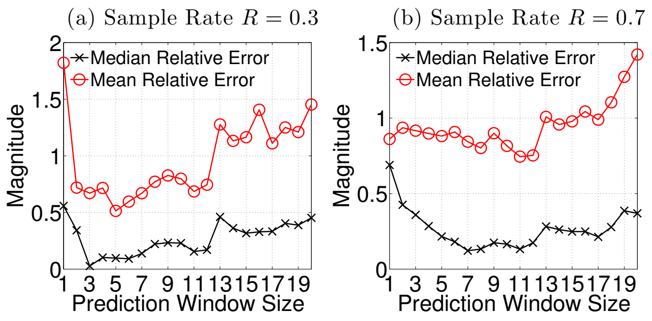


Fig. 2.11. Influence of the prediction window T for the Seattle dataset.

2.5.2.2 Impact of Prediction Window T

Fig. 2.11 shows how the median and mean relative estimation errors for missing values in frame T vary when the prediction window T increases. We make two interesting observation. *First*, the best performance is achieved by $T = 3$ when the sample rate is 0.7, but is achieved by $T = 7$ when the sample rate is 0.3. *Second*, the prediction errors increase if we add more frames. When the sample rate R is high, a few recent frames are enough to predict the current missing latencies. However, when R is low, the latency between each pair of nodes is less frequently measured, and thus more historical frames are needed to recover the current latencies. However, once we have obtained enough information from some historical frames, adding more frames will hurt performance, since the rank of the column-stacked matrix will increase, making it harder to complete the stacked matrix with low error.

2.6 Concluding Remarks

Based on measurements collected from the Seattle network that consists of user-donated personal devices, in this chapter, we study the new challenges in estimating the less stable and time-varying latencies in personal device networks. We propose the distance-feature decomposition algorithm that avoids the defects of both Euclidean embedding and matrix factorization. By decomposing the network latency matrix into a distance matrix and a network feature matrix, our approach is able to capture the underlying geographical distance as well as varying network conditions among the nodes. To predict changing latencies, we further formulate the dynamic network latency estimation problem that aims to predict the current missing latencies based on frames of incomplete latency matrices collected in the past, and extend our distance-feature decomposition algorithm for such 3D sampled data, with the aid of a hybrid matrix transformation scheme. Extensive evaluation based on both Seattle and PlanetLab data shows that our algorithms outperform state-of-the-art network embedding algorithms (e.g., high-dimensional Vivaldi with/without heights) and matrix factorization (e.g., DMFSGD) by a substantial margin, especially for personal device networks. The prediction accuracy is further significantly improved by exploiting the structure inherent in the 3D sampled data through the proposed hybrid dynamic estimation mechanism.

Chapter 3

Online Purchase Prediction for E-commerce

3.1 Introduction

Enterprises are now trying to glean intelligence from the staggering data collected and translate that into business advantage. One specific problem is predicting users' future purchases based on users' historical action records and offering personalized recommendations. Traditional approaches for recommendation includes collaborative filtering (CF), matrix factorization (MF) and so on. However, given users' historical action records rather than explicit ratings from users to items, these algorithms show their limitations to fully utilize the information. In this chapter, we analysis the user historical action records dataset from the Alibaba group. To estimate the purchase probabilities between user-item pairs on the next day, we propose a new model named Multifaceted Factoring Personalized Markov Chains (Multifaceted-FPMC) that incorporates various influential context information extracted from users' historical action records. We further observe that users' historical actions' influence on their future purchase actions decays with the time intervals between historical actions and purchase actions, approximately following a power-law distribution. We put a special emphasis on the temporal influence decay phenomenon and argue that it plays an important role in users' future actions. Accordingly, we further propose our Time-decayed Multifaceted Factorizing Personalized Markov Chains (Time-decayed Multifaceted-FPMC) model, which incorporates the temporal influence decay phenomenon for purchase probability estimation. Finally, we propose a unified framework, named Bayesian Sparse Factorization Machines (BSFM), that subsumes our new models and learn model parameters by Markov chain Monte Carlo (MCMC). Extensive evaluations driven by real-world dataset show that our proposed approaches significantly outperform multiple state-of-the-art prediction algorithms.

The remainder of this chapter is organized as follows. Section 3.2 reviews the related literatures and shows their drawbacks in our case. In section 3.4, we conduct in-depth analysis to the Alibaba users' historical action records dataset. From our analysis, we conclude that all types of users' historical actions have influence on users' future purchase actions. Besides, we also notice that users' passion for online shopping varies with different day-of-week. Based on these observations, we propose our Multifaceted-FPMC model for online purchase probability estimation. We then describe the temporal influence decay phenomenon of users historical actions, and propose the Time-decayed Multifaceted-FPMC model in this section. Finally, we propose a unified feature factorization framework BSFM that subsumes our new models, and elaborate its relation and difference with general factorization machines (FM). Furthermore, we describe the Markov chain Monte Carlo learning algorithm for model parameters estimation in section 3.5. In section 3.6, we conduct extensive evaluations on the Alibaba dataset and compare the performance of our proposed approaches with other algorithms. We conclude this chapter in section 3.7.

3.2 Related Work

A lot of researches have been done in recommendation system to provide a list of recommendations that users may be interested to purchase. Content filtering [14], [30] methods create a profile for each user or item to characterize its nature and associate users with matching items. Content-based strategies is easy to express and implement. However, they require gathering external information that might not be available or easy to collect for creating the profiles. Another problem is the content similarity of recommended items. Different with movie or music recommendation, usually a user would not like to buy a similar item, such as a phone, again when they already own one.

Collaborative filtering is the most popular method in recommendation systems [15], [31]. It has been widely used since 1990s and promoted the prosperity of recommendation system [32]. Compared with content-based approaches, it doesn't requiring the creation of explicit profiles, but relies only on past user actions such as previous transactions or item ratings. Collaborative filtering can be further divided into two primary areas: the neighborhood methods and latent factor models [33]. The neighborhood methods include user-based methods and item-based methods. The user-based algorithm measures the similarity of two customers in various ways to identify users' neighbors and generates recommendations based on a few customers who are most similar to the user, while the item-based algorithm tracks user preferences by identifying similar items rather than users. Alternatively, latent factor models that tries to explain the ratings by characterizing both items and users on a fixed length vector, or a latent factor, inferred from the ratings patterns. Some of the most successful realizations of latent factor models are based on matrix factorization. It characterizes both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. However, both neighborhood methods and latent factor models will still have the content similarity problem. Besides, it has been mentioned in [34] that e-commerce recommender is different from movie or music recommendation systems. They should take into account the utility and utility plus of items, rather than recommend similar items based on users' historical records.

In order to handle the content similarity problem, some researches take the sequential and temporal information contained in users' historical records into account. [4], [35], [36] check the temporal dynamics in recommendation systems.

[37] investigate how to extract sequential patterns for next state prediction, and describe a sequential recommender based on Markov chains. [38] discover sequential patterns using pattern mining methods and generating recommendations. [39] also develop a Markov chain based recommendation system using Markov decision processes (MDP). To recommend more personalized items to different users, [4] using personalized transition graphs to combine the benefits of sequential Markov chain with time-invariant user taste. In this work, they proposed a model called Factorizing Personalized Markov Chains (FPMC) that combines the latent factor model and Markov chain model to predict what users will purchase the next time. However, the FPMC algorithm only utilizes the order of purchased products. More information, such as the time gaps between different purchases, can be utilized to improve the temporal diversity in recommendation system. In addition to historical purchase actions, other types of actions such as *click*, *collect* and *add-to-cart* can also be utilized. Our Multifaceted-FPMC model incorporates all these information and jointly factorizes latent feature vectors for different observable features. Besides, our proposed Time-decayed Multifaceted-FPMC model further considers the temporal influence decay phenomenon of users' historical actions to improve the accuracy of users' next purchases prediction.

Factorization models have attracted a lot of researchers with their excellent prediction capabilities shown in several applications. Factorization machines (FM) [40] was proposed to combine the advantages of general machine learning classifiers such as Support Vector Machines (SVM) with factorization models. It models the pairwise interactions between all features in a real valued feature vector. However, when considering tens of context features, the feature vectors will be quite long and not all the pairwise feature interactions are meaningful. We propose BSFM that subsumes our new proposed models and express them in a unified manner. Compared with traditional FM, it allows flexible feature interaction settings through a feature interaction matrix.

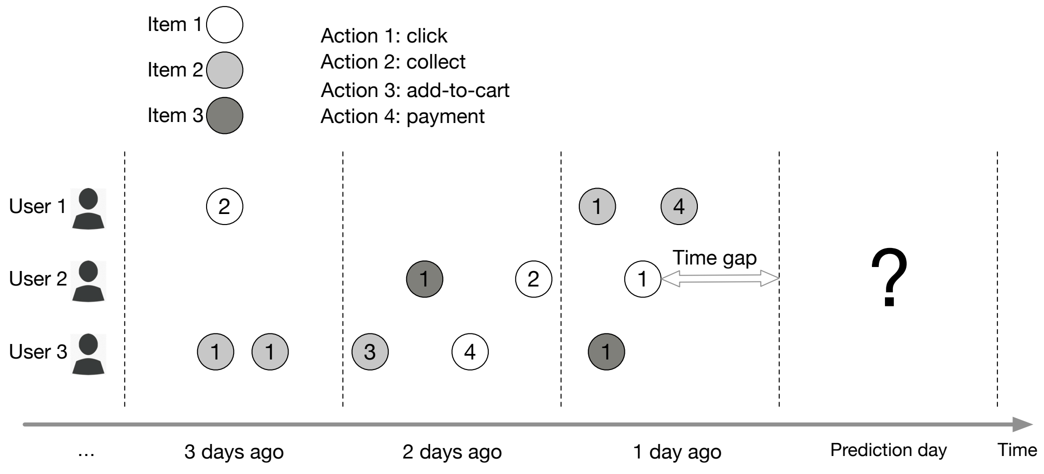


Fig. 3.1. Users' historical actions sequence on different items.

