2012 IEEE 19th International Conference on Web Services

Decision Tree Learning from Incomplete QoS to Bootstrap Service Recommendation

Qi Yu
College of Computing and Information Sciences
Rochester Institute of Technology, USA
Email: qi.yu@rit.edu

***Abstract*—Collaborative Filtering (CF) has been increasingly employed as an effective vehicle for providing personalized service recommendations in service computing. CF exploits historical user-service interaction information to predict the preference of service users. A key challenge faced by CF is to handle new users with no previous interaction information. We present a novel strategy that integrates Matrix Factorization (MF) with decision tree learning to bootstrap service recommendation systems. The proposed strategy first employs MF to partition existing users into a set of user groups. In practice, only a small amount of user-service interaction information is observed. The MF based user partitioning scheme also provides a way to estimate the missing interaction information based on the group structure. The tree learning algorithm then leverages these estimated information and exploits user groups as class labels to learn a decision tree. Few highly discriminative services are identified as tree nodes to adaptively query a new user based on the interaction results with the prior services in the tree. Through a short and intuitive bootstrapping process, the new user is classified into one of the user groups, via which the user's preference is predicted. We conduct a set of experiments on real-world service data to demonstrate the effectiveness of the proposed bootstrapping strategy.**

***Keywords*-Service recommendation; collaborative filtering; bootstrapping; cold start problem**

I. Introduction

The proliferation of Web services forms a new computing platform that facilitates service users in accessing a great variety of computing resources. As many services may compete to offer similar functionalities, service users face unprecedented opportunities to select services that best match their non-functional requirements. The non-functional properties of services are commonly known as Quality of Service (QoS) and typical examples of QoS include response time, availability, reliability, and so on. QoS has been increasingly used as a major criterion in service selection [13], [12]. Matching users with the services that provide the most preferred QoS is nontrivial. Manually testing the candidate services is both time consuming and resource demanding. Service users can be easily inundated with the large number of services.

Collaborative Filtering (CF) has recently been applied in service computing to provide personalized service recommendation [10], [15], [14], [5]. CF works based on the premise that users who have common QoS experiences with some services may share similar experiences with other services. CF has been demonstrated to be more effective than other service selection approaches, which assume that different users receive identical QoS from the same service. Nevertheless, as service users may locate in different network environments and have different physical distances with the service, the QoS they receive from the same service may be distinct. CF explicitly considers user discrepancies and employs similar users' QoS experience to accurately predict the QoS that an active user may receive from previously unknown services.

A key step in CF based service recommendation is to identify users who share similar QoS experience with the active user. The similarity between two users is measured based on the QoS of their commonly invoked services. The quality of the recommendation is tightly coupled with the amount of QoS information the system possesses for a user. For warm-start users, who have invoked a decent number of services, it is relatively easier to locate users that share some common services with them. Hence, similar users are more likely to be identified for them, which is essential for a better recommendation performance. In contrast, for users who are new to the system, since the system possesses very little or no historical QoS information from them, it may fail to provide any recommendation. This is commonly known as the *cold start problem*.

Dealing with cold-start users is a fundamental challenge for recommendation systems. The problem is more severe in service recommendation because most users are either new users or only invoked very limited number of services. An initial interview process can be used to elicit user's information. User profiles are constructed based on the interview results, which will then be used for recommendation. The initial interview process should be both short and intuitive so that a new user won't get bored or lost. Another desirable feature of the interview process is to adaptively query the user based on the results of prior interview questions [3], [8].

Decision trees have been employed to conduct initial interviews to bootstrap e-commerce recommendation systems (e.g., Amazon and Netflix) [4], [17]. A ternary tree is recursively constructed by selecting an item to split existing users assigned to a tree node into its three child nodes along branches, labeled as "like", "dislike", and "unknown", respectively. During the interview, the new user is expected

978-0-7695-4752-7/12 $26.00 © 2012 IEEE
DOI 10.1109/ICWS.2012.90

to rate an item chosen by the decision tree at each given step. Based on the rating, she will be directed to one of the child nodes. The interview continues until the new user is assigned to a leaf node, which represents a homogeneous group of existing users. These users are considered to be as the similar users and will be used to predict the new user's preferences on different items.

Bootstrapping service recommendation poses some new challenges, which hinder a direct application of ternary decision trees. In e-commerce recommendation systems, users' preferences on items are typically represented by few categorical rating values, (e.g., 1-5). This enables a straightforward way to assign users into different groups based on their ratings on an item. For example, the ternary tree assigns a user into the "like" group if her rating is no less than a predefined threshold (e.g., 3) and "dislike" group if otherwise. In contrast, the QoS data used in service recommendation is described by continuous attributes (e.g., 1.2s response time and 0.95 availability). Therefore, dividing users into groups is less intuitive and demands some principled criterion.

Dealing with incomplete QoS data gives rise to the second key challenge. Since an existing user may only invoke a limited number of services, only a small subset of QoS data is observed. The ternary tree introduces the "unknown" tree node to group together users with no ratings on the selected item. Users in this node share a similar opinion on the item, which may be interpreted as either "not know" or "no interest". Correspondingly, no response is also allowed during the interview process, which directs the new user into the "unknown" node. When bootstrapping service recommendation, a new user is expected to invoke a small number of selected services during the interview [1]. Invoking a service always generates a response. Even when the service is down, a "time out" message is returned, indicating that the service is not available. This is different from rating an item (e.g., a product or a movie), which may result in no response when the user does not know or has no interest in the item. In contrast, no response is no longer an option during the interview for service recommendation. If a ternary tree is used, the "unknown" nodes will never be visited during the interview. This in essence ignores a large number of users that are assigned into these nodes and hence dramatically reduces the chance to locate similar users.

