An Effective Scheme for QoS Estimation via Alternating Direction Method-based Matrix Factorization

Abstract—Accurately estimating unknown quality-of-service (QoS) data based on historical records of Web-service invocations is vital for automatic service selection. This work presents an effective scheme for addressing this issue via alternating direction method-based matrix factorization. Its main idea consists of a) adopting the principle of the alternating direction method to decompose the task of building a matrix factorization-based QoS estimator into small subtasks, where each one trains a subset of desired parameters based on the latest status of the whole parameter set; b) building an ensemble of diversified single models with sophisticated diversifying and aggregating mechanism; and c) parallelizing the construction process of the ensemble to drastically reduce the time cost. Experimental results on two industrial QoS datasets demonstrate that with the proposed scheme, more accurate QoS estimates can be achieved than its peers with comparable computing time with the help of its easy parallel realization.

Index Terms—Quarlity-of-service, QoS Estimation, Alternating Direction Method, Matrix Factorization, Ensemble, Collaborative Filtering

1 INTRODUCTION

IN THIS ERA of cloud computing, Web-services are indispensible for most industrial Web applications, thereby resulting in drastically increasing number of online Web-services. In such context, how to select suitable services from a large candidate set becomes a thorny issue [1-4]. To conduct warming-up tests for service selection is one feasible approach to this issue [1-3]. However, it is inefficient due to high economical expenses and computing time [4-7]. Hence, automatic mechanisms of Web-service selection become vital for industrial Web applications [3-11].

Service selection should be taken according to various characteristics of Web-services [4-11]. Quality-of-service (QoS) is one of the most important characteristics of a Web-service [1-5]. It can be measured by many specific metrics, e.g., request-time and throughput. Given a group of candidate services, once their QoS data are available, reliable service selection can be taken accordingly. Although it is expensive to conduct warming-up tests for QoS-data, historical QoS records are usually available. Therefore, the key problem is how to estimate unobserved QoS data based on observed ones with high accuracy.

According to pioneering research [4-11], collaborative filtering (CF)-based approaches are highly efficient to QoS estimation. A CF-based QoS estimator can be achieved by several approaches, where a popular choice is the similarity-based K-nearest-neighborhood (S-KNN) model [4-7, 12-15]. In general, an S-KNN-based QoS estimator works by a) modeling historical QoS data into user/service feature vectors, b) building the similarities among users/services to select the K-nearest-neighbors of each user/service, and c) estimation unobserved QoS data based on the observed ones by the neighbors of involved users/services. Shao *et al*. [4] propose a user S-KNN model to implement automatic QoS estimation for Web-services. Zheng *et al*. [5] propose the architecture of a Web-service recommender system whose key component is an S-KNN model for QoS estimation. Their further research [6] offers a mixed approach by combining user-oriented and service-oriented S-KNN models to obtain QoS estimates with higher accuracy. Cao *et al*. [7] propose a hybrid S-KNN model which combines the user/service similarity from the historical QoS data and inverse consumer frequency to obtain the final estimates of unknown QoS data.

Another widely-adopted kind of approaches to CF-based QoS estimation is via matrix factorization (MF) [16-22]. A QoS estimator of this kind generally works by a) modeling the historical QoS data on a given QoS metric, i.e., request time, with a user-service matrix. In this matrix, each row stands for a specified user, each column for a specified service and each entry for the QoS record by a specified user on a specified service. Note that since each user can only touch a finite subset of the whole service set, this user-item matrix is highly incomplete with numerous missing data; b) building a low-rank approximation to this incomplete user-service matrix based on its known entries only; and c) estimation unobserved QoS data in the user-service matrix with corresponding entries in its low-rank approximation. As unveiled by recent research in the recommender system community [12-19], MF-based CF models are highly accurate and scalable for many cases. They are also very effective in addressing the problem of QoS estimation [8-11, 20]. However, motivated by practical requirements, it is still desired to further

•

improve the accuracy of MF-based QoS estimators.