TABLE 3.1
HISTORICAL ACTION RECORDS TABLE FIELDS

COLUMN	DESCRIPTION
user id	Identity of users
item id	Identity of items
action type	The user action type, including *click, collect, add-to-cart* and *payment*.
time	The time of the action to the nearest hours.

3.3 Next-Day-Purchase Prediction

In this section, we first present the users' historical action records dataset provided by the Alibaba Mobile Recommendation Algorithm Competition that organized by the Alibaba Cloud Computing Ltd. The competition is based on the real users-commodities action data on Alibaba's M-Commerce platforms. We then define our problem formally and present the key notations used.

3.3.1 Data Description

The Alibaba dataset contains the complete action records data of $10,000$ users on $2,876,947$ items. The time span of historical records is from November 18^{th}, 2014 to December 18^{th}, 2014. Figure 3.1 illustrates the concept of online shopping records. For each user, we record his/her different kinds of actions on different items, together with the time he/she perform that action. For each historical action, the *user id*, *item id*, *action type* and *time* information are recorded. Table 3.1 describes the different table fields for recording each historical action.

3.3.2 Definitions and Notations

We now formally define our next-day-purchase prediction problem. For ease of reference, we define the notations used in this chapter, and list the key notations in Table 3.2.

Historical user action records: denote a set of users as $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and a set of items as $\mathcal{I} = \{i_1, i_2, ..., i_{|\mathcal{I}|}\}$. A historical user action record is a tuple $(u, i, a, t) \in \mathcal{R}$, where $u \in \mathcal{U}$, $i \in \mathcal{I}$, $a \in \{1, 2, 3, 4\}$ that represent action *click*, *collect*, *add-to-cart* and *payment* respectively, and t represents the time when user u performs action a on item i with accuracy of hours. $\mathcal{R}$ is the set of all historical action records.

Given the definition of users' historical action records, our problem is defined as following.

Problem definition: given a set of users $\mathcal{U}$, a set of items $\mathcal{I}$, and the historical user action records $\mathcal{R}$ between $\mathcal{U}$ and $\mathcal{I}$ during the last T days, the task is to predict which users will purchase which items on the next day.

3.4 Modeling Next-Day-Purchase Probability

In this section, we present our new proposed models for online purchase prediction problem. First, we describe the Matrix Factorization approach and the Factorizing Personalized Markov Chains model for modeling users' interest on items, and show their limitations for fully utilizing the context information contained in users' historical action records data. Second, we characterize the Alibaba dataset and propose our Multifaceted-FPMC model. Third, we further explore the temporal influ-

TABLE 3.2
NOTATIONS USED IN THIS CHAPTER

SYMBOL	DESCRIPTION
$\mathcal{U}, \mathcal{I}, \mathcal{R}$	set of users, items, historical action records
u, i, a, t, T	a user, an item, action, time, history length by day
d	day-of-week
$p(u, i)$	the probability user u will buy item i
$p(u, i\|j)$	the conditional probability user u will buy item i given that he/she already purchased item j
$p(u, i\|j, a, d)$	the conditional probability user u will buy item i on predicted day d given that he/she already performed action a on item j
v	latent feature vector
$\mathcal{B}_u$	the item set user u previously purchased
$\mathcal{B}_{u,a}$	the item set user u previously performed action a on
$C_{u,j,a}$	the total times user u performed action a on item j
$t_{u,j,a,c}$	the time interval length between the time user u perform the c-th action a on item j and the predicted day
$\mathbf{x}, y$	feature vector, target
$w_0, \mathbf{w}, V, \Phi$	model parameters of BSFM
$\mu^0, \lambda^0, \mu_\pi^w, \lambda_\pi^w, \mu_{f,\pi}^v, \lambda_{f,\pi}^v$	Regularization parameters of BSFM
$\alpha_0, \beta_0, \alpha_\lambda, \beta_\lambda, \mu_0, \gamma_0$	hyperparameters of BSFM

ence decay phenomenon of users' historical actions, and propose the Time-decayed Multifaceted-FPMC model that incorporates this phenomenon for purchase probability estimation.

3.4.1 From Matrix Factorization to Factorizing Personalized Markov Chains

Traditional MF algorithm characterizes users and items by latent feature vectors inferred from user-item ratings. Figure 3.2 illustrates how matrix factorization approach works. It assumes a low-rank structure for the rating matrix and factorizes it. The interest of user u to item i is estimated to be proportional to the rating. Assuming the rating for user u on item i is $r_{u,i}$, matrix factorization model approximates

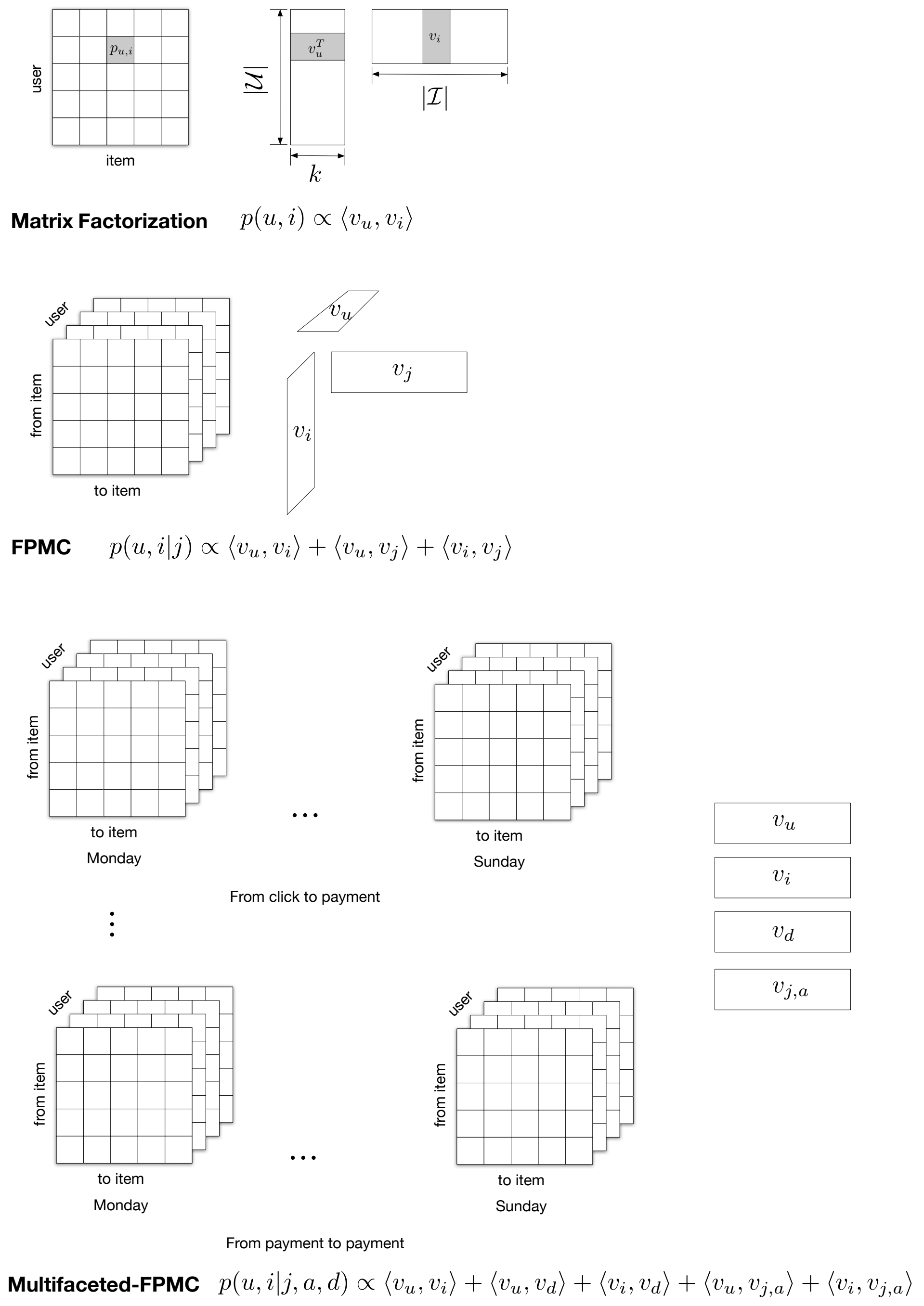

user
item
$p_{u,i}$
v_u^T
$|\mathcal{U}|$
k
v_i
$|\mathcal{I}|$
Matrix Factorization $p(u,i) \propto \langle v_u, v_i \rangle$
user
from item
to item
v_u
v_j
v_i
FPMC $p(u,i|j) \propto \langle v_u, v_i \rangle + \langle v_u, v_j \rangle + \langle v_i, v_j \rangle$
user
from item
to item
Monday
user
from item
to item
Sunday
From click to payment
v_u
v_i
v_d
$v_{j,a}$
user
from item
to item
Monday
user
from item
to item
Sunday
From payment to payment
Multifaceted-FPMC $p(u,i|j,a,d) \propto \langle v_u, v_i \rangle + \langle v_u, v_d \rangle + \langle v_i, v_d \rangle + \langle v_u, v_{j,a} \rangle + \langle v_i, v_{j,a} \rangle$

Fig. 3.2. Compare different models. The more information utilized, the more latent feature vectors extracted.

the interest of user u on item i by

$$p(u,i) \propto r_{u,i} = \langle v_u, v_i \rangle, \tag{3.1}$$

where $v_u \in \mathbb{R}^k$ is the latent vector that describes user u and $v_i \in \mathbb{R}^k$ is the latent vector of item i. More details about matrix factorization for recommender systems can be found in [2].