We develop a novel strategy to provide high-quality service recommendations for cold-start users. The proposed strategy integrates Matrix Factorization (MF) with decision tree learning to tackle the challenges as highlighted above. MF is employed to partition existing users into a set of user groups. The MF based user partitioning also provides an estimation of the missing entries in historical QoS data. This enables to learn a decision tree from incomplete QoS data without generating "unknown" tree nodes. The user groups obtained by MF will be used as class labels to learn a decision tree. We show that an optimal tree classifies users into groups that possess homogeneous QoS experiences. The decision tree will then be used in the initial interview to adaptively query a new user based on the invocation results on the services in the tree nodes. The new user will be classified into one of the user groups, via which the QoS from services that are unknown to the new user can be predicted.

The remainder of the paper is organized as follows. We review some existing works that are most relevant to ours in Section II. We describe in detail the proposed strategy for bootstrapping service recommendation systems in Section III. We assess the effectiveness of the proposed strategy via a set of experiments on real Web service QoS data in Section IV. We conclude and identify some interesting future directions in Section V.

II. Related Work

The ever increasing number of Web services demands systematic approaches to facilitate service users in efficiently and accurately retrieving services that match both their functional and non-functional requirements. Collaborative Filtering (CF) based techniques have been recently adopted to provide personalized service recommendation to users [10], [15], [14], [5]. Shao et. al. present a service recommendation system by assuming that similar users tend to receive similar QoS from similar services [10]. This is in essence a standard user-based CF algorithm. Zheng et. al. enhance the user-based approach by integrating item-based CF, which results in a hybrid algorithm with better prediction accuracy [15]. Complementary information, such as users' locations [1], [2], invocation frequencies of services [9], and query histories of users [14], has also been leveraged to improve the quality of recommendation.

Both user- and item-based approaches follow the neighborhood centric strategy in CF, which explores the local neighborhood to identify similar users or items for recommendation. Zheng et. al. recently proposed a model based CF algorithm that achieves higher prediction accuracy [16]. The proposed algorithm uses the user-based approach as a precursor to identify top-k similar users. Based on the user neighborhood information, matrix factorization is employed to construct a global model, which can be used to predict unobserved QoS data. Different from our strategy, an unconstrained version of matrix factorization is used, which does not discover the user groups.

All existing service recommendation approaches focus on predicting QoS for warm-start users. To our best knowledge, there is no existing work that provides a systematic approach for cold-start service recommendation. Dealing with cold-start users has received considerable attention in e-commerce

[1]The service invocation code can be wrapped as a small software toolkit that is easily accessed by end users.

recommendation systems. An initial interview has been suggested as an effective way to quickly build new users' profiles. Based on how the seed questions are selected, there are two types of interviews. The first type of interview chooses a static seed set based on some principled criteria, such as coverage [3], popularity [7], and discriminative power [8]. Users always answer a fixed set of questions, which does not fully leverage the interactive nature of the interview process. Recent works propose to adaptively query users based on their responses to the prior interview questions [4], [17], [8]. Decision trees appear as an ideal vehicle to carry out the adaptive initial interview. A ternary tree suits perfectly for the rating-based recommendation systems, which are commonly used in e-commerce. As discussed in Section I, service recommendation poses some new challenges that make a ternary tree inapplicable. In particular, since the "unknown" nodes in a ternary tree will never be visited during the interview, valuable information carried by existing users cannot be fully leveraged to provide high quality recommendations.

III. Cold-start Service Recommendation

We present the bootstrapping strategy for cold-start service recommendation in this section. The cornerstone of the proposed strategy is the integration of Matrix Factorization (MF) and decision tree learning. MF discovers the hidden user group structure from a set of incomplete QoS data that captures the historical user-service interactions. The tree learning algorithm then constructs a decision tree to partition the users to fit the group structure discovered by MF. The simple structure and interpretability of the decision tree serve ideally for an initial interview process, which adaptively queries new users and builds their profiles to provide high-quality service recommendations.

A. MF and Decision Tree Integration

Before delving into the technical details, we first describe the symbols and notations that are used throughout the paper. Assume that there are n existing users and m Web services. The QoS attribute (e.g., response time, reliability, and availability) under consideration takes positive real values. We use a matrix $\mathbf{A} \in \mathbb{R}_+^{m \times n}$ to denote the QoS data, where $\mathbf{A}_{ij}$ represents the QoS that service i delivered to user j. In this regard, the i-th row of $\mathbf{A}$ represents service s_i while the j-th column of $\mathbf{A}$ represents user u_j. This essentially models user u_j as an m-dimensional feature vector, in which each element u_{jq} signifies u_j's interaction with s_q.

1) Matrix Factorization: Matrix factorization computes two low-rank matrices $\mathbf{F} \in \mathbb{R}_+^{m \times k}$ and $\mathbf{G} \in \mathbb{R}_+^{n \times k}$ to approximate the original QoS matrix $\mathbf{A}$, i.e., $\mathbf{A} \approx \mathbf{F}\mathbf{G}'$. More specifically,

$$a_j \approx \sum_{q=1}^{k} \mathbf{G}_{jq} f_q \tag{1}$$

where a_j is the j-th column vector in $\mathbf{A}$, representing user u_j, and f_q is the q-th column of $\mathbf{F}$. Eq. (1) shows that each user vector a_j is approximated by a linear combination of the column vectors in $\mathbf{F}$ weighted by the components of $\mathbf{G}$. In practice, we have $k \ll m$ and $k \ll n$. Hence, $\mathbf{F}$ can be regarded as a new basis that contains much less number of basis vectors than $\mathbf{A}$. These new basis vectors capture the latent features that affect the QoS delivery process. Consequently, $\mathbf{G}$ is the new representation of the service users under the new basis. It can also be regarded as a projection of $\mathbf{A}$ onto the latent feature space $\mathbf{F}$.