Zhang et al. [8] propose an MF-based QoS estimator that integrates the time interval as an additional factor. Lo et al. [9] propose an extended MF-based QoS estimator with the consideration of location information in each historical QoS record. These approaches have proven effective in generating accurate QoS estimates. However, their applications are restricted when the required auxiliary information is not available [8, 9]. Zheng et al. [10] propose to improve the MF-based QoS estimator by considering the neighborhood information. Nonetheless, it cost much memory to store the pair-wise factors for identifying the neighbors of each user/service in real applications. Luo *et al*. [19, 20] propose to ensemble a set of diversified non-negative latent factor models to achieve a highly accurate QoS estimator. This ensemble model is able to achieve high prediction accuracy. However, its base models train desired parameters through the single-element-dependent non-negative multiplicative update, which usually requires excessive training iterations to converge, thereby resulting in much computational time.

In general, most of current QoS estimators achieve high accuracy by extending the information inputs. As proven in prior works [8-11, 19, 20], with properly modeled learning objectives, additional information besides historical QoS data can be integrated into an MF-based QoS estimator seamlessly. However, the performance of such extensions relies heavily on the benchmark models. Hence, it is highly significant to develop effective QoS estimators relying on QoS data only. This work focuses on designing an effective MF-based scheme for QoS estimation with a) high prediction accuracy, b) low time cost, and c) dependence on the QoS data only. Its main idea consists of a) adopting the principle of the alternating direction method to accelerate the training process of an MF model without loss of prediction accuracy; b) building the ensemble of single models with several diversifying and aggregating mechanism; and c) parallelizing the construction process of the ensemble to reduce the computational time.

The rest of this paper is organized as follows; Section II formulates the problem. Section III presents the scheme. Section IV gives the experiments and discusses the results. And finally, Section V concludes this paper.

2 Problems Formulation

For an MF-based QoS estimator, the fundamental data source is a user-service QoS-matrix as defined in [4-11, 20]:

Definition 1. Given a user set U and a service set S, a *user-service QoS-matrix* Q is a $|U|\times|S|$ matrix consisting of the historical records of a specified QoS-property by U on S where each known entry $q_{u,s}$ denotes the QoS-record by user u on service s.

As mentioned before, Q is highly incomplete due to the impossibility for a user to invoke all services from S. Let Q_K and Q_U denote the known and unknwon entry sets of Q respectively, we have the following problem [4-11, 20]:

Definition 2. Given Q, the *problem of MF-based QoS estimation* is to build a rank-f approximation $\hat{Q}$ to Q based on Q_K, such that the most accurate estimate $\hat{q}_{u,s}$ of each unknown entry $q_{u,s}\in Q_U$ is generated.

As illustrated in [8-20], $\hat{Q}$ usually consists of two rank-f matrices P and E, where P has dimension $|U|\times f$, E has $f\times|S|$ and $f\ll\min\{|U|,|S|\}$. Note that P and E are usually interpreted as the user and service latent-feature-matrices, which reflect the user and service characteristics hidden in the historical QoS-data. With them, the desired $\hat{Q}$ is denoted by $\hat{Q}=PE$, and the objective turns to solve P and E according to Q_K. This is implemented by building the cost function, e.g., Euclidean distance or Kullback-Leibler divergence, to measure the difference between Q and $\hat{Q}$ [8-20], and then minimize it with respect to P and E. With the loss function based on Euclidean distance, the problem is formulated by:

$$\varepsilon(P,E)=\left\|Q-\hat{Q}\right\|_F^2=\left\|Q-PE\right\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. Note that the incompleteness of Q makes (1) not solvable. Nevertheless, according to [16-22], to expand (1) into a single-element-dependent form enables the optimization process to focus on Q_K rather than on Q, thereby resulting in the following problem:

$$\varepsilon(P,E)\approx\sum_{(u,s)\in Q_K}\left(q_{u,s}-P_{u,\cdot}E_{\cdot,s}\right)^2=\sum_{(u,s)\in Q_K}\left(q_{u,s}-\sum_{k=1}^{f}p_{u,k}e_{k,s}\right)^2 \tag{2}$$

where $P_{u,\cdot}$ and $E_{\cdot,s}$ denotes the uth row-vector of P and sth column-vector of E, respectively. Moreover, as indicated in [8-20], a more efficient extension of (2) is achieved by integrating the linear-bias-factors and l_2 regularization into it:

