Knowl Inf Syst (2015) 43:417–443
DOI 10.1007/s10115-013-0723-x

REGULAR PAPER

CloudRec: a framework for personalized service Recommendation in the Cloud

Qi Yu

Received: 8 February 2013 / Revised: 30 September 2013 / Accepted: 04 December 2013 /
Published online: 9 January 2014

Abstract The elastic computing power and the pay-as-you-go model of the cloud offer an attractive platform to deploy software as a service applications. The large number of applications expected to heavily take advantage of the cloud will result in an explosive growth of various cloud services. As many cloud services may compete to offer similar functionalities, it is desirable to consider user preferences on the nonfunctional service properties (aka, quality of service, or QoS) when delivering cloud services to the end users. Unfortunately, current approaches primarily rely on the descriptions from the cloud service providers or expert-provided rankings, which are completely orthogonal to the open and distributed nature of the cloud. We present a novel framework (referred to as *CloudRec*) that exploits a user-centric strategy to achieve personalized QoS assessment of cloud services. *CloudRec* integrates a novel community-based QoS assessment model with an iterative algorithm to accurately discover a set of homogenous user and service communities from scarce and large-scale QoS data. The communities can serve as a bridge to relate users and services and hence provide an effective means to estimate the QoS of unknown cloud services. The effectiveness of the proposed framework is demonstrated through a rigorous theoretical analysis and an extensive empirical study on real QoS data.

Keywords Cloud computing · QoS · Service recommendation · Matrix factorization

1 Introduction

Cloud computing offers an attractive paradigm for the provisioning of computing resources across a wide spectrum of domains. The large number of applications expected to heavily take advantage of the cloud will lead to the deployment of substantial cloud services [45]. Hence, a fundamental task is to help users select their desired cloud services from an open

Q. Yu (✉)
College of Computing and Information Sciences,
Rochester Institute of Tech, Rochester, NY, USA
e-mail: qi.yu@rit.edu

and dynamic cloud environment. Existing proposals that can be applied to cloud service selection fall into two categories that are complementary to each other: functionality-based and QoS-based. The former primarily focuses on the discovery of cloud services that satisfy user required functionality (e.g., map, weather, and traffic). Data mining techniques [20] and semantic support (e.g., using ontologies) [32] have been exploited to improve the accuracy of service discovery. QoS-based approaches, on the other hand, are used to differentiate service providers based on their QoS performance [39,40]. The QoS is mainly made of user-centered quality parameters and examples include availability, reliability, and response time. QoS-aware service selection is critical in the cloud as the large number of services will inevitably incur the competition among service providers that offer similar functionality. As an example, when searching Map using webservices.seekda.com, more than 1,000 results are returned. Selecting services solely based on their functionality may result in services with undesirable QoS. The expected large number of competing cloud services and the lack of technical skills of casual users demand a principled approach for assessing the QoS of cloud services.

Accurate QoS assessment of cloud services is further complicated by a set of cloud-specific characteristics that are embedded in user–service interactions. The elastic provisioning of cloud resources is rooted in a distributed infrastructure instead of a few isolated powerful machines [2]. *Virtualization* makes everything transparent, where users' requests are handled by a number of virtual machines that share the computing power underneath. Thus, cloud services essentially run on multiple server instances that are dynamically assigned. Despite that multiple virtual machines can share resources such as CPUs and main memory fairly well, the I/O sharing seems to pose some problems [3]. The unequal sharing of the computing power will result in unpredictable performance of the cloud services which will affect the QoS delivered to the users. On the other hand, cloud users may be located in different network environments, using different development tools, and having different physical distances with the cloud services they access [45]. These factors determine that the QoS (e.g., response time and reliability) delivered to cloud service users will be disparate. The *discrepancy* between cloud service users and the *unpredictable performance* of cloud providers imply that users may indeed receive significantly different QoS from the same cloud service.

Unfortunately, current QoS assessment approaches primarily rely on the QoS descriptions from the cloud service providers or expert-provided rankings. Some monitoring mechanisms have also been exploited to collect the QoS data, which is typically presented as averaged values computed from past transactions with service users. Thus, these approaches are completely orthogonal to the open and distributed nature of the cloud environment, which makes them incapable of handling the cloud-specific characteristics as described above. As a result, these approaches will lead to QoS assessment that may be dramatically different from what is actually received by the users.

Thus, in this paper, we propose a *user-centric framework*, referred to as *CloudRec*, which takes into account cloud users' QoS experiences to achieve *personalized* assessment of cloud services. The ability to provide personalized QoS assessment is crucial to recommending suitable cloud services to users due to the inherent characteristics of the cloud environment (i.e., user discrepancy and performance unpredictability). The premise of the user-centric strategy is that users who share similar historical experiences have some *latent* (cloud-related) features in common (e.g., network quality, development environment, and physical distance to the server instances), which implies a similar behavior among these users in future. Similarly, cloud services that deliver similar QoS to users also share some common *latent* (cloud-related) features (e.g., virtual machine performance). A central ingredient of the proposed *CloudRec*

framework is a set of user and service *communities* that group together users and services sharing similar latent (cloud-related) features. The communities serve as a bridge to relate users and services and hence provide an effective means to estimate the QoS of unknown cloud services.

To accurately construct the user and service communities, we draw upon and significantly extend the recently developed matrix factorization-based clustering approach. More specifically, the user and service communities are constructed through matrix tri-factorization that essentially performs "co-clustering" on both users and services [17]. In this way, we not only fully leverage the QoS data obtained from the historical user-service interactions, but also take into account the embedded "duality" relationship between services and users [15]. Co-clustering has been shown to be more effective than "one-side" clustering in mining the fast increasing dyadic data (e.g., user and service, and document and term). However, most existing co-clustering algorithms adopt a graph-based model that requires solving expensive eigenproblems [11]. In contrast, matrix factorization-based clustering is computationally more efficient [11,23]. Furthermore, its workload can be easily parallelized and deployed to a distributed infrastructure [30], like MapReduce [13]. In this way, it is capable of handling extremely large-scale datasets, which is imperative for a cloud environment.

Nevertheless, there are *three central challenges* that hinder us from directly applying any existing matrix factorization-based approach to construct user and service communities. In the sequel, we describe each of these challenges along with the proposed strategies that tackle these challenges.

1.1 Challenge 1

Directly applying matrix factorization without any constraints will result in cluster[1] indicator matrices whose entries can take arbitrary values. Thus, no stable community structure will be obtained. To attack this issue, we propose a *principled constraint* on matrix factorization to ensure proper cluster assignment. Entries in the cluster indicator matrices under the constrained matrix factorization can be directly interpreted as the cluster posterior probability.

1.2 Challenge 2

Since a typical user may have only used a very limited number of services, the QoS data obtained from historical user–service interactions are usually very limited. Thus, it is necessary to learn from other types of information available in the cloud environment. We propose to leverage the *geometric structures* of user and service distributions and integrate them with the constrained matrix factorization to improve the overall clustering accuracy. The key premise is that users or services that are "close" to each other in their respective geometric structures should have similar community memberships [4].

1.3 Challenge 3

Existing matrix factorization-based clustering approaches are only applicable to complete matrices. However, we need to construct user and service communities from an incomplete QoS matrix with many missing entries. In fact, the QoS matrix being incomplete is exactly why we propose the user-centric assessment strategy, which aims to help users estimate the QoS of an unknown cloud service so that they can choose the best one to use. A straightforward extension of existing approaches is to leverage a weighting mechanism that only

[1] We use community and cluster in an exchangeable manner in the rest of the paper.

uses observable QoS entries [23,41]. A major limitation with this approach is that it solely relies on the information carried by the available QoS data, which may be very limited. We believe that the community information obtained during the community learning process is useful for estimating the missing QoS entries. Based on this intuition, we propose *an iterative algorithm* that amalgamates community construction and QoS assessment into an integrated process. This algorithm aims to leverage both the available QoS data and the community information obtained during the community construction process. That is, the community information obtained during the community construction process will be immediately used to estimate the missing entries; the estimated missing entries will in turn be used to refine the communities. In this regard, the communities and the missing entries will keep being refined in an iterative fashion until a (local) optimal is achieved.

1.4 Summary of contributions

We now summarize our key contributions as follows:

1. We propose a novel community-based QoS assessment model that augments matrix factorization with a principled constraint and the geometric structures of user and service distributions (Sect. 3).
2. We develop an iterative algorithm, which amalgamates community construction and QoS assessment into an integrated, mutually refining process (Sect. 4).
3. We prove the correctness and convergence of the proposed algorithm through a rigorous theoretical analysis (Sect. 5).
4. We conduct an empirical study on real QoS data to demonstrate the effectiveness of the proposed model and algorithm (Sect. 6).

2 Related work

In this section, we provide a detailed review of existing works that are most relevant to our proposed *CloudRec* framework.

2.1 Cloud service and service selection

Cloud computing has attracted significant attention from both industry and academia. Leading cloud vendors have provided different types of clouds, including infrastructure as a service (IaaS) clouds, platform as a service (PaaS) clouds, and software as a service (SaaS) clouds. The cloud computing market is expected to reach $270 billion by 2020 [1]. Important research prototypes are also being developed in academia (e.g., epiC [9]) that moves key operations such as OLTP and OLAP into the cloud [8,38]. Therefore, we anticipate the deployment of a large number of cloud services and the competition between multiple cloud services that offer similar functionalities. Selecting services with user-desired QoS from a large number of competing service providers has received considerable attention from the service computing community [39,40]. A number of cloud service selection techniques have also recently been developed to help user choose the best cloud service from a large number of candidates [12,24, 33,37]. However, most approaches primarily rely on the QoS information provided by service providers or service registries. More importantly, they do not consider the discrepancies between different users and services and assume that different users will receive identical QoS from same services.

2.2 Collaborative-filtering-based recommendation

Collaborative filtering (CF) has emerged as a key technology in e-commerce and online content distribution [5,7,21,25,26]. CF exploits the similarity between users' experiences to predict user preference on unknown items. The intuitive idea is to identify "similar" users with the active user and predict the active user's preference based on these similar users' feedbacks. The *similarity* between two users is measured using the feedbacks on the common items. Most existing CF approaches fall into two categories: neighborhood-based and model-based. The neighborhood-based approaches suffer from the data scarcity issue that arises in practice because a typical user may only provide feedbacks for a limited number of items. This is even more serious in the cloud considering the large number of users and cloud services. Users have to invoke at least two common cloud services in order to be considered as similar. Model-based CF approaches alleviate feedback scarcity by generating a global model based on the given training data and using the model to predict user preference on unknown items. Typical models include aspect models [26], latent factor models [7], Bayesian models [42], and decision trees [5]. A major issue with the existing model-based approaches is their high computational overhead which is caused by the tuning of a large number of parameters embedded in the models. This makes these models inapplicable to large-scale datasets, which are typical for a cloud environment.