The latent feature space $\mathbf{F}$ together with the new representation matrix $\mathbf{G}$ should provide a good approximation of the original QoS data matrix $\mathbf{A}$. Since only a small subset of QoS data is observed, we introduce a weight matrix $\mathbf{W}$, where $\mathbf{W}_{ij} = 1$ if $\mathbf{A}_{ij}$ is observable and $\mathbf{W}_{ij} = 0$ otherwise. Therefore, we compute $\mathbf{F}$ and $\mathbf{G}$ by solving the following optimization problem:

$$\min_{F \geq 0, G \geq 0} J = \left\| \mathbf{W} \circ (\mathbf{A} - \mathbf{F}\mathbf{G}') \right\|^2 \tag{2}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{W}_{ij} \left(\mathbf{A}_{ij} - (\mathbf{F}\mathbf{G}')_{ij} \right)^2 \tag{3}$$

where $\circ$ is component-wise matrix product and $|| \cdot ||$ is matrix norm. Since all the components of $\mathbf{A}$ take non-negative values, we also enforce a non-negative constraint on matrices $\mathbf{F}$ and $\mathbf{G}$. As can be seen from Eq. (1), the non-negative constraint ensures that a user vector is an additive linear combination of the new basis vectors. This allows a more intuitive interpretation than other matrix factorization approaches, such as Singular Value Decomposition (SVD), where negative values are allowed in the matrix components.

2) Decision Tree Learning: Due to its simplicity, interpretability, and the ability to adaptively query users, decision tree becomes an ideal tool to perform the initial interview via which a new user's profile can be constructed. As motivated in Section I, the continuous nature of the QoS attributes and the limited observable QoS data pose key challenges to build a decision tree. The latent feature space discovered via matrix factorization carries rich information that is instrumental to understand the interaction patterns between users and services. It plays a critical role in learning a decision tree from a set of incomplete QoS data. More specifically, the latent feature space enables us to:

- *discover user groups* that contain users sharing similar QoS experiences;
- *estimate the unobserved entries* in the QoS matrix $\mathbf{A}$.

Since the matrix $\mathbf{G}$ is a projection onto the latent feature space, it naturally captures the user group structure. More intuitively, users that share similar latent features should have similar representations in the latent feature space. To make sure that a user is assigned to only one user group (i.e., hard group membership), we enforce a constraint

$\mathbf{G}\mathbf{G}' = \text{diag}(|\mathcal{U}_1|, ..., |\mathcal{U}_k|)$. This makes $\mathbf{G}$ a group indicator matrix:

$$\mathbf{G}_{jq} = \begin{cases} 1 & \text{if } u_j \in \mathcal{U}_q \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $\mathcal{U}_q$ is the q-th user group and $|\mathcal{U}_q|$ denotes the number of users assigned to the group. The constraint ensures that each row of $\mathbf{G}$ has only one non-zero element, which denotes the group that the user is assigned to.

The second key usage of the latent feature space is to estimate the missing QoS entries. If the latent features indeed capture the interaction patterns between users and services, they are expected to provide a good estimation of the unobserved QoS data. More specifically, the QoS that an unknown service s_i will deliver to a user u_j can be estimated as:

$$\mathbf{A}_{ij} \approx \hat{\mathbf{A}}_{ij} = \sum_{p=1}^{k}\sum_{q=1}^{k} \mathbf{F}_{ip}\mathbf{G}_{jq} \quad (5)$$

The j-th column vector of the completed matrix $\hat{\mathbf{A}}$ corresponds to user u_j and the j-th row vector of matrix $\mathbf{G}$ encodes the class (or group) label for the user. The class labels from $\mathbf{G}$ allow us to exploit the classical information gain as the principled criterion to select services to be used as the tree nodes. Using the completed matrix $\hat{\mathbf{A}}$ avoids the generation of "unknown" nodes, which are never visited during the initial interview for service recommendation. Instead of a ternary tree, our tree learning algorithm generates a binary decision tree, via which all existing user information can be leveraged to construct a new user's profile. To ensure a concise interview process, we employ two strategies to control the depth of the tree. First, we stop splitting the a node if the number users assigned to it is less than a predefined threshold value. Second, we exploit a standard pruning process to merge and join leaf nodes after the tree is fully grown.

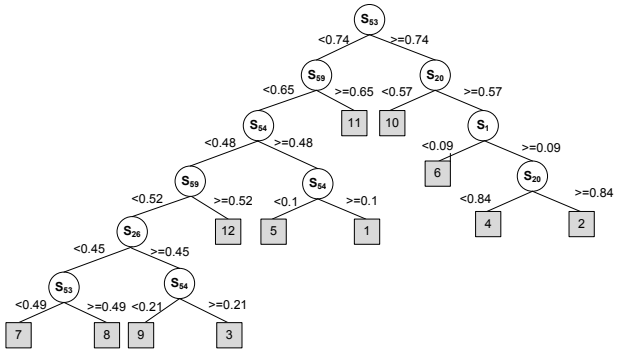


Figure 1. An Example Decision Tree

Figure 1 shows an example decision tree constructed from a real-world QoS dataset obtained from [15]. Each internal node of the tree represents a service. Based on the QoS value, users are directed to one of its child nodes. For example, if the response time that a user received from service s_{53} is less than 0.74 second, she will be directed to child node s_{59}. At this node, the response time of the user will be evaluated against the service in the node. This process continues until the user reaches one of the leaf nodes, which corresponds to one of the user groups.