The Factorizing Personalized Markov Chains (FPMC) model proposed in [4] further considers the most recent shopping basket of a user and factorizing a transition cube that contains the transition matrix of each user. Assuming C is a transition cube, then $C_{u,i,j}$ denotes the probability user u will buy item i if he/she already bought item j. The original FPMC model utilizes a special case of Tucker Decomposition. Its form is further simplified when implemented by factorization machines [41], [42]. Assuming $\mathcal{B}_u$ is the item set of user u's last shopping basket, $p(u,i|j)$ gives the probability user u will buy item i when he/she already bought item j in the most recent shopping basket. The FPMC model approximates the probability by

$$p(u,i|j) \propto \langle v_u, v_i \rangle + \langle v_u, v_j \rangle + \langle v_i, v_j \rangle, \tag{3.2}$$

where $v_u \in \mathbb{R}^k$, $v_i \in \mathbb{R}^k$ and $v_j \in \mathbb{R}^k$ are the latent feature vectors of the user, the predicted item, and the item the user purchased in the most recent shopping basket. The FPMC model assumes the same prior probability $p(j) = \frac{1}{|\mathcal{B}_u|}$ for different historical purchased items j. Thus, the overall probability that user u will buy item i in the next shopping basket is given by

$$\begin{aligned} p(u,i) &= \sum_{j \in \mathcal{B}_u} p(u,i|j)p(j) \\ &\propto \sum_{j \in \mathcal{B}_u} \frac{1}{|\mathcal{B}_u|}(\langle v_u, v_i \rangle + \langle v_u, v_j \rangle + \langle v_i, v_j \rangle) \\ &\propto \langle v_u, v_i \rangle + \sum_{j \in \mathcal{B}_u} \frac{1}{|\mathcal{B}_u|}(\langle v_u, v_j \rangle + \langle v_i, v_j \rangle), \end{aligned} \tag{3.3}$$

By jointly training these latent feature vectors, the FPMC model combines the

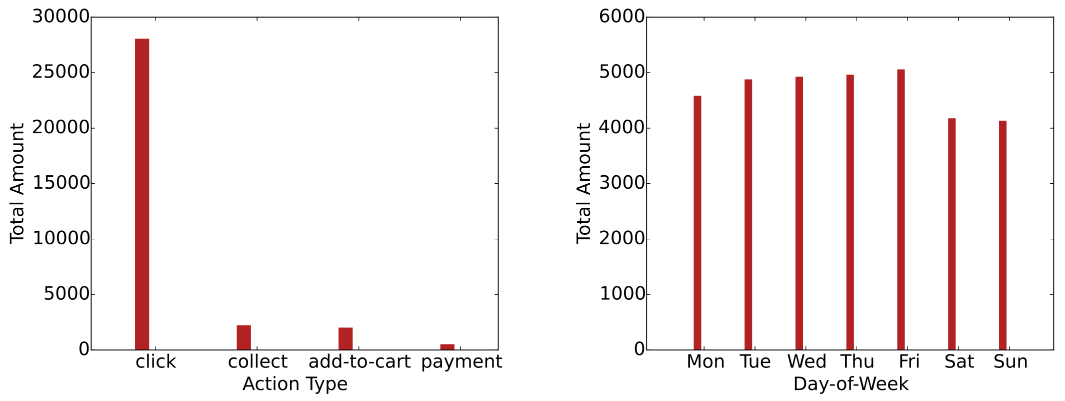


Fig. 3.3. Left: amounts of different types of actions in the Alibaba dataset. Right: amounts of all actions on different day-of-week.

advantages of both MF and Markov chain model. We refer to [4] for more details about FPMC model.

However, given users' historical action records data, more context information can be considered for next-day-purchase prediction. First, the FPMC model only considers the item set a user bought in his/her last purchases. However, rather than only considering the last purchases of a user, we can also take other types of actions into account. Besides, other factors, such as time, will also have influence on users' future decisions and thus can be utilized to improve prediction accuracy. Second, the FPMC model utilize the most recent shopping basket of a user. The time interval between a shopping basket and the previous shopping basket is not considered. Third, when summarizing the conditional probabilities $p(u, i|j)$, the same prior probability $p(j) = \frac{1}{|\mathcal{B}_u|}$ is assigned to different items $j \in \mathcal{B}_u$. However, it is not reasonable to assume that different historical purchased items are of the same importance without considering the time and amount an item has been purchased.

3.4.2 Multifaceted Factorizing Personalized Markov Chains

We perform extensive analysis on the Alibaba dataset. Figure 3.3 compares the amounts of different kinds of actions, and the amounts of total user actions on different day-of-week. First, the amount of *click* is much more than other actions, which is reasonable as customers will browse lots of similar items before they perform any further actions on an item. Besides, the amounts of action *collect* and *add-to-cart* are also larger than action *payment*. Without taking all kinds of historical actions into account, the information contained in the dataset won't be fully utilized. Second, as we can see, customers are more active during weekdays than weekends. The potential explanation is that people are more prefer to go outside for other activities or just buy things in a real mall rather than stay home and shopping online.

Based on the observations, we propose our Multifaceted-FPMC model to estimate a user-item pair's next-day-purchase probability. Denote the item set user u performed action a on during the last T days as $\mathcal{B}_{u,a}$. Assuming $d \in \{1, 2, ..., 7\}$ represents which day-of-week it is on the predicted day, where value $1 \sim 7$ represents Monday$\sim$Sunday respectively. The probability user u will buy item i on the predicted day d, given that he/she performed historical action a on item j during the last T days, is given by

$$p(u, i|j, a, d) \propto \langle v_u, v_i \rangle + \langle v_u, v_d \rangle + \langle v_i, v_d \rangle + \langle v_u, v_{j,a} \rangle + \langle v_i, v_{j,a} \rangle, \tag{3.4}$$

where $v_d \in \mathbb{R}^k$ is the latent feature vector for different day-of-week d, and $v_{j,a}$ is the latent feature vector of item $j \in \mathcal{B}_{u,a}$.

The Multifaceted-FPMC model estimates the probability of user u will purchase item i on the next day d by

$$\begin{aligned} p(u, i|d) &= \sum_{a=1}^{4} \sum_{j \in \mathcal{B}_{u,a}} p(u, i|j, a, d) p(j, a|d) \\ &\propto \langle v_u, v_i \rangle + \langle v_u, v_d \rangle + \langle v_i, v_d \rangle + \sum_{a=1}^{4} \sum_{j \in \mathcal{B}_{u,a}} \frac{1}{|\mathcal{B}_{u,a}|} (\langle v_u, v_{j,a} \rangle + \langle v_i, v_{j,a} \rangle), \end{aligned} \tag{3.5}$$

where $p(j, a|d) = \frac{1}{|\mathcal{B}_{u,a}|}$ is the prior probability of item $j \in \mathcal{B}_{u,a}$.

Compared with the FPMC model, our Multifaceted-FPMC model makes two differences. First, rather than only considering the historical *payment* actions, it

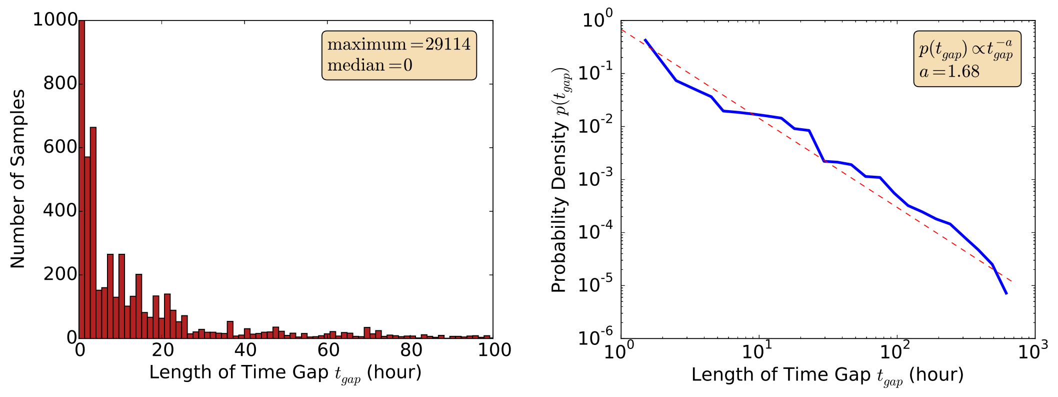


Fig. 3.4. Left: the histogram of time gaps between payment action and the previous action. Right: fit the relation between the sample numbers and the time gaps by power-law distribution.

also take other three types of actions into account. Second, it learns a latent feature vector for each day-of-week, as it has been shown in Figure 3.3 that users' passion for online purchase varies with different day-of-week. Figure 3.2 shows the idea intuitively. From MF to Multifaceted-FPMC model, more and more latent feature vectors are learned so that more context information is utilized for modeling purchase probability.