The proposed community-based QoS assessment model provides an effective means to address data scarcity. More specifically, two users can be related by having invoked the services in the same service community instead of having invoked the same service. Similarly, two services can be related through users from the same user community instead of having used by the same user. Our experimental results on real QoS data also clearly demonstrate the effectiveness of the community-based approach in handling very sparse datasets. Furthermore, using matrix factorization, the communities can be more efficiently constructed [23] and the workload can be easily parallelized to handle very large-scale QoS data [30].

CF has also been adopted for service selection in service computing community [35,44]. An augmented neighborhood-based approach is presented in [10] for personalized Web service recommendation. A region model is constructed by explicitly integrating users' geographical locations. Nevertheless, there is no direct relationship between the physical distances of users and services and QoS delivery. For instance, a remote user may get a fast response if s/he is connected through a high-speed network.

2.3 Matrix factorization-based clustering

Matrix factorization-based approaches have been increasingly adopted for data clustering and co-clustering due to their effectiveness in dealing with high-dimensional datasets. The SVD- or eigenvector-based approaches are commonly used for matrix decomposition to generate data clusters [14,19,36]. The basic idea is to project the original data space into a *latent semantic space*, which is represented by singular vectors or eigenvectors. As singular vectors or eigenvectors do not directly correspond to the individual clusters, traditional clustering algorithms (e.g., K-means) need to be applied to generate the final clusters. Due to the existence of the latent semantic space, it is usually difficult to interpret the clustering result. To address this issue, nonnegative matrix factorization (NMF) techniques have been recently applied to clustering with the benefit of providing an intuitive interpretation for the clustering result [17,27,28]. It has been demonstrated that, under certain constraints, NMF clustering is equivalent to some widely used clustering schemes, including k-means clustering [17], spectral clustering [18], and probabilistic latent semantic indexing [29].

NMF has also been exploited in recommendation systems to predict user ratings on unknown items. In [41], a rating matrix A is factorized as UV, where each column vector $U^{(i)} \in U$ can be regarded as a user-cluster centroid and each column vector $V^{(j)} \in V$ can be regarded as user j's affinities for all user communities. In this regard, only users are clustered in the proposed framework, which is different from the co-clustering scheme used in our approach. In [23], nonnegative matrix tri-factorization (NMTF) is exploited to co-cluster users and items. However, NMTF is not constrained. Therefore, it may lead to arbitrary cluster assignment and the result does not offer any intuitive interpretation. In addition, a weighting mechanism is adopted that minimizes a new objective function $||W \odot (F - URS^T)||_F^2$, where $\odot$ is Hadamard product (i.e., element-wise product), W_{ij} is set to one if F_{ij} is an observed entry and zero otherwise. As discussed in "Introduction," this approach only relies on the information carried by the observed entries, which may be very limited.

3 The QoS assessment model

In this section, we first describe the basic community-based QoS assessment model. We then present the principled constraint on matrix factorization and the integration of geometric structures of user and service distributions as two key extensions of the basic model.

3.1 The basic model

The QoS assessment model is built around two sets of objects: users $\mathcal{U} = \{u_1, \ldots, u_n\}$ and cloud services $\mathcal{S} = \{s_1, \ldots, s_m\}$. A *dyad* is a scalar value $f(u, s)$ that is used to represent the relationship between user u and service s, where $u \in \mathcal{U}$ and $s \in \mathcal{S}$. This type of data is usually known as *dyadic* data, which can be represented as an n-by-m two-dimensional matrix F if we map the row indices into $\mathcal{U}$ and the column indices into $\mathcal{S}$ [31]. Each entry F_{ij} represents the QoS that service s_j delivered to user u_i.

The premise behind the proposed model is that there exists a small number of latent factors that influence users' perception on the QoS delivered by the cloud services. Users and services that share similar values on these latent factors can be assigned to a limited number of user and service communities. We can then leverage these communities to predict users' perception on QoS delivered by a priori unknown cloud services. In particular, we propose to use NMTF to find a low-rank matrix Y as an approximation of the original matrix F, i.e., $F \approx Y$, where Y can be factorized as $Y = URS^T$. More specifically, $U \in \mathbb{R}^{n \times k}$ is the cluster indicator matrix for clustering users (i.e., rows of F), $S \in \mathbb{R}^{m \times l}$ is the cluster indicator matrix for clustering services (i.e., columns of F), $R \in \mathbb{R}^{k \times l}$ is the cluster association matrix that captures the relationship between user clusters and service clusters. Hence, NMTF essentially simultaneously clusters $\mathcal{U}$ into k disjoint user communities $(\hat{u_1}, \ldots, \hat{u}_k)$ and $\mathcal{S}$ into l disjoint service communities $(\hat{s_1}, \ldots, \hat{s}_l)$. Considering the dyadic nature of the data space, co-clustering can effectively exploit the *duality* between rows and columns to improve the clustering accuracy.

Let us now illustrate how the model can be used to derive user u_i's perception on the QoS delivered by an unknown cloud service s_j. Recall that matrix R is the cluster association matrix, where each entry R_{pq} essentially captures the QoS perception of user community $\hat{u}_p$ on service community $\hat{s}_q$. Since S is the service-cluster indicator matrix, S_{qj}^T is the cluster coefficient of service s_j on service community $\hat{s}_q$. Thus, the QoS perception of user community $\hat{u}_p$ on service s_j can be formulated as:

$$(RS^T)_{pj} = \sum_{q=1}^{l} R_{pq} S_{qj}^T \tag{1}$$

We can view $RS^T \in \mathbb{R}^{k \times m}$ as the matrix that contains the basis of the user space $\mathcal{U}$, where each entry $(RS^T)_{pj}$ captures the perception user community $\hat{u}_p$ on cloud services s_j. Since U_{ip} is the cluster coefficient of user u_i on user community $\hat{u}_p$, u_i's QoS perception on s_j can be formulated as:

$$F_{ij} \approx Y_{ij} = \sum_{p=1}^{k} U_{ip} (RS^T)_{pj} = \sum_{p=1}^{k} U_{ip} \left(\sum_{q=1}^{l} R_{pq} S_{qj}^T \right) \tag{2}$$

Equation (2) reveals that u_i's QoS perception on s_j is approximated by a linear combination of all user communities' QoS perception on s_j, weighted by the cluster coefficients of u_i on these communities. More generally, each row vector $\mathbf{f}_i^T$ of F can be approximated as a linear combination of rows in the basis matrix RS^T, i.e.,

$$\mathbf{f}_i^T = \sum_{p=1}^{k} U_{ip} \mathbf{v}_p^T \tag{3}$$

where $\mathbf{v}_p^T$ is a row vector in the basis matrix RS^T. Therefore, $\mathbf{u}_i^T$, which is the i-th row of U, can be regarded as the new representation of the i-th user in the new basis RS^T. Since we have $k \ll n$ and $l \ll m$ in practice, we essentially use a small number of basis vectors to represent a large number of user vectors. When the basis vectors can capture the intrinsic latent structure of the data space, a good approximation can be achieved [28].

3.1.1 A scenario: setting up a cloud-based enterprise

In what follows, we describe a scenario to further illustrate the community-based model.

Consider the development of a cloud-based enterprise, TravelAssistant, which provides travel assistance services for users, including map, weather, flight, and local attractions. The most cost effective way to set up TravelAssistant is to mash up existing cloud resources to create all functional layers of the enterprise, including data storage, system softwares, and software services. Resources, such as storage and system softwares, are typically obtained from large data centers, such as Amazon, Microsoft, and Google. The developer will face much more choices in selecting software services as many software vendors expect to take the key advantages of the cloud to make their services more attractive. For example, by deploying a Point Of Interest (POI) service in the cloud, the POI service will be equipped with the elastic computing power to handle the increased demand in the tourist season.

Since TravelAssistant is expected to be highly available and response fast to users' requests, the challenge is to select cloud services that meet these QoS requirements. Assume that the Map, Weather and FlightStats services have already been chosen as the developer has used these three services and perceived satisfactory QoS. The remaining task now is to select a POI service with the desired QoS (i.e., high availability and fast response time).

Assume that the QoS of the cloud services can be derived from the historical user–service interactions (e.g., the transaction logs of these services). Table 1 shows the response times received by six different users (including the developer) when interacting with the component services of TravelAssistant and two candidate POI services. As the users may not interact with all the services, n/a in Table 1 signifies that the user has not used the corresponding service

Table 1 The user received response times

Users	Map	Weather	FlightStats	POI_1	POI_2
U_1 (i.e., developer)	3.072	2.24	1.984	n/a	n/a
U_2	2.688	2.56	1.216	1.472	2.752
U_3	3.008	2.88	n/a	1.984	2.24
U_4	1.536	1.024	2.56	2.048	1.664
U_5	1.856	1.472	2.88	2.496	n/a
U_6	1.664	1.856	2.688	n/a	1.216

and hence no response time is provided. Since NMTF is not applicable to an incomplete matrix, we fill out the missing entries for the time being. The iterative algorithm is presented in Sect. 4 to deal with an incomplete QoS matrix.