In what follows, we present an important property of the binary decision tree as constructed by following the above procedure. This helps justify why it can provide high-quality service recommendations for cold-start users.

THEOREM 1: A decision tree that exploits class labels provided by matrix $\mathbf{G}$ partitions users into cohesive user groups, where $\mathbf{G}$ is computed by minimizing objective function J with constraint in Eq. (4).

PROOF. Since each column vector of $\mathbf{A}$ corresponds to a user, we reformulate the objective function J using column vectors of $\mathbf{A}$.

$$J = \sum_{j=1}^{n} \left\| w_j \circ [a_j - \sum_{q=1}^{k} \mathbf{G}_{jq} f_q] \right\|^2 \quad (6)$$

$$= \sum_{j=1}^{n} \left\| \sum_{q=1}^{k} \mathbf{G}_{jq} [w_j \circ (a_j - f_q)] \right\|^2 \quad (7)$$

$$= \sum_{j=1}^{n}\sum_{q=1}^{k} \mathbf{G}_{jq} \left\| w_j \circ (a_j - f_q) \right\|^2 \quad (8)$$

$$= \sum_{q=1}^{k} \sum_{u_j \in \mathcal{U}_q} \left\| w_j \circ (a_j - f_q) \right\|^2 \quad (9)$$

where w_j is the j-th column of $\mathbf{W}$ and $\circ$ is element-wise vector product. Due to Eq. (4) and the fact that one user is assigned to only one group, we have $\sum_{q=1}^{k} \mathbf{G}_{jq} = 1$, which leads Eq. (6) to Eq. (7). From Eq (7) to Eq (8), we use the fact $\mathbf{G}_{jq}^2 = \mathbf{G}_{jq}$ since $\mathbf{G}_{jq} = 1$ or 0. Finally, $\mathbf{G}_{jq} = 1$ only when $u_j \in \mathcal{U}_q$ gives Eq. (9).

Minimizing objective function J is equivalent to find the optimal set $\{(\mathcal{U}_q, f_q) | q \in (1, k)\}$ that minimizes Eq. (9). We know that $\mathcal{U}_q$ denotes the q-th user group and the group membership is encoded by $\mathbf{G}$. If we can find out what the latent feature vector f_d denotes, we are able to interpret what matrix factorization actually achieves under constraint specified in Eq. (4). Since the optimal $\mathbf{F}$ minimizes J, in order to find out $\mathbf{F}$, we take the partial derivative of J with respect to $\mathbf{F}$:

$$\frac{\partial J}{\partial \mathbf{F}} = -2(\mathbf{W} \circ \mathbf{A})\mathbf{G} + 2(\mathbf{W} \circ (\mathbf{F}\mathbf{G}'))\mathbf{G} \quad (10)$$

$$= -2(\mathbf{W} \circ (-\mathbf{A} + \mathbf{F}\mathbf{G}'))\mathbf{G} \quad (11)$$

Setting $\frac{\partial J}{\partial \mathbf{F}} = 0$ gives $-\mathbf{A} + \mathbf{F}\mathbf{G}' = 0$. Multiplying both sides by $\mathbf{G}$ gives $\mathbf{F}\mathbf{G}'\mathbf{G} = \mathbf{A}\mathbf{G}$. Using the fact $\mathbf{G}\mathbf{G}' =$

diag($|\mathcal{U}_1|, ..., |\mathcal{U}_k|$), we get

$$|\mathcal{U}_q| f_q = \sum_{j=1}^{n} \mathbf{G}_{jq} a_j \tag{12}$$

$$f_q = \frac{1}{|\mathcal{U}_q|} \sum_{j=1}^{n} \mathbf{G}_{jq} a_j \tag{13}$$

$$= \frac{1}{|\mathcal{U}_q|} \sum_{u_j \in \mathcal{U}_q} \mathbf{G}_{jq} a_j \tag{14}$$

We exploit Eq. (4) in the last step of derivation.

Eq. (14) reveals that f_q is actually the centroid of the q-th user group. Hence, we conclude that minimizing J with constraint in (4) is equivalent to performing k-means clustering on the existing users. The result is a set of cohesive user groups with minimal total squared deviation from their group means (or centroids). As $\mathbf{G}$ encodes the group memberships, our tree learning algorithm aims to construct a decision tree that partitions users into the same set of cohesive user groups. ■

3) Cold-start Service Recommendation: A cohesive group of users have homogeneous QoS experience. Through a short interview carried out by using the binary decision tree, a new user will be directed into one of the cohesive user groups. The new user hence is expected to share similar QoS experience with other users in the same group. Therefore, the new user should not deviate much from the group mean, which makes the group mean a good estimate for the new user's QoS experience. For example, if a new user u_j is assigned to user group $\mathcal{U}_q$, $f_q \in \mathbb{R}_+^m$, the q-th column vector of $\mathbf{F}$, will be used to estimate the QoS that u_j may receive from m different services. Based on the QoS values, the best services can recommended to the user.