$$\begin{aligned}\varepsilon(B,C,P,E)=\sum_{(u,s)\in Q_K}\Bigg(&\left(q_{u,s}-b_u-c_s-\sum_{k=1}^{f}p_{u,k}e_{k,s}\right)^2\\&+\lambda_B b_u^{\ 2}+\lambda_C c_s^{\ 2}+\lambda_P\sum_{k=1}^{f}p_{u,k}^2+\lambda_E\sum_{k=1}^{f}e_{k,s}^2\Bigg);\end{aligned} \tag{3}$$

where b_u and c_s are linear-bias-factors for user u and service s, B and C are length-$|U|$ and $|S|$ vectors consisting of all user and service biases, and λ_B, λ_C, λ_P and λ_E are positive constants denoting the regularizing coefficients for B, C, P and Q, respectively. With loss function (3), the objective is to minimize ε with respect to B, C, P and E,

$$\left(B,C,P,E\right)=\underset{B,C,P,E}{\arg\min}\quad\varepsilon\left(B,C,P,E\right), \tag{4}$$

for obtaining a converging model able to generate estimates for unknown entries in Q.

3 Proposed Scheme

3.1 Training Scheme with Alternating Direction Method

The principle of ADM [21, 22] is to decompose the original optimization task into tiny sub-ones, and then solve each subtask sequentially. The optimization process of a subtask relies on the updated status of those previously solved ones, thereby leading to fast convergence.

Hence, the first step to implement an ADM-based QoS estimator is to decompose the optimization task (4).

Note that (4) is bi-linear and non-convex. It is thus not analytically solvable. A widely-adopted training scheme effective in addressing such a problem is the alternating least square (ALS)-based optimization [17, 23, 24], which works by fixing part of the decision parameters, e.g., B, C, and P in (3), to make the system convex and analytically solvable with respect to the remaining parameters, e.g., Q in (3). With ALS, the original training task (4) is actually decomposed into four subtasks as follows,

$$\begin{aligned}
B' &\leftarrow \arg\min_{B} \ \varepsilon(B,C,P,E),\\
C' &\leftarrow \arg\min_{C} \ \varepsilon(B,C,P,E),\\
P' &\leftarrow \arg\min_{P} \ \varepsilon(B,C,P,E),\\
E' &\leftarrow \arg\min_{E} \ \varepsilon(B,C,P,E),
\end{aligned} \tag{5}$$

To introduce the ADM principle into the ALS-based training process of (3), we further decompose each ALS task in (5) into small sub tasks where each one deals with the optimization of a single parameter only. Here we take the training process of B for example. For a single parameter b_u in B, which is the linear bias for user u, let (B-b_u) denote parameters in B except b_u, (3) is convex with b_u by fixing the others, i.e., (B-b_u), C, P and E. Hence, we analytically solve (3) with respect to b_u by adopting ALS as follows,

$$\begin{aligned}
& b'_u \leftarrow \arg\min_{b_u} \ \varepsilon(B,C,P,E),\\
& \Rightarrow \frac{\partial\varepsilon}{\partial b_u} = -2\sum_{s\in Q_K(u)}\left(\left(q_{u,s}-b_u-c_s-\sum_{k=1}^{f}p_{u,k}e_{k,s}\right)-\lambda_B b_u\right)=0\\
& \Rightarrow \sum_{s\in Q_K(u)}\left(q_{u,s}-c_s-\sum_{k=1}^{f}p_{u,k}e_{k,s}\right)=|Q_K(u)|(1+\lambda_B)b_u\\
& \Rightarrow b'_u \leftarrow \frac{\sum_{s\in Q_K(u)}\left(q_{u,s}-c_s-\sum_{k=1}^{f}p_{u,k}e_{k,s}\right)}{|Q_K(u)|(1+\lambda_B)};
\end{aligned} \tag{6}$$

where $Q_K(u)$ denotes the subset of Q_K related to user u.