Equation (4) shows the result of NMTF. It is easy to tell that the first three rows of F, which represent the developer and users U_2 and U_3 are grouped into the first user community $\hat{u}_1$ (because $U_{i,1} > U_{i,2}$, where $i \in \{1, 2, 3\}$). The last three rows, representing users $U_4 \sim U_6$, are grouped into the second user community $\hat{u}_2$ (because $U_{i,1} < U_{i,2}$, where $i \in \{4, 5, 6\}$). Similarly, columns 1, 2, and 5, which represent Map, Weather, and POI_2 services, are grouped into the first service community $\hat{s}_1$, and FlightStats and POI_1 services are grouped into the second service community $\hat{s}_2$.

$$
\begin{aligned}
&F \approx URS^T, \text{ where} \\
&F = \begin{pmatrix} 3.072 & 2.24 & 1.984 & 1.728 & 3.008 \\ 2.688 & 2.56 & 1.216 & 1.472 & 2.752 \\ 3.008 & 2.88 & 1.984 & 1.984 & 2.24 \\ 1.536 & 1.024 & 2.56 & 2.048 & 1.664 \\ 1.856 & 1.472 & 2.88 & 2.496 & 1.344 \\ 1.664 & 1.856 & 2.688 & 2.88 & 1.216 \end{pmatrix}, \quad
U = \begin{pmatrix} 0.24 & 0.01 \\ 0.23 & 3.76 \times 10^{-7} \\ 0.22 & 0.04 \\ 1.8 \times 10^{-3} & 0.21 \\ 2.03 \times 10^{-4} & 0.24 \\ 6.55 \times 10^{-5} & 0.24 \end{pmatrix}, \\
&R = \begin{pmatrix} 20.75 & 43.19 \\ 36.17 & 24.61 \end{pmatrix}, \quad
S^T = \begin{pmatrix} 0.01 & 0.28 \\ 0.01 & 0.24 \\ 0.32 & 2.34 \times 10^{-4} \\ 0.30 & 5.0 \times 10^{-3} \\ 1.60 \times 10^{-4} & 0.25 \end{pmatrix}
\end{aligned} \tag{4}
$$

3.1.2 The objective function

Since NMTF aims to find a low-rank matrix Y to approximate the QoS matrix F, a good approximation requires that values in Y be close to the original values in F. We now derive a formal objective function to evaluate the quality of the approximation matrix Y. Assume that the QoS matrix and the approximation matrix are related via the following equation:

$$F = Y + Z \tag{5}$$

where Z is the error matrix that captures the unmodeled latent factors or random noise generated during the QoS delivery process. Furthermore, we assume that Z is a matrix of i.i.d. (i.e., independently and identically distributed) zero-mean Gaussians with some constant variance σ^2, i.e., $Z_{ij} \sim \mathcal{N}(0, \sigma^2)$. Therefore,

$$p(Z_{ij}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{Z_{ij}^2}{2\sigma^2}\right), \quad \text{which implies} \tag{6}$$

$$p(F_{ij}|Y_{ij}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(F_{ij} - Y_{ij})^2}{2\sigma^2}\right), \quad \text{and} \tag{7}$$

$$p(F|Y) = \prod_{i,j} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(F_{ij} - Y_{ij})^2}{2\sigma^2}\right) \tag{8}$$

We want to maximize the probability of the QoS matrix F given Y that satisfies a set of constraints (i.e., finding the maximum likelihood). This is equivalent to find a Y that maximizes $\log p(F|Y)$, which is evaluated as:

$$\log p(F|Y) = \sum_{i,j} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(F_{ij} - Y_{ij})^2}{2\sigma^2}\right) \tag{9}$$

$$= mn \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{ij} (F_{ij} - Y_{ij})^2 \tag{10}$$

Since $mn \log \frac{1}{\sqrt{2\pi\sigma^2}}$ is a constant, the goal is to find a low-rank matrix $Y = URS^T$ that minimizes the following objective function:

$$J(U, R, S) = ||F - URS^T||_F^2 \tag{11}$$

where $||\cdot||_F$ denotes Frobenius norm; $U \in \mathbb{R}_+^{n\times k}$, $R \in \mathbb{R}_+^{k\times l}$, $S \in \mathbb{R}_+^{m\times l}$, $k \ll n$, and $l \ll m$. The nonnegative constraints ensure that a user vector is an additive combination of a set basis vectors. This allows a more intuitive interpretation than other matrix factorization approaches, such as *Singular Value Decomposition (SVD)*, which allows negative values in the matrix components.

3.2 The constrained NMTF

Directly applying NMTF discussed in Sect. 3.1 suffers a fundamental issue. That is, if no other constraints are enforced, NMTF results in matrices U and S whose entries can take arbitrary values. In this regard, entries in U and S are no longer able to denote the cluster memberships of users and services. To attack this issue, we plan to enforce a principled constraint on NMTF to ensure proper cluster assignment. Entries in U and S under this constraint can be directly interpreted as the cluster posterior probability.

To facilitate the illustration, we introduce two latent variables, $z_p^u \in \{z_1^u, \ldots, z_k^u\}$ and $z_q^s \in \{z_1^s, \ldots, z_l^s\}$, which are cluster variables for users and services. $p(z_p^u|u_i)$ (or $p(z_q^s|s_j)$) is the posterior probability on z_p^u (or z_q^s) given u_i (or s_j). We will use user-cluster indicator matrix U as an example to explain the key ideas. The same rationale applies for the service-cluster indicator matrix S.

Each entry $U_{ip} \in U$ is expected to reflect the membership of u_i on user community $\hat{u}_p$. If hard cluster membership is used, u_i will be assigned to the user community $\hat{u}_p$, where $p = \arg\max(U_{i1}, \ldots, U_{ik})$. As indicated in [16], a fundamental problem with this approach is that it does not offer a unique cluster assignment. For example, given an arbitrary positive diagonal matrix $A_U = \text{diag}(a_1, \ldots, a_k)$, we have $Y = (UA_U)(A_U^{-1}R)S^T$. This implies that u_i will be assigned to the user community $\hat{u}_p$, where $p = \arg\max(U_{i1}a_1, \ldots, U_{ik}a_k)$. Thus, different A_U will result in different user communities. Some existing approaches choose to

use L_2 normalization on rows of U [22], which helps achieve the uniqueness of the cluster assignment. However, L_2 normalization does not offer any intuitive interpretation of the result.

Inspired by [16], we propose to use L_1 normalization on rows of U. This approach not only guarantees the uniqueness of the cluster assignment but also leads to a more intuitive, posterior probability-based interpretation on the result. If we use U_{ip} to approximate the cluster conditional probability and choose $A_U = \text{diag}(a_1, \ldots, a_k)$ such that a_p is class prior of cluster $\hat{u}_p$, $U_{ip}a_p$ is essentially the posterior probability[2] and should follow the probability normalization $\sum_{p=1}^{k} U_{ip}a_p = 1, \forall i \in [1, n]$. To get A_U, we need to solve k variables with n constraints. In reality, we have $k \ll n$, and therefore, it is not guaranteed to find a feasible solution. To address this, we propose to perform posterior probabilistic clustering [16]. The idea is to directly enforce posterior probability normalization (or L_1 norm) constraint on the rows of the cluster indicator matrices U and S. Integrating the L_1 norm leads to the addition of two constraints to the objective function specified in Eq. (11): $\sum_{p=1}^{k} U_{ip} = 1$, $\sum_{q=1}^{l} S_{jq} = 1$. As a result, L_1 norm allows entries in U and S to be directly interpreted as cluster posterior probability.

3.3 Neighborhood regularization

Recent studies reveal that many real-world data distribute on low-dimensional manifold embedded in high-dimensional ambient space [4,34]. Therefore, it is desirable for the community structure derived from NMTF to respect the intrinsic geometry of the data distribution. Assume that the users $u \in \mathcal{U}$ are sampled from a distribution $P_{\mathcal{U}}$ and the services $s \in \mathcal{S}$ are sampled from a distribution $P_{\mathcal{S}}$. We propose to improve the estimation of the cluster posterior probability $p(z_p^u|u_i)$ (or $p(z_q^s|s_j)$) by exploiting the distribution of $P_{\mathcal{U}}$ (or $P_{\mathcal{S}}$). More specifically, we expect that the posterior probability $p(z_p^u|u_i)$ (or $p(z_q^s|s_j)$) changes smoothly along the geodesics in the intrinsic geometry of the data distribution $P_{\mathcal{U}}$ (or $P_{\mathcal{S}}$) [6]. That is, if two users $u_a, u_b \in P_{\mathcal{U}}$ are close neighbors in the geometrical structure of $P_{\mathcal{U}}$, the posterior probabilities $p(z_p^u|u_a)$ and $p(z_p^u|u_b)$ are "similar" to each other. Intuitively, if two users are in each other's close neighborhood, it is reasonable to believe that the two users belong to the same user community. In another word, if we know that two users share very similar QoS experience, we expect that these two users to be assigned to the same user community. In this regard, neighborhood regularization has the effect of integrating the model-based approach (i.e., the proposed QoS assessment model) with a neighborhood-based approach to improve the overall predictive accuracy.

In the sequel, we present a *graph-based model* to integrate the geometrical structure of user and service data distributions to improve the overall clustering accuracy.

The first step is to construct a user graph, $G^u = (V^u, E^u)$, which captures the similarity between different users. Each vertex v_i^u represents a user u_i. Two vertices are connected if the similarity W_{ij}^u between users u_i and u_j is larger than a certain threshold and the edge is weighted by W_{ij}^u. If two users u_i and u_j are similar (i.e., they have a large edge weight W_{ij}^u in the similarity graph), their corresponding cluster posteriors (e.g., U_{ip} and U_{jp} for cluster $\hat{u}_p$) should be similar. Therefore, $W_{ij}^u(U_{ip} - U_{jp})^2$ is expected to be small for all i, j, and p. This is equivalent to minimizing the following function,

[2] Strictly speaking, we ignore $p(u_i)$ here as $p(z_p^u|u_i) = U_{ip}a_p/p(u_i)$. For a given u_i, $p(u_i)$ is a constant for all clusters. Thus, we can just choose another diagonal matrix $A'_U = \text{diag}(a_1/p(u_i), \ldots, a_k/p(u_i))$ to absorb the constant.

$$\begin{aligned}
\mathcal{R}^u &= \sum_{p=1}^{k} \mathcal{R}_p^u = \sum_{p=1}^{k} \left(\frac{1}{2} \sum_{i,j=1}^{n} W_{ij}^u (U_{ip} - U_{jp})^2 \right) \\
&= \sum_{p=1}^{k} \left(\sum_{i=1}^{n} D_{ii}^u U_{ip}^2 - \sum_{i,j=1}^{n} W_{ij}^u U_{ip} U_{jp} \right) \\
&= \sum_{p=1}^{k} \left(\sum_{i=1}^{n} \mathbf{u}_p^T D_{ii}^u \mathbf{u}_p - \sum_{i,j=1}^{n} \mathbf{u}_p^T W_{ij}^u \mathbf{u}_p \right) \\
&= \sum_{p=1}^{k} \left(\mathbf{u}_p^T D^u \mathbf{u}_p - \mathbf{u}_p^T W^u \mathbf{u}_p \right) \\
&= \sum_{p=1}^{k} \mathbf{u}_p^T L^u \mathbf{u}_p \\
&= \mathrm{Tr}(U^T L^u U)
\end{aligned} \tag{12}$$

where $\mathbf{u}_p$ is a column vector in U, $L^u = D^u - W^u$ is the graph Laplacian of the user similarity graph and D^u is the degree matrix with $D_{ii}^u = \sum_j W_{ij}^u$. An effective way of using this to enhance the clustering accuracy is to integrate $\mathcal{R}^u$ as a regularizer into the original objective function specified in Eq. (11).

The only remaining problem is how to evaluate the similarity between users. This can be done by using the existing neighborhood-based CF approaches using distance functions, such as Pearson correlation or cosine distance. Since we essentially focus on the local geometrical structure of the user data distribution, we define the edge weight matrix W^u as follows:

$$W_{ij}^u = \begin{cases} 1, & \text{if } u_i \in \mathcal{U}_t(u_j) \text{ or } u_j \in \mathcal{U}_t(u_i) \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

where $\mathcal{U}_t(u_j)$ is a set of top-t nearest neighbors of u_i.