B. Computing $\mathbf{G}$ *and* $\mathbf{F}$

Matrices $\mathbf{G}$ and $\mathbf{F}$ play key roles in both decision tree learning and cold-start service recommendation. $\mathbf{G}$ and $\mathbf{F}$ can be derived by solving the optimization problem in Eq. (2). However, minimizing J under constraint specified by Eq. (4) is non-trivial. Since there is no analytical solution for that, we develop an iterative algorithm to efficiently compute $\mathbf{G}$ and $\mathbf{F}$.

The constraint in Eq. (4) requires binary values on the components of $\mathbf{G}$, which makes the optimization problem unsolvable [11]. To resolve this issue, we instead enforce the following constraint: $\mathbf{G1} = \mathbf{1}$. This is equivalent to enforcing a soft group membership. From $\mathbf{G1} = \mathbf{1}$, we have $\sum_{q=1}^{k} \mathbf{G}_{jq} = 1, \forall j \in [1, n]$. Hence, $\mathbf{G}_{jq}$ can be interpreted as the probability that u_j belongs to group $\mathcal{U}_q$. User u_j will be assigned to group $\mathcal{U}_{\hat{q}}$, where

$$\hat{q} = \arg\max_q \{\mathbf{G}_{jq} | 1 \le q \le k\}$$

We incorporate this new constraint into objective function J as a penalty term, which leads to the following objective function:

$$\min_{F \ge 0, G \ge 0} J(\mathbf{G}, \mathbf{F}) = \|\mathbf{W} \circ (\mathbf{A} - \mathbf{F}\mathbf{G}')\|^2 + \theta ||\mathbf{G1} - \mathbf{1}||^2 \tag{15}$$

In order to minimize $J(\mathbf{G}, \mathbf{F})$, the proposed iterative algorithm updates $\mathbf{G}$ and $\mathbf{F}$ alternatively. That is, while $J(\mathbf{G}, \mathbf{F})$ is minimized with respect to $\mathbf{G}$, $\mathbf{F}$ will be fixed and vice versa. The update of $\mathbf{G}$ and $\mathbf{F}$ is performed by using a set of update rules, which guarantee the convergence of the iterative algorithm. The update rules are derived based on a set of auxiliary functions of objective function $J(\mathbf{G}, \mathbf{F})$, which are formally defined as follows.

DEFINITION 1: *$Z(\mathbf{G}, \tilde{\mathbf{G}})$ is an auxiliary function of function $J(\mathbf{G})$ if it satisfies the following conditions for any $\mathbf{G}$ and $\tilde{\mathbf{G}}$: $Z(\mathbf{G}, \tilde{\mathbf{G}}) \ge J(\mathbf{G})$; $Z(\mathbf{G}, \mathbf{G}) = J(\mathbf{G})$ [6].* ■

Now, let's plug the auxiliary function into our iterative algorithm and see how we can exploit it to derive the update rules. Let $J(\mathbf{G})$ denote the part of $J(\mathbf{G}, \mathbf{F})$ that is only relevant to $\mathbf{G}$. Assume that $\{\mathbf{G}^{(1)}, ..., \mathbf{G}^{(t)}, ...\}$ is a set of matrices obtained by the iterative algorithm, where (t) denotes the t-th iteration. Assume that $\mathbf{G}$ is updated using the following update rule:

$$\mathbf{G}^{(t+1)} = \arg\min_{\mathbf{G}} Z(\mathbf{G}, \mathbf{G}^{(t)}) \tag{16}$$

where $\mathbf{G}^{(t)}$ and $\mathbf{G}^{(t+1)}$ are matrix $\mathbf{G}$ at the t-th and $(t+1)$-th iterations, respectively. It is straightforward to show that $J(\mathbf{G})$ monotonically decreases under update rule in Eq. (16):

$$J(\mathbf{G}^{(t)}) = Z(\mathbf{G}^{(t)}, \mathbf{G}^{(t)}) \ge Z(\mathbf{G}^{(t)}, \mathbf{G}^{(t+1)}) \ge J(\mathbf{G}^{(t+1)})$$

Following the same lines, we can use a similar update rule for $\mathbf{F}$. Since the iterative algorithm updates $\mathbf{G}$ and $\mathbf{F}$ in turn, we have

$$J(\mathbf{F}^{(t)}, \mathbf{G}^{(t)}) \ge J(\mathbf{F}^{(t+1)}, \mathbf{G}^{(t)}) \ge J(\mathbf{F}^{(t+1)}, \mathbf{G}^{(t+1)})$$

As $J(\mathbf{G}, \mathbf{F})$ is apparently lower bounded, it is guaranteed to converge under the above update rules. What remains is to derive the update rules, which requires to find suitable auxiliary functions for $J(\mathbf{G}, \mathbf{F})$ and compute their global minima.

THEOREM 2: Let

$$J(\mathbf{G}) = \|\mathbf{W} \circ (\mathbf{A} - \mathbf{F}\mathbf{G}')\|^2 + \theta ||\mathbf{G1} - \mathbf{1}||^2 \tag{17}$$