By applying (6) to each parameters in B, we decompose the ALS task with respect to B in (6) into $|U|$ subtasks, each of which updates one single parameter in B. By analogy, we decompose the ALS taks with respect to C, P and Q to obtain their training rules as follows,

$$\forall u\in U, s\in S, k\in\{1,\ 2,\ \dots,\ f\}:$$
$$\begin{cases}
b'_u \leftarrow \dfrac{\sum_{s\in Q_K(u)}\left(q_{u,s}-c_s-\sum_{k=1}^{f}p_{u,k}e_{k,s}\right)}{|Q_K(u)|(1+\lambda_B)},\\[2ex]
c'_s \leftarrow \dfrac{\sum_{u\in Q_K(s)}\left(q_{u,s}-b_u-\sum_{k=1}^{f}p_{u,k}e_{k,s}\right)}{|Q_K(s)|(1+\lambda_C)},\\[2ex]
p'_{u,k} \leftarrow \dfrac{\sum_{s\in Q_K(u)}e_{k,s}\left(q_{u,s}-b_u-c_s-\sum_{j=1,j\neq k}^{f}p_{u,j}e_{j,s}\right)}{\sum_{s\in Q_K(u)}e_{k,s}^2+|Q_K(u)|\lambda_P},\\[2ex]
e'_{k,s} \leftarrow \dfrac{\sum_{u\in Q_K(s)}p_{u,k}\left(q_{u,s}-b_u-c_s-\sum_{j=1,j\neq k}^{f}p_{u,j}e_{j,s}\right)}{\sum_{u\in Q_K(s)}p_{u,k}^2+|Q_K(s)|\lambda_E};
\end{cases} \tag{7}$$

where $Q_K(s)$ denotes the set of data in Q_K related to service s. Note that as shown in [19, 20], because QoS-data are defined on the positive field of real numbers, it is meaningful to constrain the resulting MF-based QoS estimator to be non-negative. Here we adopt the non-negative projection strategy discussed in [25, 26] to keep the model parameters non-negative. More specifically, we extend the ALS-based parameter learning rules as follows,

$$\forall u\in U, s\in S, k\in\{1,\ 2,\ \dots,\ f\}:$$
$$\begin{cases}
b'_u \leftarrow \max\left\{\dfrac{\sum_{s\in Q_K(u)}\left(q_{u,s}-c_s-\sum_{k=1}^{f}p_{u,k}e_{k,s}\right)}{|Q_K(u)|(1+\lambda_B)}, b_u\right\},\\[2ex]
c'_s \leftarrow \max\left\{\dfrac{\sum_{u\in Q_K(s)}\left(q_{u,s}-b_u-\sum_{k=1}^{f}p_{u,k}e_{k,s}\right)}{|Q_K(s)|(1+\lambda_C)}, c_s\right\},\\[2ex]
p'_{u,k} \leftarrow \max\left\{\dfrac{\sum_{s\in Q_K(u)}e_{k,s}\left(q_{u,s}-b_u-c_s-\sum_{j=1,j\neq k}^{f}p_{u,j}e_{j,s}\right)}{\sum_{s\in Q_K(u)}e_{k,s}^2+|Q_K(u)|\lambda_P}, p_{u,k}\right\},\\[2ex]
e'_{k,s} \leftarrow \max\left\{\dfrac{\sum_{u\in Q_K(s)}p_{u,k}\left(q_{u,s}-b_u-c_s-\sum_{j=1,j\neq k}^{f}p_{u,j}e_{j,s}\right)}{\sum_{u\in Q_K(s)}p_{u,k}^2+|Q_K(s)|\lambda_E}, e_{k,s}\right\};
\end{cases} \tag{8}$$

Note that with B, C, P and E starting with non-negative initial hypothesis, i.e., all factors in B, C, P and E are initialized to fulfill

$$\begin{aligned}
&\forall u\in U, s\in S, k\in\{1,\ 2,\ \dots,\ f\}:\\
&b_u\geq 0,\ c_s\geq 0,\ p_{u,k}\geq 0,\ e_{k,s}\geq 0;
\end{aligned} \tag{9}$$

the ALS-based update rule (8) will keep them non-negative.

With (8), the original optimization task is divided into $(|U|+|S|)\times(f+1)$ subtasks. To achieve a highly-efficient ADM-based training process, it is necessary to arrange them into a properly designed solving-sequence [21, 22]. By carefully investigating the ALS-based update rule (8), we have the following results:

a) The update of parameters inside B/C is independent, e.g., for users u and v, the update of b_u does not affect that of b_v. However, B and C are interdependent, e.g., for users u and service s, the update of b_u affects c_s if $q_{u,s}\in Q_K$. Naturally, the update of B and C affects that of P and E; and

b) The update of parameters in P/E is dependent, i.e., the update of the kth parameter of a user/service affects that of the remaining k+1~fth parameters. Parameters in P and E are also interdependent.