3.4.3 Time-decayed Multifaceted Factorizing Personalized Markov Chains

Previous models only consider *what* actions users have performed but ignore *when* exactly the historical actions were performed. In other words, if user u performed action a on item j_1 and j_2 in the last T days, it makes no difference for the two items if the time of actions are different. Besides, these models also ignore how many times user u performed action a on item j. For example, if user u clicked item $j1$ for 10 times during the last day, but only clicked item $j2$ for one time, it is highly possible that u is more interested in item $j1$ than item $j2$.

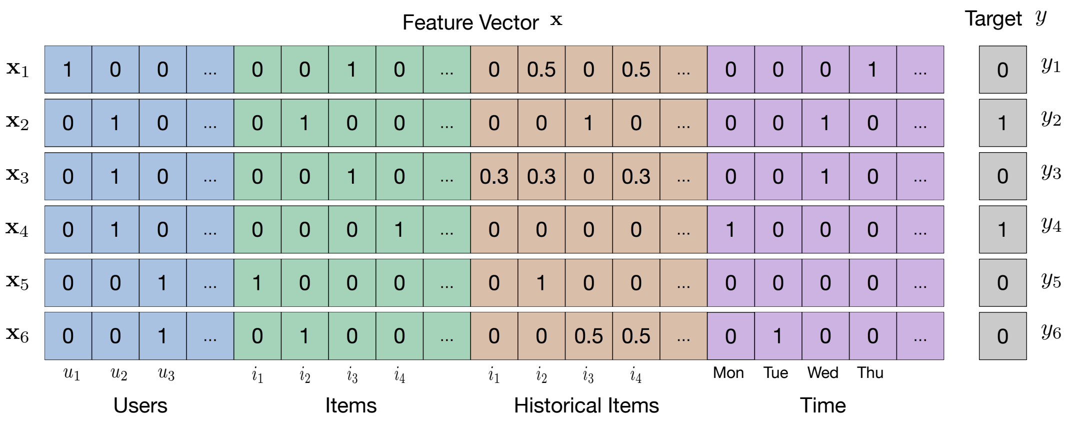
We look into how a user's previous action will influence his/her next purchase action with different time intervals between the two actions. For each user-item pair, We extract the time interval lengths between user's each purchase action and the previous action. Figure 3.4 shows the histogram of the time interval lengths extracted. The maximum value is the number of action pairs that the time interval between the two actions is less than one hour. The median value is zero, which means that more than a half of all action pairs' time interval are less than one hour. From Figure 3.4, we conclude that most of users usually make their decision to buy an item within one hour after he/she performed previous action on that item. The longer the time interval is, the less possible the user will finally purchase the item. Thus, previous actions' influence on users' purchase decision shall decay with time interval. More specifically, we use the power-law distribution [43] to model the temporal influence decay phenomenon. The right part of Figure 3.4 shows that the likelihood that a user will buy an item in t_{gap} hours after he/she performed the previous action on it fits power-law distribution very well, approximately proportional to $t_{gap}^{-1.68}$ with t_{gap} represents the time interval between purchase action and its previous action.

The temporal influence decay phenomenon of historical actions indicates that different historical actions shall be distinguished by the time interval length between the action time and the predicted day. In previous models, same prior probabilities are assigned to different historical items. To incorporate the temporal influence decay phenomenon into our online purchase prediction model, we further revise the prior probability $p(j, a)$.

Suppose user u performed action a on item j for totally $C_{u,j,a}$ times. The time interval length between the c-th action time and the predicted day d is $t_{u,j,a,c}$. We assign a prior probability $p(j, a|d)$ to item $j \in \mathcal{B}_{u,a}$ by the following equation

$$p(j, a|d) \propto \frac{\sum_{c=1}^{C_{u,j,a}} t_{u,j,a,c}^{-b}}{|\mathcal{B}_{u,a}|}. \tag{3.6}$$

According to the power-law fitting result in Figure 3.4, we set $b = 1.68$. Based on this prior distribution for different historical items j, we propose the Time-decayed Multifaceted Factorizing Personalized Markov Chains (Time-decayed Multifaceted-



	Users				Items					Historical Items					Time					Target y	
	u_1	u_2	u_3		i_1	i_2	i_3	i_4		i_1	i_2	i_3	i_4		Mon	Tue	Wed	Thu			
$\mathbf{x}_1$	1	0	0	...	0	0	1	0	...	0	0.5	0	0.5	...	0	0	0	1	...	0	y_1
$\mathbf{x}_2$	0	1	0	...	0	1	0	0	...	0	0	1	0	...	0	0	1	0	...	1	y_2
$\mathbf{x}_3$	0	1	0	...	0	0	1	0	...	0.3	0.3	0	0.3	...	0	0	1	0	...	0	y_3
$\mathbf{x}_4$	0	1	0	...	0	0	0	1	...	0	0	0	0	...	1	0	0	0	...	1	y_4
$\mathbf{x}_5$	0	0	1	...	1	0	0	0	...	0	1	0	0	...	0	0	0	0	...	0	y_5
$\mathbf{x}_6$	0	0	1	...	0	1	0	0	...	0	0	0.5	0.5	...	0	1	0	0	...	0	y_6

Feature Vector $\mathbf{x}$

Fig. 3.5. Example for representing a recommender problem with real valued feature vectors $\mathbf{x}$. Each row represents a feature vector $\mathbf{x}_i$ with its corresponding target y_i. For easier interpretation, we use different colors to group feature variables into indicators for different information such as user id, item id, historical purchased items and day-of-week.

FPMC) model that can be represented as the following equation:

$$p(u,i|d) = \sum_{a=1}^{4} \sum_{j \in \mathcal{B}_{u,a}} p(u,i|j,a,d)p(j,a|d)$$

$$\propto \langle v_u, v_i \rangle + \langle v_u, v_d \rangle + \langle v_i, v_d \rangle + \sum_{a=1}^{4} \sum_{j \in \mathcal{B}_{u,a}} \frac{\sum_{c=1}^{C_{u,j,a}} t_{u,j,a,c}^{-b}}{|\mathcal{B}_{u,a}|} (\langle v_u, v_{j,a} \rangle + \langle v_i, v_{j,a} \rangle). \tag{3.7}$$

3.4.4 Bayesian Sparse Factorization Machines

Now we have described the Multifaceted-FPMC model and the Time-decayed Multifaceted-FPMC model. We can actually further represent our models in a unified manner. Here we propose the Bayesian Sparse Factorization Machines (BSFM), a unified framework that subsumes all previously mentioned models.

Given a prediction problem, we assume it is described by a design matrix $X \in \mathbb{R}^{n \times p}$, where the i-th row $\mathbf{x}_i \in \mathbb{R}^p$ of X describes one case with p real-valued prediction variables. The prediction target of the i-th case is y_i. BSFM model the interactions between variables using factorized parameters. The model equation for a 2-order BSFM is defined as:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} \Phi_{j,j'} \langle \mathbf{v}_j, \mathbf{v}_{j'} \rangle x_j x_{j'}, \tag{3.8}$$

where $\mathbf{v}_j$ is the latent feature vector of length k for prediction variable x_j. The model parameters $\Theta = \{w_0, w_1, ..., w_p, v_{1,1}, ..., v_{p,k}\}$ are

$$w_0 \in \mathbb{R},\ \mathbf{w} \in \mathbb{R}^p,\ V \in \mathbb{R}^{p \times k}. \tag{3.9}$$

Compared with the traditional factorization machines (FM) model, the key difference we make in our BSFM model is that we add a matrix Φ to control the interactions between feature variable pairs in the feature vector $\mathbf{x}$. $\Phi \in \mathbb{R}^{p,p}$ is a sparse matrix that defines whether x_j and $x_{j'}$ have interaction with each other. With this sparse matrix, we can delete lots of meaningless feature interactions. That is why we name our unified model as Sparse Factorization Machines. Φ is defined as:

$$\Phi_{j,j'} = \begin{cases} 1 & \text{if } x_j \text{ interacts with } x_{j'} \\ 0 & \text{if } x_j \text{ doesn't interacts with } x_{j'} \end{cases} \tag{3.10}$$

The first part of BSFM is a linear regression model that contains the unary interactions of every x_j with the target. The second part contains the pairwise interactions of input variables. Compared with standard polynomial regression model, the key difference is that the interaction of x_j and $x_{j'}$ is not modeled by an independent parameter $w_{j,j'}$ but with a factorized parameterization $w_{j,j'} \approx \langle \mathbf{v_j}, \mathbf{v_{j'}} \rangle = \sum_{f=1}^{k} v_{j,f} v_{j',f}$ based on the assumption that the effect of pairwise interactions has a low rank [42]. The BSFM subsumes the factorization machines (FM) by setting all the elements of Φ to be 1.