Following the same rationale, we formulate the following function for service clusters:

$$\mathcal{R}^s = \sum_{q=1}^{l} \mathcal{R}_q^s = \mathrm{Tr}(S^T L^s S) \tag{14}$$

where $L^s = D^s - W^s$ is the graph Laplacian of the service similarity graph, W^s is the edge weight matrix, and D^s is the degree matrix with $D_{ii}^s = \sum_j W_{ij}^s$. Similarly, we define the edge weight matrix W^s for the service similarity graph as follows:

$$W_{ij}^s = \begin{cases} 1, & \text{if } s_i \in \mathcal{S}_t(s_j) \text{ or } s_j \in \mathcal{S}_t(s_i) \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

where $\mathcal{S}_t(s_j)$ is a set of top-t nearest neighbors of s_i.

3.4 The overall objective function

Incorporating both L_1 norm constraint and neighborhood regularization, we achieve an integrated QoS assessment model. The model is underpinned by a regularized posterior probabilistic nonnegative matrix factorization (RPPNMTF). The optimal community structure can be obtained by solving the following optimization problem:

$$J_{RPPNMF} = ||F - URS^T||_F^2 + \alpha_u \mathrm{Tr}(U^T L^u U) + \alpha_s \mathrm{Tr}(S^T L^s S) + \beta_u ||U\mathbf{e} - \mathbf{e}|| + \beta_s ||S\mathbf{e} - \mathbf{e}|| \text{ s.t. } U \geq 0, S \geq 0, G \geq 0$$

We convert constraints $\sum_{p=1}^{k} U_{ip} = 1$ and $\sum_{q=1}^{l} S_{jq} = 1$ into penalty terms, $\beta_u ||U\mathbf{e} - \mathbf{e}||^2$ and $\beta_s ||S\mathbf{e} - \mathbf{e}||^2$, where $\beta_u, \beta_s \geq 0$ are L_1 normalization penalty parameters. $\mathbf{e}$ is a vector of all 1's. $\alpha_u, \alpha_s \geq 0$ are regularization parameters.

4 The iterative algorithm

The QoS assessment model aims to find a low-dimensional matrix Y to maximize log-likelihood $\log p(F|Y)$. Since the QoS matrix F contains missing entries, it is not feasible to directly maximize the log-likelihood $\log p(F|Y)$. The iterative algorithm instead maximizes the expectation of log-likelihood $\log p(F|Y)$ given the observed entires in F (denoted as F^o) and the current estimate of unobserved entries of F (denoted as F^u). In this regard, the iterative algorithm follows a similar rationale as the expectation–maximization (EM) strategy. In each iteration, the algorithm either estimates the missing QoS entries or finds an optimal community structure based on the estimated QoS entries. More specifically, in QoS estimation, it derives the expression of the expected log-likelihood $\log p(F|Y)$ and uses this expectation to estimate the missing entries in F; in community construction, it chooses Y^* to maximize the expectation. In this regard, the community structure and the missing QoS values are iteratively refined through these two steps until the algorithm converges.

4.1 QoS estimation

Assume that the estimate for Y at the end of the p^{th} step is $Y^{(p)}$. The expected log-likelihood $\log p(F|Y)$ given the observed entries in the target matrix F and the current estimate $Y^{(p)}$ is defined as:

$$Q(Y, Y^{(p)}) = E[\log p(F|Y)|F^o, Y^{(p)}] \tag{16}$$

Matrix F contains both observed entries ($F_{ij} \in F^o$) and unobserved entires ($F_{ij} \in F^u$). We compute them separately and then aggregate the results. For any observed $F_{ij} \in F^o$,

$$\log p(F_{ij}|Y_{ij}) = \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(F_{ij} - Y_{ij})^2 \tag{17}$$

Thus,

$$\log p(F^o|Y) = \sum_{F_{ij} \in F_o} \left(\log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(F_{ij} - Y_{ij})^2 \right) \tag{18}$$

For any unobserved entry $F_{ij} \in F^u$, we find

$$\begin{aligned} &E[\log p(F_{ij}|Y_{ij})|F^o, Y^{(p)}] \\ &= E_{F_{ij} \sim \mathcal{N}(Y_{ij}^{(p)}, \sigma^2)} \left[\log p(F_{ij}|Y_{ij}) \right] \\ &= \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} E_{F_{ij} \sim \mathcal{N}(Y_{ij}^{(p)}, \sigma^2)} \left[(F_{ij} - Y_{ij})^2 \right] \\ &= \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \left(E[F_{ij}^2] - 2E[F_{ij}]Y_{ij} + Y_{ij}^2 \right) \end{aligned}$$

$$= \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}\left(\sigma^2 + Y_{ij}^{(p)^2} - 2Y_{ij}^{(p)}Y_{ij} + Y_{ij}^2\right)$$
$$= \left(\log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2}\right) - \frac{1}{2\sigma^2}\left(Y_{ij}^{(p)} - Y_{ij}\right)^2$$

Thus, we have

$$E[\log p(F|Y)|F^o, Y^{(p)}] = \sum_{F_{ij}\in F_u}\left(\left(\log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2}\right) - \frac{1}{2\sigma^2}\left(Y_{ij}^{(p)} - Y_{ij}\right)^2\right) \tag{19}$$

Finally, put together Eqs. (18) and (19), we find

$$Q(Y, Y^{(p)}) = E[\log p(F|Y)|F^o, Y^{(p)}]$$
$$= -\frac{1}{2\sigma^2}\left(\sum_{F_{ij}\in F^o}\left(F_{ij} - Y_{ij}\right)^2 + \sum_{F_{ij}\in F^u}\left(Y_{ij}^{(p)} - Y_{ij}\right)^2\right) + C$$
$$\text{where } C = mn \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{|F^u|}{2} \text{ is a constant.}$$

4.2 Community construction

As implied by the above QoS estimation procedure, matrix F can be completed as follows at the end of the p-th step:

$$\forall F_{ij}^{(p)} \in F^o, F_{ij}^{(p)} = F_{ij}, \tag{20}$$
$$\forall F_{ij}^{(p)} \in F^u, F_{ij}^{(p)} = Y_{ij}^{(p)} \tag{21}$$

Now, having a completed matrix F, an optimal community structure can be obtained by finding matrices U^*, R^*, and S^* to minimize objective function J_{RPPNMF} in Eq. (16). Although J_{RPPNMF} is convex in U, R, and S respectively, it is not convex in all of them together. We introduce an iterative algorithm based on a set of efficient update rules for U, R, and S. In each iteration, U, R, and S are updated alternatively, i.e., when one matrix is being updated, others are fixed. The updates continue until a local optimal is achieved. In what follows, we introduce the update rules for U, R, and S, respectively.

4.2.1 Updating U

We use the following function J_U to denote the part of J_{RPPNMF} that is only relevant to U. Therefore, minimizing J_{RPPNMF} with respect to U is equivalent to minimizing J_U.

$$J_U = ||F - URS^T||_F^2 + \alpha_u \text{Tr}(U^T L^u U) + \beta_u ||U\mathbf{e} - \mathbf{e}||^2 \text{ s.t. } U \geq 0$$

To deal with the nonnegative constraint on U, we introduce the Lagrange multiplier $\Theta^u \in \mathbb{R}^{n\times k}$ and we have the following Lagrangian function:

$$L(U) = ||F - URS^T||_F^2 + \alpha_u \text{Tr}(U^T L^u U) + \beta_u ||U\mathbf{e} - \mathbf{e}||^2 + \text{Tr}(\Theta^u U^T)$$
$$= \text{Tr}(F^T F - 2U^T FSR^T + U^T URS^T SR^T) + \alpha_u \text{Tr}(U^T L^u U) + \beta'_u \text{Tr}(U^T UEE^T$$
$$- 2U^T EE^T + E^T E) + \text{Tr}(\Theta^u U^T) \tag{22}$$

where $E \in \mathbb{R}^{k\times k}$ is a matrix of all 1's, $\beta'_u = \beta_u/k$. We use the fact $\beta_u ||U\mathbf{e} - \mathbf{e}||^2 = \beta'_u ||UE - E||^2$ for the above derivation.

The partial derivative of $L(U)$ with respect to U is as follows:

$$\frac{\partial L(U)}{\partial U} = -2FSR^T + 2URS^TSR^T + 2\alpha_u L^u U + 2\beta_u(UE - E) + \Theta^u \tag{23}$$

Using the KKT complementarity condition for the nonnegativity $\Theta^u_{ip}U_{ip} = 0$, we have

$$\left(-2FSR^T + 2URS^TSR^T + 2\alpha_u L^u U + 2\beta_u UE - 2\beta_u E\right)_{ip} U_{ip} = 0 \tag{24}$$

Equation (24) leads to the following update rule for U:

$$U_{ip} \leftarrow U_{ip}\left(\frac{(FSR^T)_{ip} + \beta_u}{\left(\alpha_u L^u U + \beta_u UE + URS^TSR^T\right)_{ip}}\right)^{\frac{1}{2}} \tag{25}$$

4.2.2 Updating S

We can derive the update rule for S in a pretty similar manner. We use the following function J_S to denote the part of J_{RPPNMF} that is only relevant to S. Therefore, minimizing J_{RPPNMF} with respect to S is equivalent to minimizing J_S.

$$J_S = ||F - URS^T||_F^2 + \alpha_s \mathrm{Tr}(S^T L^s S) + \beta_s ||S\mathbf{e} - \mathbf{e}||^2 \ \text{ s.t. } S \geq 0 \tag{26}$$

We have the following Lagrangian function by introducing the Lagrange multiplier $\Theta^s \in \mathbb{R}^{m\times l}$:

$$L(S) = ||F - URS^T||_F^2 + \alpha_s \mathrm{Tr}(S^T L^s S) + \beta_s ||S\mathbf{e} - \mathbf{e}||^2 + \mathrm{Tr}(\Theta^s S^T) \tag{27}$$

$$= \mathrm{Tr}(F^TF - 2S^TF^TUR + S^TSR^TU^TUR) + \alpha_s \mathrm{Tr}(S^TL^sS) + \beta'_s \mathrm{Tr}(S^TSEE^T - 2S^TEE^T + E^TE) + \mathrm{Tr}(\Theta^s S^T) \tag{28}$$

where $E \in \mathbb{R}^{l\times l}$ is a matrix of all 1's, $\beta'_s = \beta_s / l$.