Hence, we split the ALS-based optimization task (5) into (f+2) subtasks, and solve them sequentially following the ADM principle as follows:

$$\begin{aligned}
&1.\ \forall u\in U, b'_u \leftarrow \arg\min_{b_u}\ \varepsilon(B,C,P,E),\\
&2.\ \forall s\in S, c'_s \leftarrow \arg\min_{c_s}\ \varepsilon(B',C,P,E),\\
&3.\ \mathit{for}\ k=1\sim f,\ \forall u\in U, s\in S\\
&\begin{cases}
p'_{u,k} \leftarrow \arg\min_{p_{u,k}}\ \varepsilon\left(B',C',P'_{,1\sim k-1},P_{,k\sim f},E'_{1\sim k-1,},E'_{k\sim f,}\right),\\
e_{k,s} \leftarrow \arg\min_{e_{k,s}}\ \varepsilon\left(B',C',P'_{,1\sim k-1},P_{,k\sim f},E'_{1\sim k-1,},E'_{k\sim f,}\right);
\end{cases}
\end{aligned} \tag{10}$$

where $P_{.,*}$ and $E_{*,.}$ denote the columns/rows in P and E corresponding to specified feature dimensions, respectively. Note that the parameter optimization of (10) follows (8). An illustrative example for a single iteration of the training process (10) is given in Fig. 1.

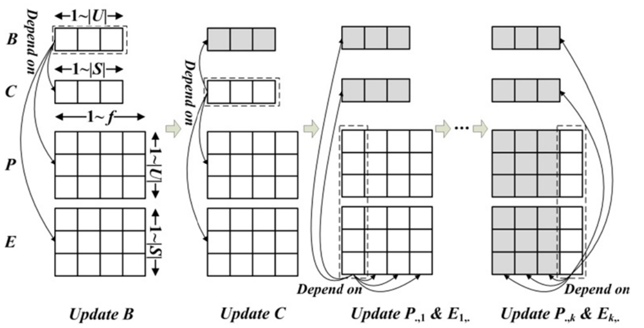


Fig. 1. The update process of all involved parameters in one iteration.

According to (10) and Fig. 1, all (f+2) subtasks are sequentially performed in each training iteration. The update of B is firstly taken, followed by the update of C taken based on the updated B, i.e., B'. Subsequently, parameters in P and E are grouped by the dimension of the latent factor space to obtain the remaining f subtasks, which are sequentially done based on B', C', and updated parameters in P and E.

3.2 Building Ensemble

As discussed in [27-37], ensemble methods are effective in boosting the accuracy of machine-learning models. Moreover, as shown in [20, 32-37], it is also feasible to further boost the performance of an MF model through building ensembles. Note that for building an effective ensemble which can outperform any of its base models, it is necessary to diversify these base models without impairing their accuracy [29, 34].

From (3), we see that an MF-based QoS-predictor estimates $q_{v,r} \in Q_U$ as follows,

$$\hat{q}_{v,r} = b_v + c_r + \sum_{k=1}^{f} p_{v,k} e_{k,r}. \tag{11}$$

From (8) we see that this rule decides the training process. For instance, if an MF-based model adopts (2) as its objective function, then it has the following rule:

$$q_{v,r} \in Q_U : \hat{q}_{v,r} = \sum_{k=1}^{f} p_{v,k} e_{k,r}. \tag{12}$$

According to (10) and (11), its ALS-based parameter training process following the ADM principle is given by

$$\textit{for } k = 1 \sim f,\ \forall u \in U, s \in S$$
$$\begin{cases} p'_{u,k} \leftarrow \underset{p_{u,k}}{\arg\min}\ \varepsilon\left(B', C', P'_{.,1\sim k-1}, P_{.,k\sim f}, E'_{1\sim k-1,.}, E'_{k\sim f,.}\right), \\ e_{k,s} \leftarrow \underset{e_{k,s}}{\arg\min}\ \varepsilon\left(B', C', P'_{.,1\sim k-1}, P_{.,k\sim f}, E'_{1\sim k-1,.}, E'_{k\sim f,.}\right); \end{cases} \tag{13}$$