We show that both our new proposed models and the traditional factorization models can be represented in the form of BSFM by defining appropriate feature vector $\mathbf{x}$ and interaction matrix Φ for each model. For different models, the corresponding feature vector $\mathbf{x}$ is defined as follows:

- **MF**: we can exactly approximate matrix factorization algorithm by defining the feature vector $\mathbf{x}$ using two categorical variables $\mathbf{x}_u \in \mathbb{R}^{|\mathcal{U}|}$ and $\mathbf{x}_i \in \mathbb{R}^{|\mathcal{I}|}$, that is,

$$\mathbf{x}_u = (\underbrace{0, ..., 0, 1, 0, ..., 0}_{|\mathcal{U}|}), \tag{3.11}$$

$$\mathbf{x}_i = (\underbrace{0, ..., 0, 1, 0, ..., 0}_{|\mathcal{I}|}), \tag{3.12}$$

where each variable in $\mathbf{x}_u$ denotes a user, and each variable in $\mathbf{x}_i$ denotes an item. For user-item pair (u, i), the u-th entry in $\mathbf{x}_u$ is 1, and similarly the i-th entry in $\mathbf{x}_i$ is 1, and the rest is 0 (e.g., see the first two groups of Figure 3.5). Using a feature vector $\mathbf{x} \in \mathbb{R}^{|\mathcal{U}|+|\mathcal{I}|}$ with binary indicator variables as the input of BSFM,

$$(u, i) \to \mathbf{x} = (\mathbf{x}_u, \mathbf{x}_i), \tag{3.13}$$

the BSFM will be exactly the same as a biased matrix factorization model [44], [45]:

$$\hat{y}(\mathbf{x}) = \hat{y}(u, i) = w_0 + w_u + w_i + \sum_{f=1}^{k} v_{u,f} v_{i,f}. \tag{3.14}$$

- **FPMC**: the FPMC algorithm incorporates the historical purchased item set and factorizing an MF model and Markov chain model jointly. We can also mimic the FPMC algorithm by adding a third part, $\mathbf{x}_4$ (4 represents action *payment*), to feature vector $\mathbf{x}$ [42]. $\mathbf{x}_4$ is a set-categorical variable for representing the items that have been purchased by a user in the past T days,

$$\mathbf{x}_4 = (\underbrace{0, ..., 1/|\mathcal{B}_u|, 0, ..., 1/|\mathcal{B}_u|, 0, ..., 0}_{|\mathcal{I}|,\ \text{historical purchased item set}}) \tag{3.15}$$

where the $|\mathcal{B}_u|$ non-zero elements in it represent the items purchased by user u. The feature vector representation $\mathbf{x} \in \mathbb{R}^{|\mathcal{U}|+2|\mathcal{I}|}$ will be

$$(u, i) \to \mathbf{x} = (\mathbf{x}_u, \mathbf{x}_i, \mathbf{x}_4). \tag{3.16}$$

- **Multifaceted-FPMC**: we can incorporate more context information by adding more feature sections into $\mathbf{x}$. Similar with $\mathbf{x}_4$ in FPMC algorithm, we add three more set-categorical variables, $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ that respectively represents the item sets a user has clicked, collected or added-to-cart in the past T days. Besides, considering users' passion for online shopping changes with different day-of-week, we can further add a categorical variable $\mathbf{x}_d$ to indicate which day-of-week it is on the predicted day.

$$\mathbf{x}_d = (\underbrace{0, ...0, 1, 0, ..., 0}_{7,\ \text{day-of-week}}). \tag{3.17}$$

  In this case, the feature vector $\mathbf{x} \in \mathbb{R}^{|\mathcal{U}|+5|\mathcal{I}|+7}$ will be

$$(user, item) \rightarrow \mathbf{x} = (\mathbf{x}_u, \mathbf{x}_i, \mathbf{x}_d, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4). \tag{3.18}$$

- **Time-decayed Multifaceted-FPMC**: the Time-decayed Multifaceted-FPMC model can be represented in the form of BSFM similar with the Multifaceted-FPMC model. Assuming set-categorical variable x_a^t $(a = 1, 2, 3, 4)$ represents the items user u performed action a on in the past T days,

$$\mathbf{x}_a^t = \underbrace{(0, ..., \sum_{c=1}^{C_{u,j1,a}} t_{u,j1,a,c}^{-b}/|\mathcal{B}_{u,a}|, 0, ..., \sum_{c=1}^{C_{u,j2,a}} t_{u,j2,a,c}^{-b}/|\mathcal{B}_{u,a}|, 0, ..., 0)}_{|\mathcal{I}|,\ \text{historical purchased item set}} \tag{3.19}$$

  In this case, the feature vector $\mathbf{x} \in \mathbb{R}^{|\mathcal{U}|+5|\mathcal{I}|+7}$ for the Time-decayed Multifaceted-FPMC model will be

$$(u, i) \rightarrow \mathbf{x} = (\mathbf{x}_u, \mathbf{x}_i, \mathbf{x}_d, \mathbf{x}_1^t, \mathbf{x}_2^t, \mathbf{x}_3^t, \mathbf{x}_4^t). \tag{3.20}$$

By defining the above feature vectors, we represent our models in the form of BSFM. The corresponding interaction matrix Φ is set according to Equation 3.10.

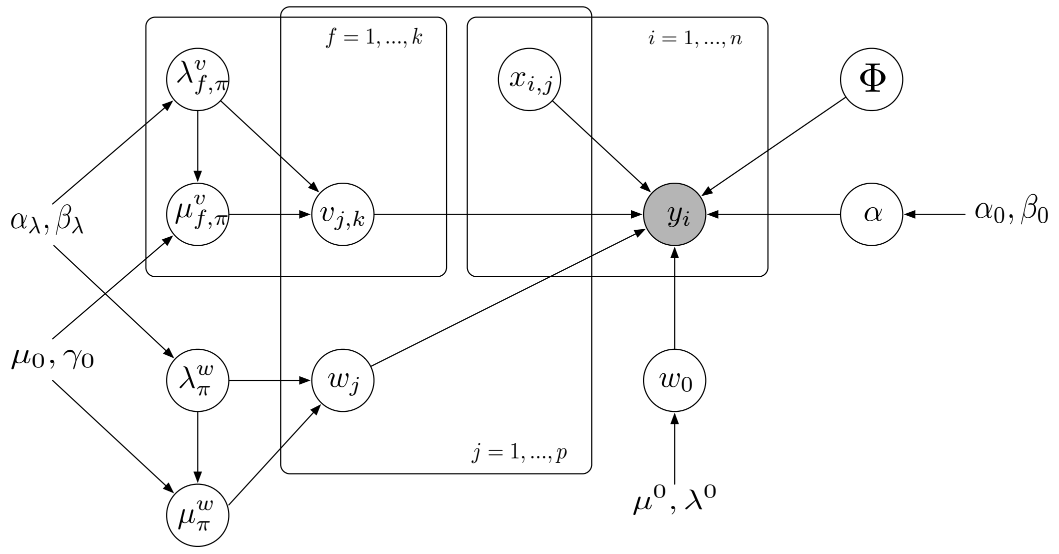


Fig. 3.6. Graphical representation of BSFM.

3.5 Markov Chain Monte Carlo Inference of Model Parameters

In this section, we introduce the Markov chain Monte Carlo (MCMC) [46] inference for learning model parameters in the form of BSFM. Compared with algorithms such as stochastic gradient descent (SGD) or alternative least square (ALS), the MCMC algorithm is able to do automatic hyperparameter learning or no learning hyperparameter. MCMC usually gives better accuracy with structured Bayesian inference [47]. Bayesian models with hierarchical hyperpriors is suggested in [48]. As we learn our model parameters by Bayesian inference, we name our unified model as Bayesian Sparse Factorization Machines. Here we present 2-order BSFM without loss of generality.

3.5.1 Model Structure

Figure 3.6 depicts the graphical representation of our hierarchically BSFM framework. According to the functionalities of different parameters, we divide them into three categories.

- Model parameters $\Theta := \{w_0, \mathbf{w}, V, \Phi\}$. The interaction matrix Φ is predefined according to the feature interactions in a specific model. Other parameters will be sampled by MCMC inference using Gibbs sampling.

- Regularization parameters $\Theta_H := \{\mu^0, \lambda^0, \mu_\pi^w, \lambda_\pi^w, \mu_{f,\pi}^v, \lambda_{f,\pi}^v\}$. These parameters regularize the model parameters to prevent overfitting. $\pi : \{1, ..., p\} \rightarrow \{1, ..., \Pi\}$ is a grouping of model parameters. For example, the regularization value for $v_{l,f}$ would be $\lambda_{f,\pi(l)}^v$.

- Hyperparameters $\Theta_0 := \{\alpha_0, \beta_0, \alpha_\lambda, \beta_\lambda, \mu_0, \gamma_0\}$. These parameter are introduced in BSFM so that the regularization values Θ_H can be automatically determined, which is a major advantage of MCMC. The number of hyperpriors $|\Theta_0|$ is smaller than the number of regularization parameters $|\Theta_H|$. More importantly, MCMC is typically insensitive to the choice of Θ_0 [42].

3.5.2 Inference: Efficient Gibbs Sampling

We use Gibbs sampling to draw from the posterior of BSFM since posterior inference is analytically intractable. Standard Gibbs sampling divides all inferred variables Θ and Θ_H into disjoint blocks and every block contains a subset of the parameters. However, it will leads to high time complexities in the final Gibbs sampler. Therefore, we use single parameter Gibbs sampling proposed in [42] instead of standard block Gibbs sampling. Similar with the procedures in [42], we exploit the multi-linear nature of BSFM for notational readability.