The partial derivative of $L(S)$ with respect to S is as follows:

$$\frac{\partial L(S)}{\partial S} = -2F^TUR + 2SR^TU^TUR + 2L^sS + 2\beta_s(SE - E) + \Theta^s \tag{29}$$

Using the KKT complementarity condition for the nonnegativity $\Theta^s_{jq}S_{jq} = 0$, we have

$$\left(-2F^TUR + 2SR^TU^TUR + 2L^sS + 2\beta_s SE - 2\beta_s E\right)_{jq} S_{jq} = 0 \tag{30}$$

Equation (30) leads to the following update rule for S:

$$S_{jq} \leftarrow S_{jq}\left(\frac{(F^TUR)_{jq} + \beta_s}{\left(\alpha_s L^s S + \beta_s SE + SR^TU^TUR\right)_{jq}}\right)^{\frac{1}{2}} \tag{31}$$

4.2.3 Updating R

The following function J_R only contains the part of J_{RPPNMF} that is relevant to R. Therefore, minimizing J_{RPPNMF} with respect to R is equivalent to minimizing J_R.

$$J_R = ||F - URS^T||_F^2 \ \text{ s.t. } S \geq 0 \tag{32}$$

We have the following Lagrangian function by introducing the Lagrange multiplier $\Theta^r \in \mathbb{R}^{k\times l}$:

$$L(R) = ||F - URS^T||_F^2 + \mathrm{Tr}(\Theta^r R^T) \tag{33}$$

The partial derivative of $L(R)$ with respect to R is as follows:

$$\frac{\partial L(R)}{\partial R} = -2U^T FS + 2U^T URS^T S + \Theta^r \tag{34}$$

Using the KKT complementarity condition for the nonnegativity $\Theta^s_{pq} R_{pq} = 0$, we have

$$\left(-2U^T FS + 2U^T URS^T S\right)_{pq} R_{pq} = 0 \tag{35}$$

Equation (35) leads to the following update rule for R:

$$R_{pq} \leftarrow R_{pq}\left(\frac{(U^T FS)_{pq}}{\left(U^T URS^T S\right)_{pq}}\right)^{\frac{1}{2}} \tag{36}$$

5 Theoretical analysis

We prove the correctness and the convergence of the proposed iterative algorithm in this section. We also provide a brief analysis of the time complexity.

Theorem 1 *If the solution converges based on the update rules in Eqs. 25, 31, and 36, the solution satisfies the KKT optimality condition.*

Proof At convergence, the values of U, R, and S will no longer change. Thus, we have

$$U_{ip} = U_{ip}\left(\frac{(FSR^T)_{ip} + \beta_u}{\left(\alpha_u L^u U + \beta_u UE + URS^T SR^T\right)_{ip}}\right)^{\frac{1}{2}} \tag{37a}$$

$$S_{jq} = S_{jq}\left(\frac{(F^T UR)_{jq} + \beta_s}{\left(\alpha_s L^s S + \beta_s SE + SR^T U^T UR\right)_{jq}}\right)^{\frac{1}{2}} \tag{37b}$$

$$R_{pq} = R_{pq}\left(\frac{(U^T FS)_{pq}}{\left(U^T URS^T S\right)_{pq}}\right)^{\frac{1}{2}} \tag{37c}$$

Equation in (37) are equivalent to the following three Eq. in (38), respectively.

$$\left(-2FSR^T + 2URS^T SR^T + 2\alpha_u L^u U + 2\beta_u UE - 2\beta_u E\right)_{ip} U_{ip}^2 = 0 \tag{38a}$$

$$\left(-2F^T UR + 2SR^T U^T UR + 2L^s S + 2\beta_s SE - 2\beta_s E\right)_{jq} S_{jq}^2 = 0 \tag{38b}$$

$$\left(-2U^T FS + 2U^T URS^T S\right)_{pq} R_{pq} = 0 \tag{38c}$$

The above three equations are essentially equivalent to the KKT complimentary conditions specified by Eqs. 24, 30, and 35. □

We now prove the convergence of the iterative algorithm. The update rules are derived for minimizing a auxiliary function, which is an convex upper bound for the original objection function [27]. In order to proceed with the proof, we first provide some important background information regarding the auxiliary function.

Definition 1 (*Auxiliary function [27]*) $Z(X, X')$ is an auxiliary function of function $J(X)$ if it satisfies the following two conditions for any X and X'.

$$(1)\ Z(X, X') \geq J(X) \quad (2)\ Z(X, X) = J(X)$$

Lemma 1 *J is nonincreasing under the following update rule if Z is an auxiliary function of J:*

$$X^{(t+1)} = \arg\min_{X} Z(X, X^{(t)}) \tag{39}$$

Proof

$$J(X^{(t)}) = Z(X^{(t)}, X^{(t)}) \geq Z(X^{(t)}, X^{(t+1)}) \geq J(X^{(t+1)})$$

□

Lemma 2

$$\begin{aligned} Z(U, \tilde{U}) = {} & ||F||^2 + n\beta_u - \sum_{ip} 2(FSR^T)_{ip}\tilde{U}_{ip}\left(1 + \log\frac{U_{ip}}{\tilde{U}_{ip}}\right) \\ & + \sum_{ip} \frac{\left(\tilde{U}RS^TSR^T\right)_{ip} U_{ip}^2}{\tilde{U}_{ip}} + \alpha_u \sum_{ip} \frac{\left(L_u\tilde{U}\right)_{ip} U_{ip}^2}{\tilde{U}_{ip}} \\ & + \beta_u \sum_{ip}\left([\tilde{U}\mathbf{e}]_i \frac{U_{ip}^2}{\tilde{U}_{ip}}\right) - \beta_u \sum_{ip} 2\tilde{U}_{ip}\left(1 + \log\frac{U_{ip}}{\tilde{U}_{ip}}\right) \end{aligned}$$

is an auxiliary function for

$$J(U) = ||F - URS^T||_F^2 + \alpha_u Tr(U^T L^u U) + \beta_u ||U\mathbf{e} - \mathbf{e}||^2,$$

which is the part of J_{NRNMF} that is relevant to U. $Z(U, \tilde{U})$ is also convex in U and its global minimum is

$$U_{ip} = U_{ip}\left(\frac{(FSR^T)_{ip} + \beta_u}{\left(\alpha_u L^u U + \beta_u UE + URS^TSR^T\right)_{ip}}\right)^{\frac{1}{2}}. \tag{40}$$

Proof To prove Lemma 2, we first derive the upper bounds for the three terms in $J(U)$, respectively. We then combine them together.

To derive the upper bound for $||F - URS^T||_F^2$, we need the following two inequalities:

$$x \geq 1 + \log(x), \quad \forall x > 0 \tag{41a}$$

$$\sum_{i=1}^{n}\sum_{p=1}^{k} \frac{(A\tilde{C}B)_{ip}C_{ip}^2}{\tilde{C}_{ip}} \geq \mathrm{Tr}(C^TACB) \tag{41b}$$

(41b) is derived in [17], which holds for any matrices $A \in \mathbb{R}_+^{n\times n}$, $B \in \mathbb{R}_+^{k\times k}$, $C \in \mathbb{R}_+^{n\times k}$, $\tilde{C} \in \mathbb{R}_+^{n\times k}$, and A, B are symmetric.

$$||F - URS^T||_F^2 = \mathrm{Tr}(F^T F - 2U^T FSR^T + U^T URS^T SR^T) \tag{42}$$

Using (41a) and setting $x = \frac{U_{ip}}{\tilde{U}_{ip}}$, we have

$$\mathrm{Tr}(U^T FSR^T) \geq \sum_{ip} (FSR^T)_{ip} \tilde{U}_{ip} \left(1 + \log \frac{U_{ip}}{\tilde{U}_{ip}}\right) \tag{43}$$

Using (41b) and setting $A = I$, $B = RS^T SR^T$, $C = U$, $\tilde{C} = \tilde{U}$, we have

$$\mathrm{Tr}(U^T URS^T SR^T) \leq \sum_{ip} \frac{\left(\tilde{U}RS^T SR^T\right)_{ip} U_{ip}^2}{\tilde{U}_{ip}} \tag{44}$$

Using (41b) and setting $A = L_u$, $B = I$, $C = U$, $\tilde{C} = \tilde{U}$, we can derive the upper bound of $\alpha_u \mathrm{Tr}(U^T L^u U)$ as

$$\alpha_u \mathrm{Tr}(U^T L^u U) \leq \alpha_u \sum_{ip} \frac{\left(L_u \tilde{U}\right)_{ip} U_{ip}^2}{\tilde{U}_{ip}} \tag{45}$$

To derive the upper bound for $\beta_u ||U\mathbf{e} - \mathbf{e}||^2$, we use Jensen's inequality and the convexity of the quadratic function.

$$\beta_u ||U\mathbf{e} - \mathbf{e}||^2 = \beta_u \sum_{i=1}^{n} \left(\sum_{p=1}^{k} U_{ip} - 1\right)^2 \tag{46a}$$

$$= \beta_u \sum_{i=1}^{n} \left(\sum_{p=1}^{k} \frac{\tilde{U}_{ip}}{[\tilde{U}\mathbf{e}]_i} - \sum_{p=1}^{k} \frac{\tilde{U}_{ip}}{[\tilde{U}\mathbf{e}]_i} \frac{[\tilde{U}\mathbf{e}]_i}{\tilde{U}_{ip}} U_{ip}\right)^2 \tag{46b}$$

$$\leq \beta_u \sum_{i=1}^{n} \sum_{p=1}^{k} \frac{\tilde{U}_{ip}}{[\tilde{U}\mathbf{e}]_i} \left(\frac{[\tilde{U}\mathbf{e}]_i}{\tilde{U}_{ip}} U_{ip} - 1\right)^2 \tag{46c}$$

$$= \beta_u \sum_{i=1}^{n} \sum_{p=1}^{k} \left(\frac{[\tilde{U}\mathbf{e}]_i}{\tilde{U}_{ip}} U_{ip}^2 - 2\tilde{U}_{ip} \frac{U_{ip}}{\tilde{U}_{ip}}\right) + \sum_{i=1}^{n} 1 \tag{46d}$$

$$\leq \beta_u \sum_{ip} \left([\tilde{U}\mathbf{e}]_i \frac{U_{ip}^2}{\tilde{U}_{ip}}\right) - \beta_u \sum_{ip} 2\tilde{U}_{ip} \left(1 + \log \frac{U_{ip}}{\tilde{U}_{ip}}\right) + n\beta_u \tag{46e}$$

Combining the upper bounds for three terms in $J(U)$, we obtain $Z(U, \tilde{U})$. It is obvious that $Z(U, \tilde{U}) \geq J(U)$ and $Z(U, U) = J(U)$. Thus, $Z(U, \tilde{U})$ is an auxiliary function of $J(U)$.