The auxiliary function of $J(\mathbf{G})$ is given by

$$Z(\mathbf{G}, \tilde{\mathbf{G}}) = Z_1(\mathbf{G}, \tilde{\mathbf{G}}) + Z_2(\mathbf{G}, \tilde{\mathbf{G}}) \tag{18}$$

where,

$$Z_1(\mathbf{G}, \tilde{\mathbf{G}}) = \sum_{ij} \mathbf{W}_{ij} \left[\mathbf{A}_{ij}^2 - 2 \sum_q \mathbf{A}_{ij} \mathbf{F}_{iq} \tilde{\mathbf{G}}_{jq} \left(1 + \log \frac{\mathbf{G}_{jq}}{\tilde{\mathbf{G}}_{jq}} \right) + \sum_q [\mathbf{F}\tilde{\mathbf{G}}']_{ij} \mathbf{F}_{iq} \frac{\mathbf{G}_{jq}^2}{\tilde{\mathbf{G}}_{jq}} \right] \tag{19}$$

$$Z_2(\mathbf{G},\tilde{\mathbf{G}}) = \theta \sum_{jq} \left([\tilde{\mathbf{G}}\mathbf{1}]_j \frac{\mathbf{G}_{jq}^2}{\tilde{\mathbf{G}}_{jq}} \right) - \theta \sum_{ip} 2\tilde{\mathbf{G}}_{jq} \left(1 + \log \frac{\mathbf{G}_{jq}}{\tilde{\mathbf{G}}_{jq}} \right) + n\theta \quad (20)$$

The global minimum of $Z(\mathbf{G},\tilde{\mathbf{G}})$ is

$$\mathbf{G}_{jq} = \tilde{\mathbf{G}}_{jq} \left[\frac{[(\mathbf{W}\circ\mathbf{A})'\mathbf{F}]_{jq} + \theta}{[(\mathbf{W}\circ(\mathbf{F}\tilde{\mathbf{G}}'))'\mathbf{F} + \theta\tilde{\mathbf{G}}\mathbf{E}]_{jq}} \right]^{\frac{1}{2}} \quad (21)$$

PROOF SKETCH:

It is straightforward to show that $Z(\mathbf{G},\mathbf{G}) = J(\mathbf{G})$. Furthermore, $J(\mathbf{G})$ has two quadratic terms. Applying Jensen's inequality and inequality $x \geq 1 + \log x, \forall x > 0$ when expanding both terms, we get

$$\|\mathbf{W}\circ(\mathbf{A} - \mathbf{F}\mathbf{G}')\|^2 \leq Z_1(\mathbf{G},\tilde{\mathbf{G}}) \quad (22)$$

$$\theta\|\mathbf{G}\mathbf{1} - \mathbf{1}\|^2 \leq Z_2(\mathbf{G},\tilde{\mathbf{G}}) \quad (23)$$

From Eqs. (22) and (23), we have $Z(\mathbf{G},\tilde{\mathbf{G}}) \geq J(\mathbf{G})$. Hence, we prove that $Z(\mathbf{G},\tilde{\mathbf{G}})$ is an auxiliary function of $J(\mathbf{G})$.

To show that Eq. (21) gives the global minimum of $Z(\mathbf{G},\tilde{\mathbf{G}})$, we need to first prove that $Z(\mathbf{G},\tilde{\mathbf{G}})$ indeed has a global minimum. This can be achieved by showing that $Z(\mathbf{G},\tilde{\mathbf{G}})$ is a convex function on $\mathbf{G}$. We compute the second order derivative of $Z(\mathbf{G},\tilde{\mathbf{G}})$ with respect to $\mathbf{G}$, which gives the Hessian matrix of $Z(\mathbf{G},\tilde{\mathbf{G}})$:

$$\frac{\partial^2 Z(\mathbf{G},\tilde{\mathbf{G}})}{\partial\mathbf{G}_{jq}\partial\mathbf{G}_{pr}} = \delta_{jp}\delta_{qr} \left(\frac{2[\mathbf{W}\circ\mathbf{A}\mathbf{F}]_{jq}\tilde{\mathbf{G}}_{jq} + 2\theta\tilde{\mathbf{G}}_{jq}}{\mathbf{G}_{jq}^2} + \frac{2[\mathbf{W}\circ(\mathbf{F}\tilde{\mathbf{G}}')\mathbf{F}]_{jq} + 2\theta[\tilde{\mathbf{G}}\mathbf{E}]_{jq}}{\tilde{\mathbf{G}}_{jq}} \right)$$

where $\delta_{ab} = 1$ when $a = b$ and 0 otherwise. Hence, the Hessian is a diagonal matrix with positive diagonal elements, which makes it positive definite. Therefore, $Z(\mathbf{G},\tilde{\mathbf{G}})$ is a convex function on $\mathbf{G}$. To compute the global minimum, it is sufficient to compute its local minimum. We set $\frac{\partial Z(\mathbf{G},\tilde{\mathbf{G}})}{\partial\mathbf{G}_{jq}} = 0$ and through some algebra, we get Eq. (21). ■

Following the same lines, we can derive the update rule for $\mathbf{F}$:

$$\mathbf{F}_{iq} = \tilde{\mathbf{F}}_{iq} \left[\frac{[\mathbf{W}\circ\mathbf{A}\mathbf{G}']_{iq}}{[\mathbf{W}\circ(\tilde{\mathbf{F}}\mathbf{G}')\mathbf{G}']_{iq}} \right]^{\frac{1}{2}} \quad (24)$$

Having update rules (21) and (24), the iterative algorithm essentially updates $\mathbf{F}$ and $\mathbf{G}$ alternatively in each iteration. The algorithm continues until it converges or a predefined number of iterations is reached.