[Multi-linear Nature of BSFM] For each model parameter $\theta \in \Theta$, the BSFM is a linear combination of two functions g_θ and h_θ that are independent of the value θ. Therefore, equation 3.8 can be rewritten as [40]:

$$\hat{y}(\mathbf{x}|\Theta) := g_\theta(\mathbf{x}) + \theta h_\theta(\mathbf{x}), \forall \theta \in \Theta \tag{3.21}$$

where

$$h_\theta(\mathbf{x}) = \frac{\partial \hat{y}(\mathbf{x}|\Theta)}{\partial \theta} = \begin{cases} 1 & \text{if } \theta \text{ is } w_0 \\ x_l & \text{if } \theta \text{ is } w_l \\ x_l \sum_{j \neq l} \Phi_{l,j} v_{j,f} x_j & \text{if } \theta \text{ is } v_{j,f} \end{cases} \tag{3.22}$$

The value of g_θ will be computed by $g_\theta(\mathbf{x}) = \hat{y}(\mathbf{x}|\Theta) - \theta h_\theta(\mathbf{x})$ instead of computing directly, therefore its definition is omitted here.

We now describe the MCMC inference for BSFM. MCMC samples from the posterior distributions of parameters rather than learn an optimal value for each of them. For hyperparameters, as MCMC is typically insensitive to the choice of Θ_0, we choose $\alpha_0 = \beta_0 = \alpha_\lambda = \beta_\lambda = \gamma_0 = 1$ trivially.

For regularization parameters, MCMC places distributions on the priors for integration of Θ_H. By integrating regularization parameters into the model, it avoids a time-consuming search for these parameters. Specifically, for each pair $(\mu_\theta, \lambda_\theta) \in \Theta_H$ of prior parameters, we assume a Gamma distribution for each prior precision λ_θ and α except λ^0, and a normal distribution for each mean μ_θ of all model parameters $\theta \in \Theta$ but μ^0:

$$\lambda_\pi^w \sim \Gamma(\alpha_\lambda, \beta_\lambda),\ \mu_\pi^w \sim \mathcal{N}(\mu_0, \gamma_0 \lambda_\pi^w), \tag{3.23}$$

$$\lambda_{f,\pi}^v \sim \Gamma(\alpha_\lambda, \beta_\lambda),\ \mu_{f,\pi}^v \sim \mathcal{N}(\mu_0, \gamma_0 \lambda_{f,\pi}^v), \tag{3.24}$$

$$\alpha \sim \Gamma(\alpha_0, \beta_0). \tag{3.25}$$

Given n observed samples $(y_i, \mathbf{x}_i) \in \mathbb{R}^{p+1}$, the corresponding conditional posterior distributions for Θ_H are [42]:

$$\alpha | y, X, \Theta_0, \Theta \sim \Gamma\left(\frac{\alpha_0 + n}{2}, \frac{1}{2}\left[\sum_{i=1}^{n} \left(y_i - \hat{y}\left(\mathbf{x}_i|\Theta\right)\right)^2 + \beta_0\right]\right), \tag{3.26}$$

$$\lambda_\pi | \Theta_0, \Theta_H \backslash \{\lambda_\pi\}, \Theta \sim \Gamma\left(\frac{\alpha_\lambda + p^\pi + 1}{2}, \frac{1}{2}\left[\sum_{i=1}^{p} \delta\left(\pi(j) = \pi\right)\left(\theta_j - \mu_\theta\right)^2 + \gamma_0\left(\mu_\pi - \mu_0\right)^2 + \beta_\lambda\right]\right), \tag{3.27}$$

$$\mu_\pi|\Theta_0,\Theta_H\backslash\{\lambda_\pi\},\Theta\sim\mathcal{N}\left(\frac{1}{p^\pi+\gamma_0}\left[\sum_{i=1}^{p}\delta\left(\pi(j)=\pi\right)\theta_j+\gamma_0\mu_0\right],\frac{1}{\left(p^\pi+\gamma_0\right)\lambda_\pi}\right), \tag{3.28}$$

where

$$p^\pi:=\sum_{i=1}^{p}\delta\left(\pi(j)=\pi\right), \tag{3.29}$$

and δ is the indicator function

$$\delta(b):=\begin{cases}1 & \text{if } b \text{ is true}\\ 0 & \text{if } b \text{ is false}\end{cases} \tag{3.30}$$

For model parameters, we assume normal distribution. With n observed samples $(y_i, \mathbf{x}_i)$, the corresponding conditional posterior distributions for Θ satisfy:

$$p(\Theta|\mathbf{y},X,\Theta_H)\propto\prod_{i=1}^{n}\sqrt{\alpha}e^{-\frac{\alpha}{2}(y_i-y(\mathbf{x}_i,\Theta))^2}\prod_{\theta\in\Theta}\sqrt{\lambda_\theta}e^{-\frac{\lambda_\theta}{2}(\theta-\mu_\theta)^2}. \tag{3.31}$$

Using the multi-linear representation form of BSFM, we can infer that the conditional posterior distribution for each model parameter $\theta\in\Theta$ is:

$$\theta|X,y,\Theta\setminus\{\theta\},\Theta_H\sim\mathcal{N}(\tilde{\mu}_\theta,\tilde{\sigma}_\theta^2), \tag{3.32}$$

where

$$\tilde{\sigma}_\theta^2:=\left(\alpha\sum_{i=1}^{n}h_\theta^2\left(\mathbf{x}_i\right)+\lambda_\theta\right)^{-1}, \tag{3.33}$$

$$\tilde{\mu}_\theta:=\tilde{\sigma}_\theta^2\left(\alpha\theta\sum_{i=1}^{n}h_\theta^2\left(\mathbf{x}_i\right)+\alpha\sum_{i=1}^{n}h_\theta\left(\mathbf{x}_i\right)e_i+\mu_\theta\lambda_\theta\right), \tag{3.34}$$

e_i is the prediction error of the i-th sample:

$$e_i:=y_i-\hat{y}(\mathbf{x}_i|\Theta). \tag{3.35}$$

3.5.3 Learning Procedures

Algorithm 3.1 depicts the learning procedures of Gibbs sampling for BSFM.

Algorithm 3.1 Markov Chains Monte Carlo Inference (MCMC) for BSFM

Input: Training data S_{train}, test data S_{test}, initialization σ, Φ.
Output: Prediction $\hat{y}_{test}$ for test cases.
Initialization Step:

```
1: $w_0 \leftarrow 0$; $\mathbf{w} \leftarrow (0, ..., 0)$; $\mathbf{V} \sim \mathcal{N}(0, \sigma)$
2: $\#_{samples} \leftarrow 0$
```

Gibbs Sampling Step:

```
 1: repeat
 2:     $\hat{\mathbf{y}} \leftarrow$ predict all cases $S_{train}$
 3:     $\mathbf{e} \leftarrow \mathbf{y} - \hat{\mathbf{y}}$
 4:     //update the regularization parameters
 5:     sample $\alpha$ using Equation 3.26
 6:     for $(\mu_\pi, \lambda_\pi) \in \Theta_H$ do
 7:         sample $\lambda_\pi$ using Equation 3.27
 8:         sample $\mu_\pi$ using Equation 3.28
 9:     //update the model parameters
10:     sample $w_0$ from $\mathcal{N}(\tilde{\mu}_{w_0}, \tilde{\sigma}^2_{w_0})$
11:     for $l \in \{1, ..., p\}$ do
12:         sample $w_l$ from $\mathcal{N}(\tilde{\mu}_{w_l}, \tilde{\sigma}^2_{w_l})$
13:         update $\mathbf{e}$
14:     for $f \in \{1, ..., k\}$ do
15:         for $l \in \{1, ..., p\}$ do
16:             sample $v_{l,f}$ from $\mathcal{N}(\tilde{\mu}_{v_{l,f}}, \tilde{\sigma}^2_{v_{l,f}})$
17:             update $\mathbf{e}$
18:     $\#_{samples} \leftarrow \#_{samples} + 1$
19:     $\hat{\mathbf{y}}^*_{test} \leftarrow$ predict all cases $\mathbf{S}_{test}$
20:     $\hat{\mathbf{y}}_{test} \leftarrow \hat{\mathbf{y}}_{test} + \hat{\mathbf{y}}^*_{test}$
21: until stopping criterion is met
22: $\hat{\mathbf{y}}_{test} \leftarrow \frac{1}{\#_{samples}} \hat{\mathbf{y}}_{test}$
```

First, we initialize the model parameters to be zero or random values. For every sampling iteration, we sample the regularization parameters and model parameters in sequence. Before sampling the next parameter, the depending variables and parameters must be updated using the sampled new parameters.

We need to make two changes for binary classification task. First, after we get the probabilities, we need to map the normal distributed $\hat{y}$ to a probability $P(\hat{y}) \in [0, 1]$ that defines the Bernoulli distribution for binary classification [48]. Here we use the CDF function of a normal distribution for mapping:

$$P(\hat{y}) := \Phi(\hat{y}). \tag{3.36}$$

Second, in algorithm 3.1, instead of regressing to y, we sample it in each iteration from its posterior that has a truncated normal distribution

$$y_i'|\mathbf{x}_i, y_i, \Theta \sim \begin{cases} \mathcal{N}(\hat{y}(\mathbf{x}_i, \Theta), 1)\delta(y_i' < 0) & \text{if } y_i \text{ belongs to negative class} \\ \mathcal{N}(\hat{y}(\mathbf{x}_i, \Theta), 1)\delta(y_i' \geqslant 0) & \text{if } y_i \text{ belongs to positive class} \end{cases}. \tag{3.37}$$

Sampling from this distribution is efficient [49].