To find a local minimum of $Z(U, \tilde{U})$, we take the partial derivative of $Z(U, \tilde{U})$ with respect to U_{ip} and get

$$\frac{\partial Z(U,\tilde{U})}{\partial U_{ip}} = -2(FSR^T)_{ip}\frac{\tilde{U}_{ip}^2}{U_{ip}} + 2\left(\tilde{U}RS^TSR^T\right)_{ip}\frac{U_{ip}}{\tilde{U}_{ip}} + 2\alpha_u\left(L_u\tilde{U}\right)_{ip}\frac{U_{ip}}{\tilde{U}_{ip}} + 2\beta_u(\tilde{U}E)_{ip}\frac{U_{ip}}{\tilde{U}_{ip}} - 2\beta_u\frac{\tilde{U}_{ip}^2}{U_{ip}} \tag{47}$$

We set $\frac{\partial Z(U,\tilde{U})}{\partial U_{ip}} = 0$ and solve for U, from which we can get Eq. (40). The only remaining question is to prove that $Z(U, \tilde{U})$ is convex in U so that the local minimum is indeed the global minimum. To achieve this, we compute the Hessian matrix for $Z(U, \tilde{U})$:

$$\frac{\partial^2 Z(U,\tilde{U})}{\partial U_{ip}\partial U_{ab}} = \delta_{ia}\delta_{pb}\left(2(FSR^T)_{ip}\frac{\tilde{U}_{ip}^2}{U_{ip}^2} + 2\frac{\left(\tilde{U}RS^TSR^T\right)_{ip}}{\tilde{U}_{ip}} + 2\frac{\alpha_u\left(L_u\tilde{U}\right)_{ip}}{\tilde{U}_{ip}} + 2\frac{\beta_u(\tilde{U}E)_{ip}}{\tilde{U}_{ip}} + 2\beta_u\frac{\tilde{U}_{ip}^2}{U_{ip}^2}\right) \tag{48}$$

The Hessian matrix is a diagonal matrix with positive diagonal elements. Thus, $Z(U, \tilde{U})$ is convex in U so that the local minimum we computed above is also the global minimum. □

Lemma 3

$$Z(S,\tilde{S}) = ||F||^2 + m\beta_s - \sum_{jq}2(F^TUR)_{jq}\tilde{S}_{jq}\left(1+\log\frac{S_{jq}}{\tilde{S}_{jq}}\right) + \sum_{jq}\frac{\left(\tilde{S}R^TU^TUR\right)_{jq}S_{jq}^2}{\tilde{S}_{jq}} + \alpha_s\sum_{jq}\frac{\left(L_s\tilde{S}\right)_{jq}S_{jq}^2}{\tilde{S}_{jq}} + \beta_s\sum_{jq}\left([\tilde{S}\mathbf{e}]_i\frac{S_{jq}^2}{\tilde{S}_{jq}}\right) - \beta_s\sum_{jq}2\tilde{S}_{jq}\left(1+\log\frac{S_{jq}}{\tilde{S}_{jq}}\right)$$

is an auxiliary function for

$$J(S) = ||F - URS^T||_F^2 + \alpha_s Tr(S^TL^sS) + \beta_s||S\mathbf{e}-\mathbf{e}||^2,$$

which is the part of J_{NRNMF} that is relevant to S. $Z(S, \tilde{S})$ is also convex in S and its global minimum is

$$S_{jq} = S_{jq}\left(\frac{(F^TUR)_{jq}+\beta_s}{\left(\alpha_sL^sS+\beta_sSE+SR^TU^TUR\right)_{jq}}\right)^{\frac{1}{2}} \tag{49}$$

Proof Similar to the proof of Lemma 2. □

Lemma 4

$$Z(R,\tilde{R}) = ||F||^2 - \sum_{pq}2(U^TFS)_{pq}\tilde{R}_{pq}\left(1+\log\frac{R_{pq}}{\tilde{R}_{pq}}\right) + \sum_{pq}\frac{\left(U^TU\tilde{R}S^TS\right)_{pq}R_{pq}^2}{\tilde{R}_{pq}}$$

is an auxiliary function for

$$J(R) = ||F - URS^T||_F^2,$$

which is the part of J_{NRNMF} *that is relevant to* R. $Z(R, \tilde{R})$ *is also convex in* R *and its global minimum is*

$$R_{pq} = R_{pq}\left(\frac{(U^T FS)_{pq}}{\left(U^T URS^T S\right)_{pq}}\right)^{\frac{1}{2}} \tag{50}$$

Proof Similar to the proof of Lemma 2. □

Theorem 2 *When two matrices of* U, R, S *are fixed,* J_{NRNMF} *decreases monotonically and hence converges under the update rules of Eqs.* (25), (36), *and* (31).

Proof Assume R *and* S are fixed and we prove that $J(U)$, which is the part of J_{NRNMF} that is only relevant to U, decreases monotonically under update rule of Eq. (25). By Lemmas 1 and 2, we have

$$J(U^{(0)}) = Z(U^{(0)}, U^{(0)}) \geq Z(U^{(1)}, U^{(0)}) \geq J(U^{(1)}) \geq \ldots$$

Thus, J_{NRNMF} is nonincreasing when updating U with R, S fixed. Alternatively, when U, R or U, S are fixed, J_{NRNMF} is nonincreasing under update rules of Eqs. (31), and (36), respectively. Since J_{NRNMF} is bounded below, it converges under these update rules. □

5.1 Time complexity

The time complexity of performing RPP-NMF is in the order of $O(mn)$, where m and n are the number of services and users, respectively. More specifically, assume that the iterative algorithm stops after t_1 iterations. In each iteration, t_2 steps are required for updating U, R, and S to find (local) optimal communities. Thus, the overall complexity will be $O(t_1 t_2 (k+l)mn)$, where k and l are the number of user and service communities, respectively.

6 Framework evaluation

In this section, we use a case study and extensive experiments to evaluate the effectiveness of the proposed *CloudRec* framework.

6.1 Case study

The case study aims to illustrate how the proposed QoS assessment model and the prediction algorithm can be used for service recommendation. We continue to use the TravelAssistant scenario as discussed in Sect. 3.1.1. As there are multiple candidate POI services, the goal is to estimate the response times of these two services with respect to the developer and recommend the most efficient one.

We apply RPPNMF to the example QoS data as shown in Table 1. The result also helps demonstrate the effectiveness of the iterative algorithm in handling incomplete QoS data. As shown in Eq. (51), the same user and service clusters are generated as in Eq. (4) where the complete QoS matrix is used. Based on the community information, the response times that the developer uses to invoke POI_1 and POI_2 are predicted as 2.03 and 2.45 s, respectively. It

is worth to note that correctly predicting the relative order on the QoS delivered by the cloud services will be sufficient to help users select their desired services. Based on the estimation result, POI_1 will be recommended to the developer. The recommendation result is accurate because the same service will be recommended if the actual response times (i.e., 1.728 and 3.008 s, respectively) of the candidate POI services are known in advance.

6.2 Experiment datasets

We conduct a set of experiments to evaluate the effectiveness of the proposed QoS assessment model and the iterative algorithm. The experiments are conducted on two real-world QoS datasets obtained from [43,44,46]:

- **Dataset_1**: This dataset consists of 1.5×10^4 Web service invocation results on 100 Web services from 150 "users."

$$\begin{pmatrix} 3.072 & 2.24 & 1.984 & n/a & n/a \\ 2.688 & 2.56 & 1.216 & 1.472 & 2.752 \\ 3.008 & 2.88 & n/a & 1.984 & 2.24 \\ 1.536 & 1.024 & 2.56 & 2.048 & 1.664 \\ 1.856 & 1.472 & 2.88 & 2.496 & n/a \\ 1.664 & 1.856 & 2.688 & n/a & 1.216 \end{pmatrix}_F \approx \begin{pmatrix} 0.41 & 0.24 \\ 0.53 & 0.07 \\ 0.46 & 0.21 \\ 0.04 & 0.44 \\ 0.04 & 0.51 \\ 0.06 & 0.46 \end{pmatrix}_U \begin{pmatrix} 24.76 & 1.68 \\ 13.35 & 18.71 \end{pmatrix}_R \times \begin{pmatrix} 0.20 & 0.03 \\ 0.18 & 0.01 \\ 0.04 & 0.27 \\ 0.07 & 0.20 \\ 0.17 & 0.02 \end{pmatrix}_S^T \tag{51}$$

- **Dataset_2**: This dataset consists of 1, 974, 675 Web service invocation results on 5,825 Web services from 339 "users."

We choose these real-world QoS datasets for evaluation because they capture the discrepancies on both the user and the service sides. More specifically, the users are simulated by the computing nodes from the PlanetLab wide area network,[3] where these computing nodes may come with different hardware/software configurations and are distributed across multiple counties. Similarly, the services are developed within different development environment and deployed on different servers across a large number of countries. These datasets clearly show that users actually receive significant different QoS from the same services.

6.3 Experiment design and parameter setting

We organize the 1.5×10^4 RTT (round-trip time in seconds) records in the Dataset_1 into a 150×100 matrix, where each row represents a user and each column represents a service. This RTT matrix will be used as the QoS matrix F in our QoS assessment model. Since a lot of QoS data may be missing in a real-world setting, we randomly remove 75–95 % RTT records and use them as the testing set. The rest are used as the training set to build the communities. We then compare the removed RTT entries with the predicted values to evaluate the effectiveness of QoS assessment. The 1, 974, 675 RTT records in Dataset_2 are organized into a $339 \times 5, 825$ matrix and then same setting applies for evaluations.

To further demonstrate the effectiveness of the proposed *CloudRec* framework, we also implement two competitive model-based collaborative-filtering algorithms and apply them

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

to the QoS datasets. These algorithms include weighted nonnegative matrix factorization (WNMF) [41] and graph-regularized weighted nonnegative matrix tri-factorization (GWNMTF) [23]. For WNMF, we conduct matrix tri-factorization instead of just two-factor factorization. As discussed in the introduction, tri-factorization performs co-clustering on columns and rows and tends to generate better results than one-side clustering. We also include the classical neighborhood-based approach, which locates similar users based on the commonly invoked services and then uses these users' QoS to make the prediction. Since Pearson correlation coefficient is used to evaluate the similarity between users, we refer to this algorithm as u_PCC. To assess the impact of L_1 normalization and neighborhood regularization, respectively, we implement two alternative versions of RPPNMF, where in RPPNMF (neighbor only), the penalty parameters β_u and β_s are set to 0, and in RPPNMF (L_1 norm only), the regularization factor α_u and α_s are set to 0.