Consider the following matrix $\mathbf{A}$, which stores a subset of response times extracted from a real-world QoS dataset [15]. Each row of $\mathbf{A}$ denotes a user, each column denotes a service, and some response times are missing (denoted as '-').

$$\mathbf{A} = \begin{pmatrix} 0.69 & 1.26 & 0.42 & 0.64 \\ 0.61 & - & 0.31 & 0.65 \\ 0.24 & 1.86 & 0.66 & 1.04 \\ 0.19 & 1.68 & - & 1.11 \\ - & - & 0.67 & 1.15 \end{pmatrix}$$

Following the above procedure, we get

$$\mathbf{F} = \begin{pmatrix} 0.14 & 3.5 \\ 1.13 & 3.07 \\ 0.40 & 0.94 \\ 0.69 & 1.51 \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} 0.27 & 0.43 \\ 0.23 & 0.41 \\ 0.62 & 0.15 \\ 0.60 & 0.10 \\ 0.55 & 0.39 \end{pmatrix}$$

The user group membership is clearly captured by matrix $\mathbf{G}$. More specifically, the first two users (i.e., the first two rows of $\mathbf{A}$) belong to the first user group. This is due to the fact that in the first two rows of $\mathbf{G}$, the second column is larger than the first column. Similarly, the last three users belong to the second user group.

The time complexity of the above procedure is dominated by the two update rules (i.e., Eq. (21) and Eq. (24)), which essentially perform a number of matrix multiplications. Assuming that the procedure takes t iterations to converge, thus the overall time complexity is $O(mnkt)$.

IV. EXPERIMENTS

We carry out a set of experiments to evaluate the effectiveness of the proposed strategy for cold-start service recommendation. The experiments are conduced on a real-world QoS dataset that consists of 1.5 million service invocation records. 150 computer nodes from the Planet-Lab [2], which are located in over twenty countries, are leveraged to automatically invoke a hundred selected Web services. These services are distributed across more than twenty countries. Each computer node invokes each service for 100 times and the average Round-Trip Time (RTT) is used as the QoS dataset in our experiments.

A. Experiment Design

We organize the QoS data into a 100×150 matrix $\mathbf{A}$, in which entry $\mathbf{A}_{ij}$ denotes the averaged RTT that user j used to invoke service i. We randomly remove a certain percentage (80%–96%) of entries from $\mathbf{A}$ to simulate a real-world QoS dataset, where only a small subset of entries are observed. To assess the proposed bootstrapping strategy, we follow a similar design as in [17], which splits users (i.e., columns in $\mathbf{A}$) into two disjoint subsets: the training set and the test set, consisting of 80% and 20% users, respectively. We apply MF to the training set to discover the user groups and estimate the missing QoS entires. We then construct the decision tree for the initial interview. The actual RTT records

[2]http://www.planet-lab.org

of the test users are used to simulate the results of invoking the services in the decision tree.

We employ Mean Absolute Error (MAE), one of the most widely used metric in recommendation systems, to assess the quality of recommendation:

$$MAE = \sum_{i,j} \frac{|\mathbf{A}_{ij} - \hat{\mathbf{A}}_{ij}|}{N} \tag{25}$$

where $\mathbf{A}_{ij}$ and $\hat{\mathbf{A}}_{ij}$ denote the actual and estimated RTT respectively. N is the total number of estimated QoS entries. Since the RTT entries are randomly removed, all the results reported below are obtained by computing the average over 20 runs. The default value for the number of user groups is 30 and the default value for the penalty term θ is 10. These default values will be used in all the experiments unless specified otherwise.

B. Quality of Cold-start Recommendation

To our best knowledge, there is no existing work on providing service recommendation for cold-start users. As discussed in Section I, the ternary tree approach presented in [4], [17] is not suitable for the initial interview of service recommendation, either. To demonstrate the effectiveness of the proposed bootstrapping strategy, we implemented four representative collaborative filtering methods, including:

- the user based algorithms using both Pearson Correlation Coefficient (UPCC) and cosine similarity (referred to as UCOS) as similarity measures [10];
- the item based algorithm using Pearson Correlation Coefficient (IPCC) as similarity measure;
- the hybrid collaborative algorithm that combines both user and item based approaches using their prediction accuracy as the aggregation weights (referred to as WSRec) [15].
- the constrained matrix factorization model (referred to as MF), as discussed in Section III, in which Eq. (5) is utilized to make the prediction after the model is constructed.

We apply the above methods to the warm-start users and use the obtained result as the baseline to assess our cold-start performance. As indicated in [17], using a ternary tree model, the cold-start performance is always worse than the warm-start performance considering that more information is available for the warm-start users. Therefore, the relative warm/cold start performance is a good indicator about the effectiveness of the bootstrapping process.

Figure 2(a) compares the warm-start performance from the four representative CF algorithms with the cold-start performance of the proposed bootstrapping strategy (referred to as MF-DT). We vary the sparsity ratio of the QoS matrix $\mathbf{A}$ from 80% to 96% and achieve two important observations. First, the cold-start performance of MF-DT outperforms the warm-start performance of other algorithms in all cases. This clearly demonstrates the effectiveness of the bootstrapping strategy. As can be seen later in Figure 2(c), a new user only needs to invoke few services (3-6 on average) during the interview process. This is usually much smaller than the number of services invoked by a warm-start user. For example, if the sparsity of $\mathbf{A}$ is 80%, since we have 100 services in total, each existing user invoked 20 services on average. The fundamental reason for this is that the integration of MF with decision tree learning identifies the most important few services to invoke for the new user. The QoS collected from these services captures the key (latent) features of the user. This is critical to discover the most similar user group, which is used to predict the QoS from other services that are unknown to the new user.