3.6 Experiments

In this section, we compare our approaches with multiple state-of-the-art algorithms and show the benefits of incorporating various context information and the temporal influence decay phenomenon of historical actions.

3.6.1 Experimental Setup and Metrics

Our performance evaluation is conducted on a subset of the Alibaba dataset. Specifically, we only keep the users who have more than 20 purchase (or *payment*) actions during November 18^{th}, 2014 to December 18^{th}, 2014 and the items that have been bought by at least one user. After filtered by this criteria, the dataset contains the historical action records of 1299 users and 1445 items. Our objective is predict the user-item pairs that will have purchase actions on a prediction date based on previous action records. We run experiments on different prediction dates and observed similar results. Thus, without loss of generality, we fix the prediction date to be December $18th$, 2014 and report our experimental results to evaluate different approaches' performance.

To evaluate the performance of different approaches, we are interested in finding out how many user-item pairs that have purchase actions on the predicted day can be correctly predicted by different methods. Denote N_{TP} as the number of correctly predicted user-item pairs, N_T as the number of total user-item pairs that actually have purchase actions on the predicted day, and N_P the number of total predictions, we compute the following metrics for evaluation:

- *Precision@N*: the ratio of correctly predicted pairs to the N_P predicted pairs

$$Precision@N = \frac{N_{TP}}{N_P}, \tag{3.38}$$

- *Recall@N*: the ratio of correctly predicted pairs to the N_T pairs that really have purchase actions on the predicted day

$$Recall@N = \frac{N_{TP}}{N_T}, \tag{3.39}$$

- *F1 Score*: the comprehensive evaluation metric that combines both prediction precision and recall rate

$$F1@N = \frac{2 \times Precision@N \times Recall@N}{Precision@N + Recall@N}. \tag{3.40}$$

In our experiments, we test the performance with $N = 300$, 600 and 1000.

3.6.2 Evaluated Approaches

We compare our approaches with multiple state-of-the-art algorithms. By incorporating different features into the feature vector that describes a user-item pair, we are able to utilize different context information and factorize multifaceted latent factors for prediction and recommendation. Specifically, we compare the performance of the following methods: MF, FPMC, Multifaceted-FPMC and Time-decayed Multifaceted-FPMC.

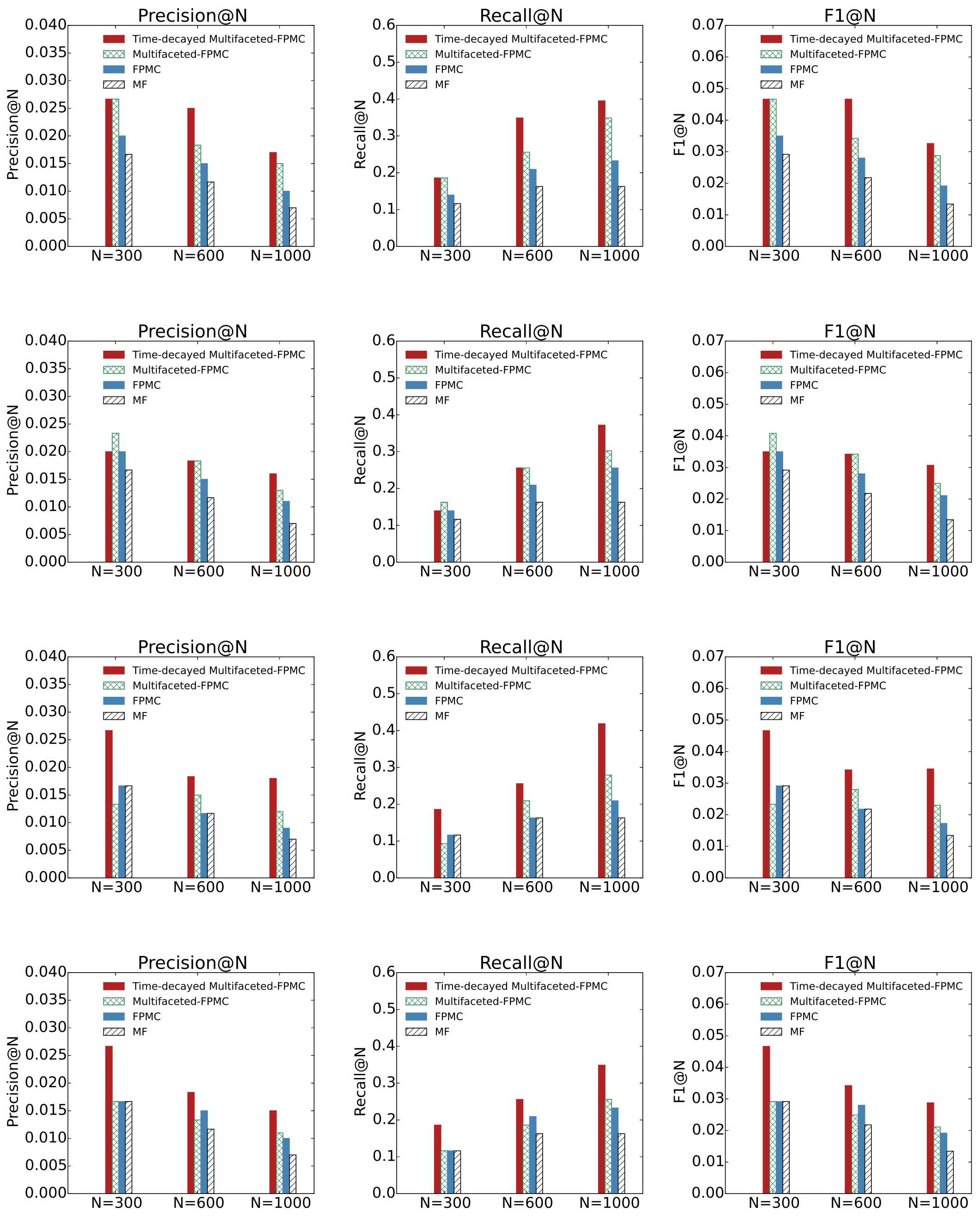


Fig. 3.7. Compare different algorithms with various history lengths. From top to down, the history lengths are $T = 1\ day,\ 3\ days,\ 14\ days,\ 28\ days.$

3.6.3 Performance Evaluation

We compare the performance of different approaches on the filtered Alibaba dataset. The lengths of latent vectors are fixed to be $k = 10$. The $Precision@N$, $Recall@N$

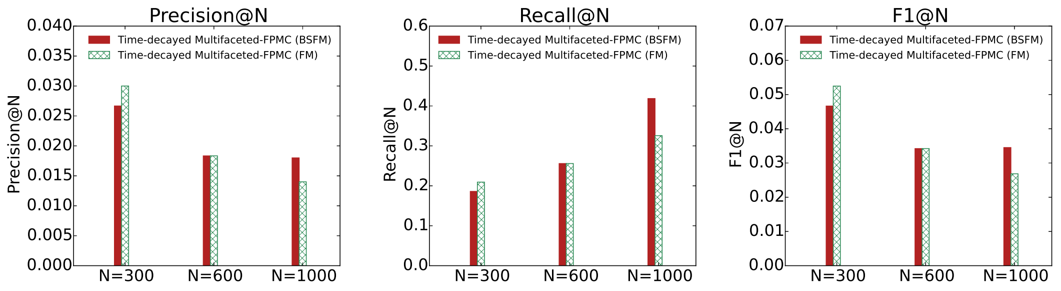


Fig. 3.8. Compare the performance of Time-decayed Multifaceted-FPMC model and the FM version Time-decayed Multifaceted-FPMC model (history length $T = 14\ days$). The FM version replaces all elements of interaction matrix Φ with 1.

and $F1@N$ with $N \in \{300, 600, 1000\}$ and the time span of historical records $T \in \{1\ day, 3\ days, 14\ days, 28\ days\}$ are reported in Figure 3.7.

Figure 3.7 significantly demonstrates the effectiveness of our new proposed models. The Multifaceted-FPMC model outperforms the FPMC model and MF model significantly in most cases, which proves the importance of taking all types of historical actions into account. When history length is not long (such as 3 days or 7 days), incorporating actions other than *payment* is quite necessary. The reason is that it is highly possible that users may have only a few purchase actions or even no purchase action during the last few days. However, the times of other actions such as *click* are usually much more than *payment* action. In this case, the Multifaceted-FPMC model achieves benefit by utilizing other actions' information for future purchase prediction.

Compared with other approaches, the performance of Time-decayed Multifaceted-FPMC keeps being the best under different experiment settings. As we can see, after taking the time intervals of historical actions into account and model the temporal influence decay phenomenon by power-law distribution, the Time-decayed Multifaceted-FPMC model further improves the prediction accuracy and outperforms all other approaches. This demonstrates the key role of time intervals for purchase prediction.

We also compare the performance of Time-decayed Multifaceted-FPMC model with the fully interaction version of Time-decayed Multifaceted-FPMC model. In the fully interaction version of Time-decayed Multifaceted-FPMC model, the difference is that all the elements of interaction matrix Φ is 1. In this case, it is equivalent with an factorization machine (FM) model. Figure 3.8 shows the performance comparison of two versions of Time-decayed Multifaceted-FPMC model using 14 days of history. As we can see, the FM version of Time-decayed Multifaceted-FPMC model doesn't outperform our proposed Time-decayed Multifaceted-FPMC model. The reason is that FM models the pairwise interactions between all variable pairs. However, with long feature vector $\mathbf{x}$ that contains lots of prediction variables, not every pairwise interaction will be meaningful.