We use mean absolute error (MAE) and root mean square error (RMSE) for result evaluation and comparison between different approaches. These are widely employed metrics to measure the quality of recommendation systems, which are defined as follows:

$$\text{MAE} = \frac{\sum_{i,j} |F_{ij} - Y_{ij}|}{N} \tag{52}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i,j} (F_{ij} - Y_{ij})^2}{N}} \tag{53}$$

where N denotes the total number of predicted QoS values. We perform k-means clustering to initialize matrices U and S. R is initialized as $U^T F S$ [17]. Since entries are randomly selected and removed to create an incomplete QoS matrix F, the algorithms are run 20 times and the average MAE and RMSE are reported.

6.4 MAE performance comparison

Table 2 reports the MAE and RMSE performance of different approaches on the Dataset_1. The testing size refers to the percentage of missing entries in F. For GWNTMF [23], we set the two regularization factors μ and λ as 10 and the number is determined through the best result of a set of trail runs. To reduce parameter tuning, we set $\alpha_s = \alpha_u = \alpha$, $\beta_u = \beta_s = \beta$. In RPPNMF, $\alpha = 10$ and $\beta = 5$; in RPPNMF (neighbor only), $\alpha = 10$; and in RPPNMF (L_1 norm only), $\beta = 5$. The number of nearest neighbors in Eqs. (13) and (15) is set as 10. The numbers of user and service communities are both set to 28, i.e., $k = l = 28$. We investigate the effect of number of communities via another set of experiments.

Table 2 MAE and RMSE performance on dataset_1

Evaluation metric	MAE					RMSE				
Test size (%)	75	80	85	90	95	75	80	85	90	95
WNMTF	0.74	0.80	n/a	n/a	n/a	0.82	0.84	n/a	n/a	n/a
GWNMTF	0.71	0.76	n/a	n/a	n/a	0.86	0.87	n/a	n/a	n/a
u_PCC	0.77	0.85	1.01	1.25	1.67	1.83	1.95	2.23	2.72	3.54
RPPNMF	0.68	0.69	0.69	0.69	0.99	0.79	0.77	0.76	0.76	1.11
RPPNMF (neighbor)	0.69	0.76	n/a	n/a	n/a	0.82	0.84	n/a	n/a	n/a
RPPNMF (L_1 norm)	0.69	0.70	0.68	0.72	2.12	0.81	0.76	0.77	0.80	2.50

Table 3 P values of the paired t test on the MAE performance

Evaluation metric	P values				
Test size (%)	75	80	85	90	95
WNMTF	1.46×10^{-6}	3.11×10^{-7}	n/a	n/a	n/a
GWNMTF	1.36×10^{-6}	1.46×10^{-6}	n/a	n/a	n/a
u_PCC	1.65×10^{-8}	1.62×10^{-7}	1.33×10^{-16}	4.88×10^{-18}	7.4×10^{-12}
RPPNMF (neighbor)	2.12×10^{-6}	4.72×10^{-7}	n/a	n/a	n/a
RPPNMF (L_1 norm)	0.14	0.90	0.81	7.74×10^{-4}	2.48×10^{-5}

Table 4 MAE and RSME performance on dataset_2

Evaluation metric	MAE					RSME				
Test size (%)	75	80	85	90	95	75	80	85	90	95
WNMTF	0.060	0.068	0.072	0.082	0.072	0.097	0.104	0.105	0.116	0.085
GWNMTF	0.046	0.051	0.057	0.066	0.071	0.059	0.061	0.068	0.075	0.080
RPPNMF	0.044	0.046	0.050	0.056	0.064	0.050	0.052	0.057	0.063	0.069
RPPNMF (neighbor)	0.046	0.051	0.057	0.065	0.071	0.058	0.061	0.071	0.074	0.079
RPPNMF (L_1 norm)	0.062	0.068	0.074	0.074	0.069	0.089	0.098	0.120	0.101	0.079

We test the MAE and RMSE performance by varying the percentage of missing entries in F from 75 to 95 %. We have the following key observations. First, RPPNMF consistently generates the best results over different sparsity ratios with only few exceptions. Second, among all the cases where RPPNMF does not report the best result, RPPNMF (L_1 norm only) generates the best result for all of them. This also justifies the effectiveness of L_1 normalization. Third, as the dataset becomes more sparse, other algorithms either fail to converge or produce much worse results. The obvious performance advantage of RPPNMF over other algorithms on the sparse datasets demonstrates its effectiveness in dealing with the data sparsity issue, which makes it especially suitable for QoS assessment in the cloud.

To show that the prediction performance of RPPNMF is indeed different those of other algorithms, a paired t test has been performed between RPPNMF and the five other algorithms, respectively. The P values are computed for testing the null hypothesis that the means of the paired observations on the MAE performance[4] are equal. The alpha level is chosen as 0.05. It can be observed from Table 3 that small P values have been obtained, indicating the algorithms are not equal for most cases. For less sparse datasets, RPPNMF (L_1 norm only) has similar predictive performance as RPPNMF, which is evidenced through the relatively high P values.

To demonstrate how the proposed model and algorithm scale to a large number of users and services, Table 4 reports the MAE and RMSE performance on Dataset_2. We did not include the neighborhood-based algorithm in this dataset because such algorithms need to perform a similarity search for each individual user and hence run very slow for a large dataset. Since we have already demonstrated the performance advantage of RPPNMF over the neighborhood-based algorithms and these algorithms are known to suffer from the data

[4] The P values for the RMSE show a similar result so we skip them to avoid redundancy.

sparsity issue, we focus on comparing model-based algorithms using Dataset_2. As both RPPNMF and other baseline algorithms rely on the clustering of users and services to make the QoS prediction, we perform a column normalization on the data matrix to avoid that the clustering result is dominated by columns or rows with very large values.

We keep all the experimental setting as in Dataset_1 except for changing the numbers of user and services clusters to 80 as much larger number of users and services are involved in this dataset. The MAE and RMSE performance is consistent with that from Dataset_1. RPPNMF achieves the best performance over different sparsity ratios. The normalization has also been demonstrated to be effective as all the algorithms converge for Dataset_2.

6.5 Impact of the model parameters

We study the impact of different model parameters in this section, including the number of communities, the regularization factor, and the normalization penalty factor. These experiments are conducted on Dataset_1 as the results from Dataset_2 show a similar trend. Since the RMSE performance is consistent with the MAE performance, we only report the latter to avoid redundancy.

6.5.1 Impact of the number of communities

The number of communities plays a key role in addressing the data sparsity problem. As it is shown in Fig. 1, when missing entries are below 40 % in the target matrix F, the MAE performance fluctuates with k, l. On the other hand, when the missing entries are over 60 % in the target matrix F, the MAE performance consistently increases as k, l increase. This is because as the number of communities increases, more compact communities will be generated, which only contain highly similar users and services. Since only a small number of highly relevant QoS data will be used for prediction, it is helpful to reduce the error that may be introduced by imprecise data. Imprecise data may be very common in a sparse dataset because the missing data are typically obtained through estimation. Due to this reason, we suggest to generate more compact communities (i.e., the community size is smaller than the natural community size) for very sparse datasets in order to improve the prediction accuracy. However, this does not apply to less sparse datasets. Since most data are present and precise, more data can be used in a large community to achieve better perdition. On the other hand, one may not want to generate overly large communities because more irrelevant data may be involved that affect the prediction performance. Therefore, for less sparse datasets, it is better to stick to the sizes of natural communities.

Figure 2 shows the effect of different numbers of user and service communities. To reduce the burden of specifying multiple algorithm parameters, we assume the same number of user and service communities in our previous experiments. In this set of experiments, we fix

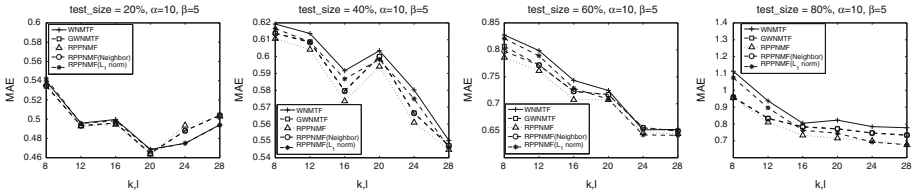


Fig. 1 Effect of the number of user and service communities, k, l

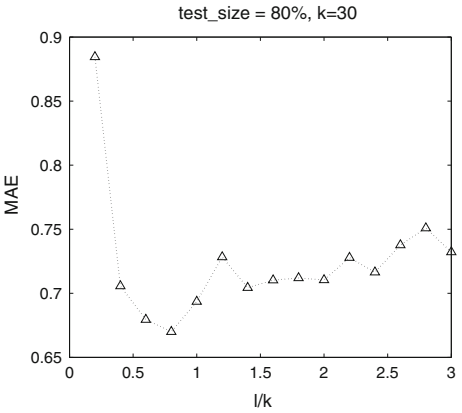


Fig. 2 Effect of different number of user and service communities

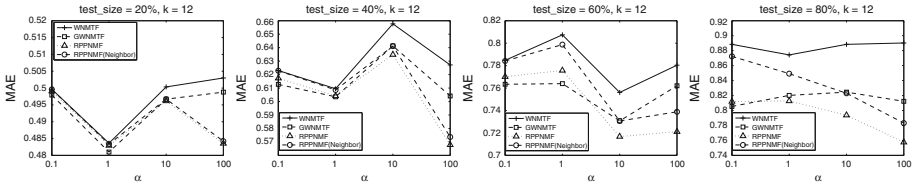


Fig. 3 Effect of the regularization parameter α

the number of user communities as 30 and change the number of service communities by varying the ratio $\frac{l}{k}$ from $\frac{1}{5}$ to 3. As can be seen, the best MAE performance is achieved when $l = 24$, which means that the number of service communities is less than the number of user communities. Hence, if a clear knowledge of the user and service clusters, a fine tuning of cluster numbers will help further improve the predictive accuracy. Otherwise, using a default set of parameters still produces reasonably good result while saving significant effort in turning the parameters.

6.5.2 Impact of the regularization parameter

We study the impact of the regularization parameter α in Fig. 3. We fix the number of communities as 12 and vary the percentage of missing entries in F from 20 to 80%. It is worth to note that the performance of WNMTF and GWNMTF do not change with α. The performance difference for different α values showed in Fig. 3 is due to the random selection of the missing entries in F. In another word, WNMTF and GWNMTF may be performed on a different QoS matrix F for a different α value. Therefore, instead of directly checking the MAE values, it is more informative to check the performance advantage of RPPNMF over WNMTF and GWNMTF. As can be seen, GWNMTF reports better MAE performance than RPPNMF when α takes a very small value (i.e., 0.1 in the experiments). As α increases, the MAE performance of RPPNMF keeps getting improved and the advantage of RPPNMF over both WNMTF and GWNMTF becomes more obvious. RPPNMF (neighbor only) performs neighborhood regularization without L_1 normalization. It exhibits a similar trend as RPPNMF, which further confirms the effectiveness of neighborhood regularization in the model. Meanwhile, RPPNMF performs better than RPPNMF (neighbor only) also justifies the effectiveness of L_1 normalization.