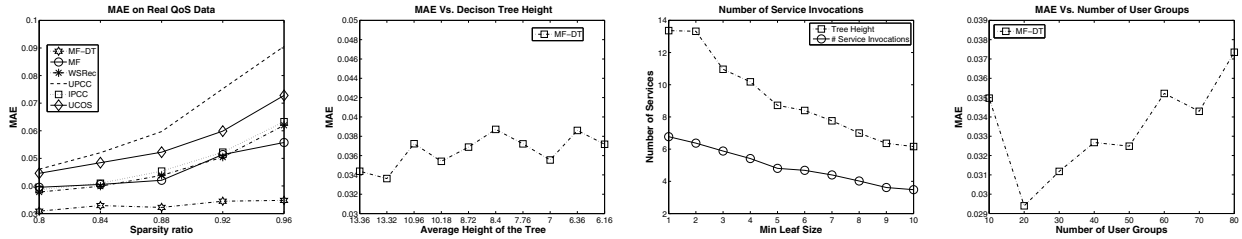
Second, as the sparsity of $\mathbf{A}$ increases, the performance advantage of MF-DT becomes more significant. This is because the warm-start performance drops quickly when less information is available for users. For a very sparse QoS dataset, most users may invoke very few or even zero services. In fact, these algorithms essentially suffer from the cold-start problem, which we aim to resolve in this paper. The MAE performance of MF-DF also drops as sparsity increases because it relies on the similar user groups to make the prediction. However, the performance goes down much slower than other algorithms. It is also interesting to note that MF suffers less than other algorithms for the cold-start issue. This also contributes to the good performance of MF-DT, in which MF serves as a precursor of the entire bootstrapping process.

C. Impact of Parameters

We investigate the impact of two important parameters in this section, including the height of the decision tree and the number of user groups. The sparsity ratio of $\mathbf{A}$ is kept as 80%.

We control the height of the decision tree by restricting the minimum number of services per leaf node, referred to as min_leaf_size. As we vary min_leaf_size from 1 to 10, the average tree height decreases from 13.36 to 6.16 (see Figure 2(b)). The MAE performance also decreases slowly as tree height decreases although there are some small fluctuations due to the randomness in removing entries from $\mathbf{A}$ and the initialization of $\mathbf{F}$ and $\mathbf{G}$.

A service may appear multiple times in the decision tree. For example, in Figure 1, services S_{53}, S_{59} and S_{54} all appear more than one times in the example decision tree. Therefore, the number of services that need to be invoked by a new user during the interview is actually much smaller than the tree height. Figure 2(c) reports the average number of service invocations versus the min_leaf_size, which confirms our hypothesis. It is obvious that users only need to invoke a very small number (3-6 on average) of services to achieve good cold-start recommendation performance.



(a) MAE Performance Comparison (b) Impact of Decision Tree Height (c) Number of Service Invocations (d) Number of User Groups

Figure 2. Experimental Results

Figure 2(d) shows the impact of the number of user groups. An optimal MAE performance is reached when the number of groups is 20. As the number of groups further increases, many smaller groups will be generated. Restricted by the group size, similar users may be spit into different groups, which will lower the prediction accuracy.

V. Conclusion and Future Work

We develop a novel bootstrapping strategy to provide high-quality service recommendations for cold-start service users. The proposed strategy integrates constrained matrix factorization with decision tree learning, which enables a fast and adaptive initial interview to effectively elicit information from new users. The constrained matrix factorization discovers homogeneous user groups and estimates the unobserved QoS entries. The tree learning algorithm then leverages these estimated information and exploits user groups as class labels to learn a decision tree for the interview process. Experimental results on real-world QoS data clearly justify the effectiveness of the proposed bootstrapping strategy. In this work, we primarily focus on providing service recommendations for cold-start users. One interesting future direction is to deal with *cold-start services*, which are new services that have not been invoked by any users.

References

[1] X. Chen, X. Liu, Z. Huang, and H. Sun. Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In *ICWS*, pages 9–16, 2010.

[2] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun. Personalized qos-aware web service recommendation and visualization. *IEEE Transactions on Services Computing*, 99(PrePrints), 2011.

[3] N. Golbandi, Y. Koren, and R. Lempel. On bootstrapping recommender systems. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1805–1808, New York, NY, USA, 2010. ACM.

[4] N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 595–604, New York, NY, USA, 2011. ACM.

[5] Y. Jiang, J. Liu, M. Tang, and X. F. Liu. An effective web service recommendation method based on personalized collaborative filtering. In *ICWS*, pages 211–218, 2011.

[6] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[7] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, IUI '02, pages 127–134, New York, NY, USA, 2002. ACM.

[8] A. M. Rashid, G. Karypis, and J. Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *SIGKDD Explor. Newsl.*, 10:90–100, December 2008.

[9] W. Rong, K. Liu, and L. Liang. Personalized web service ranking via user group combining association rule. *Web Services, IEEE International Conference on*, 0:445–452, 2009.

[10] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei. Personalized qos prediction forweb services via collaborative filtering. In *ICWS*, pages 439–446, 2007.

[11] F. Wang, T. Li, and C. Zhang. Semi-supervised clustering via matrix factorization. In *SDM*, pages 1–12, 2008.

[12] Q. Yu and A. Bouguettaya. Framework for web service query algebra and optimization. *TWEB*, 2(1), 2008.

[13] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web*, 1(1):6, 2007.

[14] Q. Zhang, C. Ding, and C.-H. Chi. Collaborative filtering based service ranking using invocation histories. In *ICWS*, pages 195–202, 2011.

[15] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Wsrec: A collaborative filtering based web service recommender system. In *ICWS*, pages 437–444, 2009.

[16] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Collaborative web service qos prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing*, 99(PrePrints), 2011.

[17] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 315–324, New York, NY, USA, 2011. ACM.