3.7 Conclusions

To summarize, traditional approaches for recommendation and future purchase prediction, such as MF and FPMC, are insufficient to fully utilize the various context information contained in users' historical records data.

In this chapter, based on the users' historical action records dataset from Alibaba group, we investigate the characters of real users' actions and get some insights. First, we show that different types of actions user performed previously are all helpful for users' future purchase prediction. Based on this discovery, we propose our Multifaceted-FPMC model that utilizes all different kinds of actions. Second, we further observe that users' historical actions' influence on their future purchase actions decays with the time intervals between historical actions and purchase actions. The decay speed is approximately following a power-law distribution. Based on this temporal influence decay phenomenon, we further propose our Time-decayed Multifaceted-FPMC model for future purchase probability estimation. Finally, we show that our models can be represented in a unified manner and propose the BSFM framework. Extensive evaluations show that the proposed models achieves better performance than previous approaches such as MF and FPMC.

Chapter 4

Conclusion and Future Work

Given sparse observed data, we refer to latent factor models related to matrix or tensor approximation to reveal the hidden structure of it and explain it by learning the latent factors beneath the data. In the thesis, we look into two different problems and solve them by proposing new models based on matrix or tensor approximation techniques.

The first problem is about mobile network latency prediction. Given a small percentage of measured latencies between mobile network nodes, our objective is to predict the remaining unmeasured latencies. Traditional methods include Euclidean embedding and matrix factorization. We argue that both of the two methods are not appropriate to explain the structure of the network latency. We decompose the network latency matrix into two components: the Euclidean distance matrix part, which comes from the geographical Euclidean distances between network nodes; the network feature matrix part, which accounts for the different network conditions of different nodes. We assume a low-rank structure for the network feature matrix and learn it by matrix completion. For the distance matrix part, we learn it by traditional Euclidean embedding algorithm (Vivaldi). We further propose an iterative algorithm to learn the two components jointly. Extensive evaluations on two datasets show that our proposed models and algorithms outperform state-of-the-art algorithms significantly. To further improve the prediction accuracy, we incorporate the information from historical network latency matrices by stacking these matrices into a big matrix and applying our algorithms on it. In this way, we further improve the prediction accuracy of network latencies.

The second problem is about predict users' next-day-purchases based on his/her historical action records. This will help enterprises to offer their customers personalized services and recommendations. We analysis the dataset from the Alibaba group. Based on our observations, we propose the Multifaceted-FPMC and Time-decayed Multifaceted-FPMC models to estimate user-item pairs' purchase probabilities using various information from users' historical records data. We further represents our models in a unified framework, Bayesian Sparse Factorization Machines (BSFM), and learn the model parameters using MCMC inference. Compared with traditional factorization machines, BSFM further introduces a interaction matrix that limits the interactions of feature variable pairs in the model, as not all the interactions between feature variables are meaningful. Thus, BSFM subsumes factorization machines as a special case, and it is able to mimic existing factorization models exactly by defining appropriate feature vectors and interaction matrices. We learn the model parameters by Markov chain Monte Carlo algorithm. Evaluations on the Alibaba dataset shows the effectiveness of our proposed models.

Lots of works can be done in the future. First, distributed algorithms and systems can be developed to improve the running speed of current prediction algorithms. Second, things are always changing. Network latencies or peoples' interests keep changing with time. Thus, online learning algorithms that are able to update model parameters based on newly collected data is in need. For example, in our D-F Decomposition algorithm for network latency problem, the network feature matrix F is used to characterize the network conditions. As the network conditions of personal devices are keep changing, online algorithms that can update the F matrix based on new measured latencies are highly valuable. Besides, customers' interest on items keeps changing with time, thus, developing online prediction algorithm is also very important and helpful for e-commerce. Third, BSFM learns a latent vector for each feature variable. However, each feature variable will have interactions with different kinds of other feature variables. Instead of learning a single latent vector for each variable, it may be more appropriate to learn multiple latent vectors for different interactions. Last but not the least, we can apply our proposed BSFM on other real-world problems.

References

[1] J. Bennett and S. Lanning, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007, 2007, p. 35.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[3] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[4] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 811–820.

[5] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 81–90.

[6] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, "Characterizing and modeling internet traffic dynamics of cellular devices," in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*. ACM, 2011, pp. 305–316.

[7] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4. ACM, 2004, pp. 15–26.

[8] T. E. Ng and H. Zhang, “Predicting internet network distance with coordinates-based approaches,” in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 2002, pp. 170–179.

[9] Y. Liao, W. Du, P. Geurts, and G. Leduc, “Dmfsgd: A decentralized matrix factorization algorithm for network distance prediction,” *Networking, IEEE/ACM Transactions on*, vol. 21, no. 5, pp. 1511–1524, 2013.

[10] J. Ledlie, P. Gardner, and M. I. Seltzer, “Network coordinates in the wild.” in *NSDI*, vol. 7, 2007, pp. 299–311.

[11] Y. Chen, X. Wang, C. Shi, E. K. Lua, X. Fu, B. Deng, and X. Li, “Phoenix: A weight-based network coordinate system using matrix factorization,” *Network and Service Management, IEEE Transactions on*, vol. 8, no. 4, pp. 334–347, 2011.

[12] G. Wang, B. Zhang, and T. Ng, “Towards network triangle inequality violation aware distributed systems,” in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 175–188.

[13] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, “Seattle: a platform for educational cloud computing,” in *ACM SIGCSE Bulletin*, vol. 41, no. 1. ACM, 2009, pp. 111–115.

[14] M. J. Pazzani and D. Billsus, “Content-based recommendation systems,” in *The adaptive web*. Springer, 2007, pp. 325–341.

[15] G. Linden, B. Smith, and J. York, “Amazon. com recommendations: Item-to-item collaborative filtering,” *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, 2003.

[16] B. Donnet, B. Gueye, and M. A. Kaafar, “A survey on network coordinates systems, design, and security,” *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 4, pp. 488–503, 2010.

[17] S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha, "On suitability of euclidean embedding of internet hosts," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 1. ACM, 2006, pp. 157–168.

[18] Y. Mao, L. K. Saul, and J. M. Smith, "Ides: An internet distance estimation service for large networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 12, pp. 2273–2284, 2006.

[19] Y. Liao, P. Geurts, and G. Leduc, "Network distance prediction based on decentralized matrix factorization," in *NETWORKING 2010*. Springer, 2010, pp. 15–26.

[20] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[21] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[22] Z. Lu, Y. Zhang, and X. Li, "Penalty decomposition methods for rank minimization," *Optimization Methods and Software*, no. ahead-of-print, pp. 1–28, 2014.

[23] K. LaCurts and H. Balakrishnan, "Measurement and analysis of real-world 802.11 mesh networks," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 123–136.

[24] J. Sommers and P. Barford, "Cell vs. wifi: on the performance of metro area mobile connections," in *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM, 2012, pp. 301–314.

[25] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 225–238.

[26] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[27] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[28] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 2003, pp. 143–152.

[29] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of optimization theory and applications*, vol. 109, no. 3, pp. 475–494, 2001.

[30] T. Westergren, "The music genome project," *Online: http://pandora.com/mgp*, 2007.

[31] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[32] W. Gu, S. Dong, and Z. Zeng, "Increasing recommended effectiveness with markov chains and purchase intervals," *Neural Computing and Applications*, vol. 25, no. 5, pp. 1153–1162, 2014.

[33] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 89–115, 2004.

[34] B. Li, A. Ghose, and P. G. Ipeirotis, "Towards a theory model for product search," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 327–336.

[35] Y. Koren, “Collaborative filtering with temporal dynamics,” *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[36] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun, “Temporal recommendation on graphs via long-and short-term preference fusion,” in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 723–732.

[37] A. Zimdars, D. M. Chickering, and C. Meek, “Using temporal data for making recommendations,” in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2001, pp. 580–588.

[38] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, “Using sequential and non-sequential patterns in predictive web usage mining tasks,” in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 2002, pp. 669–672.

[39] G. Shani, R. I. Brafman, and D. Heckerman, “An mdp-based recommender system,” in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 453–460.

[40] S. Rendle, “Factorization machines,” in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 995–1000.

[41] I. Bayer and S. Rendle, “Factor models for recommending given names,” *ECML PKDD Discovery Challenge*, p. 81, 2013.

[42] S. Rendle, “Factorization machines with libfm,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, p. 57, 2012.

[43] J. Alstott, E. Bullmore, and D. Plenz, “powerlaw: a python package for analysis of heavy-tailed distributions,” 2014.

[44] A. Paterek, “Improving regularized singular value decomposition for collaborative filtering,” in *Proceedings of KDD cup and workshop*, vol. 2007, 2007, pp. 5–8.

[45] N. Srebro, J. Rennie, and T. S. Jaakkola, “Maximum-margin matrix factorization,” in *Advances in neural information processing systems*, 2004, pp. 1329–1336.

[46] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, “An introduction to mcmc for machine learning,” *Machine learning*, vol. 50, no. 1-2, pp. 5–43, 2003.

[47] C. Freudenthaler, L. Schmidt-Thieme, and S. Rendle, “Bayesian factorization machines,” 2011.

[48] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis*. Taylor & Francis, 2014, vol. 2.

[49] C. P. Robert, “Simulation of truncated normal variables,” *Statistics and computing*, vol. 5, no. 2, pp. 121–125, 1995.