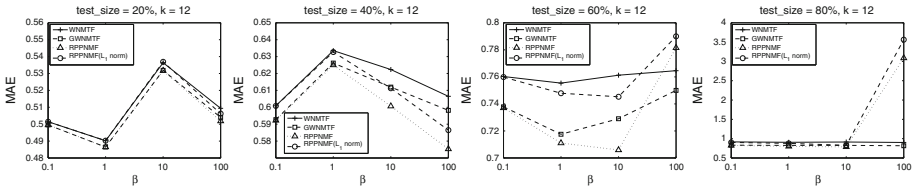


Fig. 4 Effect of the penalty parameter β

6.5.3 Impact of the normalization parameter

We study the impact of the normalization parameter β in Fig. 4. We fix the number of communities as 12 and vary the percentage of missing entries in F from 20 to 80 %. For less sparse datasets, the MAE performance keeps getting improved when β increases. While for sparse datasets, the MAE performance gets improved with β until β achieves a certain value. Then, the performance decreases when β increases further. Recall that L_1 normalization is to ensure that cluster memberships are accurately assigned by following the cluster posterior probability. However, when more data are missing, enforcing such a constraint needs to be performed based on estimated data, which may affect the accuracy of the model. Therefore, we suggest to relax the L_1 norm constraint (i.e., use a relatively small β) for very sparse datasets.

7 Conclusion

We propose *CloudRec*, a novel user-centric framework that provides personalized QoS assessment of cloud services. The cornerstone of *CloudRec* is a regularized posterior probabilistic nonnegative matrix factorization (RPPNMF) that captures the inherent cloud-related features and uses these features to group users and cloud services into a set of communities. These communities are then leveraged to predict users' QoS perception on a priori unknown cloud services. Experimental results demonstrate the good prediction accuracy of RPPNMF and its ability to handle data scarcity, which is inherent in a cloud environment.

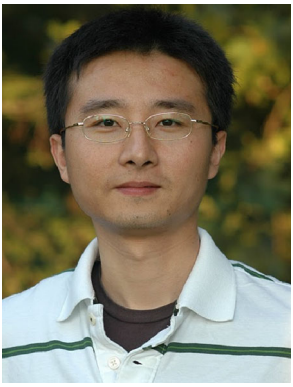
An interesting future direction is to investigate strategies to parallel the workload of *CloudRec* so that it can be deployed and handled by a powerful distributed infrastructure, such as MapReduce [13]. In this way, *CloudRec* can be scaled up to handle extremely large-scale data. *CloudRec* employs an iterative strategy to update matrices U, R, and S. A central operation in the update rules is matrix multiplication. Hence, in order to parallel the workload, we can partition matrices U, R, and S into several sub-matrices, compute the multiplications among these sub-matrices and then aggregate the results. The partition and aggregation steps can be implemented via the map and reduce operations, respectively, using the MapReduce paradigm. Some promising result has already been reported in [30], where NMF is used to factorize a tens of millions by hundreds of millions matrix with billions of nonzero entries.

References

1. (2012) http://www.cloudcomputingmarket.com/
2. Abadi DJ (2009) Data management in the cloud: limitations and opportunities. IEEE Data Eng Bull 32(1):3–12

3. Armbrust M, Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2009) Above the clouds: a berkeley view of cloud computing. Technical report, 2009. University of California at Berkeley Technical, Report No. UCB/EECS-209-28
4. Belkin M, Niyogi P (2001) Laplacian eigenmaps and spectral techniques for embedding and clustering. In NIPS '01, pp 585–591
5. Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In UAI '98, pp 43–52
6. Cai D, Wang X, He X (2009) Probabilistic dyadic data analysis with local and global consistency. In: ICML '09: Proceedings of the 26th annual international conference on machine learning, pp 105–112, New York, NY, USA, ACM
7. Canny J (2002) Collaborative filtering with privacy via factor analysis. In: SIGIR '02, pp 238–245
8. Cao Y, Chen C, Guo F, Jiang D, Lin Y, Ooi BC, Vo HT, Wu S, Xu Q (2011) Es^2: a cloud data storage system for supporting both oltp and olap. In: ICDE
9. Chen C, Chen G, Jiang D, Ooi BC, Vo HT, Wu S, Xu Q (2010) Providing scalable database services on the cloud. In: WISE, pp 1–19
10. Chen X, Zheng Z, Liu X, Huang Z, Sun H (2011) Personalized QoS-aware Web service recommendation and visualization. IEEE Trans Serv Comput, (PrePrints)
11. Chen Y, Wang L, Dong M (2010) Non-negative matrix factorization for semisupervised heterogeneous data coclustering. IEEE Trans Knowl Data Eng 22(10):1459–1474
12. Choudhury P, Sharma M, Vikas K, Pranshu T, Satyanarayana V (2012) Service ranking systems for cloud vendors. Adv Mater Res 433:3949–3953
13. Dean J, Ghemawat S (2004) Mapreduce: simplified data processing on large clusters. In: OSDI'04, pp 10–10
14. Deerwester SC, Dumais ST, Landauer TK, Furnas GW, Harshman RA (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407
15. Dhillon IS (2001) Co-clustering documents and words using bipartite spectral graph partitioning. In: KDD '01, pp 269–274
16. Ding C, Li T, Luo D, Peng W (2008) Posterior probabilistic clustering using nmf. In: SIGIR '08, pp 831–832
17. Ding C, Li T, Peng W, Park H (2006) Orthogonal nonnegative matrix t-factorizations for clustering. In: KDD '06, pp 126–135
18. Ding CHQ, He X (2005) On the equivalence of nonnegative matrix factorization and spectral clustering. In: SDM
19. Ding CHQ, He X, Zha H, Gu M, Simon HD (2001) A min-max cut algorithm for graph partitioning and data clustering. In: ICDM '01: Proceedings of the 2001 IEEE international conference on data mining, pp 107–114, Washington, DC, USA. IEEE Computer Society
20. Dong X, Halevy AY, Madhavan J, Nemes E, Zhang J (2004) Similarity search for web services. In: VLDB conference
21. Goldberg D, Nichols D, Oki BM, Terry D (1992) Using collaborative filtering to weave an information tapestry. Commun ACM 35(12):61–70
22. Gu Q, Zhou J (2009) Co-clustering on manifolds. In: KDD '09, pp 359–368
23. Gu Q, Zhou J, Ding C (2010) Collaborative filtering: weighted nonnegative matrix factorization incorporating user and item graphs. In: SDM, pp 199–210
24. Han S-M, Hassan MM, Yoon C-W, Huh E-N (2009) Efficient service recommendation system for cloud computing market. In: Proceedings of the 2nd international conference on interaction sciences: information technology, culture and human, ICIS '09, pp 839–845, New York, NY, USA. ACM
25. Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: SIGIR '99, pp 230–237
26. Hofmann T (2004) Latent semantic models for collaborative filtering. ACM Trans Inf Syst 22(1):89–115
27. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788–791
28. Lee DD, Seung HS (2000) Algorithms for non-negative matrix factorization. In: NIPS, pp 556–562
29. Li T, Ding CHQ (2006) The relationships among various nonnegative matrix factorization methods for clustering. In: ICDM, pp 362–371
30. Liu C, Yang H-C, Fan J, He L-W, Wang Y-M (2010) Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In: Proceedings of the 19th international conference on world wide web, WWW '10, pp 681–690, New York, NY, USA. ACM
31. Long B, Zhang ZM, Yu PS (2005) Co-clustering by block value decomposition. In: KDD '05: Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining, pp 635–640, New York, NY, USA. ACM

32. OWL-S (2004) http://www.daml.org/services/owl-s/
33. Rehman ZU, Hussain OK, Hussain FK (2012) Iaas cloud selection using MCDM methods. In: Proceedings of the 2012 IEEE 9th international conference on e-Business engineering, ICEBE '12, pp 246–251, Washington, DC, USA. IEEE Computer Society
34. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290:2323–2326
35. Shao L, Zhang J, Wei Y, Zhao J, Xie B, Mei H (2007) Personalized QoS prediction for web services via collaborative filtering. Web Services, IEEE International Conference on 439–446
36. Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905
37. ur Rehman Z, Hussain O, Hussain F (2013) Multi-criteria IaaS service selection based on QoS history. In: Advanced information networking and applications (AINA), 2013 IEEE 27th international conference on, pp 1129–1135
38. Vo HT, Chen C, Ooi BC (2010) Towards elastic transactional cloud storage with range query support. PVLDB 3(1):506–517
39. Yu T, Zhang Y, Lin K-J (2007) Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Trans Web 1(1):6
40. Zeng L, Benatallah B, Dumas M, Kalagnanam J, Sheng Q (2003) Quality-driven web service composition, In: WWW
41. Zhang S, Wang W, Ford J, Makedon F (2006) Learning from incomplete ratings using non-negative matrix factorization. In: SDM'06: Proceedings of the 6th SIAM conference on data mining (SDM), pp 549–553
42. Zhang Y, Koren J (2007) Efficient bayesian hierarchical user modeling for recommendation system. In: SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, pp 47–54, New York, NY, USA. ACM
43. Zhang Y, Zheng Z, Lyu MR (2011) Exploring latent features for memory-based QoS prediction in cloud computing. In: SRDS, pp 1–10
44. Zheng Z, Ma H, Lyu MR, King I (2009) Wsrec: A collaborative filtering based web service recommender system. In: ICWS, pp 437–444
45. Zheng Z, Zhang Y, Lyu MR (2010) Cloudrank: a QoS-driven component ranking framework for cloud computing. In: SRDS, pp 184–193
46. Zheng Z, Zhang Y, Lyu MR (2010) Distributed QoS evaluation for real-world Web services. In: Proceedings of 8th international conference on web services (ICWS'10), pp 83–90



Qi Yu received the PhD degree in computer science from Virginia Polytechnic Institute and State University (Virginia Tech). He is an assistant professor at the College of Computing and Information Sciences of Rochester Institute of Technology. His current research interests lie in the areas of service computing, databases, and data mining. His publications have mainly appeared in well-known journals (e.g., the *VLDB journal*, *ACM Transactions on the Web*, *World Wide Web Journal*, and *IEEE Knowledge and Data Engineering*) and conference proceedings (e.g., ICSOC and ICWS). He is a guest editor of the *IEEE Transactions on Services Computing* special issue on service query models and efficient selection. He frequently serves as a program committee member on service computing and database conferences (e.g, IEEE Cloud, SOCA, CollaborateCom, IRI, ICSOC, and APSCC). He is a member of the IEEE.