with the following parameter update rule,

$$\forall u \in U, s \in S, k \in \{1, 2, ..., f\}:$$
$$\begin{cases} p'_{u,k} \leftarrow \max\left\{ \dfrac{\sum_{s \in Q_K(u)} e_{k,s}\left(q_{u,s} - \sum_{j=1, j \neq k}^{f} p_{u,j} e_{j,s}\right)}{\sum_{s \in Q_K(u)} e_{k,s}^2 + |Q_K(u)|\lambda_P}, p_{u,k} \right\}, \\ e'_{k,s} \leftarrow \max\left\{ \dfrac{\sum_{u \in Q_K(s)} p_{u,k}\left(q_{u,s} - \sum_{j=1, j \neq k}^{f} p_{u,j} e_{j,s}\right)}{\sum_{u \in Q_K(s)} p_{u,k}^2 + |Q_K(s)|\lambda_E}, e_{k,s} \right\}; \end{cases} \tag{14}$$

TABLE I
ESTIMATION RULES AND CORRESPONDING OBJECTIVE FUNCTIONS

No.	Estimation rule	Objective Function
1.	$q_{v,r} \in Q_U:$ $\hat{q}_{v,r} = b_v + c_r + \sum_{k=1}^{f} p_{v,k} e_{k,r}$	$\varepsilon = \sum_{(u,s) \in Q_K} \left(\left(q_{u,s} - b_u - c_s - \sum_{k=1}^{f} p_{u,k} e_{k,s} \right)^2 + \lambda_B b_u^2 + \lambda_C c_s^2 + \lambda_P \sum_{k=1}^{f} p_{u,k}^2 + \lambda_E \sum_{k=1}^{f} e_{k,s}^2 \right)$
2.	$q_{v,r} \in Q_U:$ $\hat{q}_{v,r} = b_v + \sum_{k=1}^{f} p_{v,k} e_{k,r}$	$\varepsilon = \sum_{(u,s) \in Q_K} \left(\left(q_{u,s} - b_u - \sum_{k=1}^{f} p_{u,k} e_{k,s} \right)^2 + \lambda_B b_u^2 + \lambda_P \sum_{k=1}^{f} p_{u,k}^2 + \lambda_E \sum_{k=1}^{f} e_{k,s}^2 \right);$
3.	$q_{v,r} \in Q_U:$ $\hat{q}_{v,r} = c_r + \sum_{k=1}^{f} p_{v,k} e_{k,r}$	$\varepsilon = \sum_{(u,s) \in Q_K} \left(\left(q_{u,s} - c_s - \sum_{k=1}^{f} p_{u,k} e_{k,s} \right)^2 + \lambda_C c_s^2 + \lambda_P \sum_{k=1}^{f} p_{u,k}^2 + \lambda_E \sum_{k=1}^{f} e_{k,s}^2 \right)$
4.	$q_{v,r} \in Q_U:$ $\hat{q}_{v,r} = \sum_{k=1}^{f} p_{v,k} e_{k,r}$	$\varepsilon = \sum_{(u,s) \in Q_K} \left(\left(q_{u,s} - \sum_{k=1}^{f} p_{u,k} e_{k,s} \right)^2 + \lambda_P \sum_{k=1}^{f} p_{u,k}^2 + \lambda_E \sum_{k=1}^{f} e_{k,s}^2 \right)$

Obviously, MF-models that respectively rely on (8) and (14) are significantly different in the training process, and different training processes result in model diversity. From this point of view, we diversify base models in the ensemble by adopting different QoS-estimation rules, which actually result in different loss-functions and parameter training processes. Table I summarizes the estimation rules along with the corresponding learning objectives adopted in our ensemble. For conciseness, we do not list corresponding parameter update rules; however, they can be easily inferred following (6)-(8).

Moreover, in order to further diversify the base models in our ensemble, we inject randomness into each base model, which is a widely-adopted model-diversifying strategy in the area of artificial neural networks [27]. For each base model, the possibly involved features are B, C, P and E. When applying randomness-injection to these base models, these features are initialized in a certain interval, e.g., (0,0.004), randomly. With such a process, each base model possesses different initial hypothesis [20, 27], thereby leading to further diversity among them. To be shown later in the experiments, the diversified base models are able to achieve closely high accuracy.

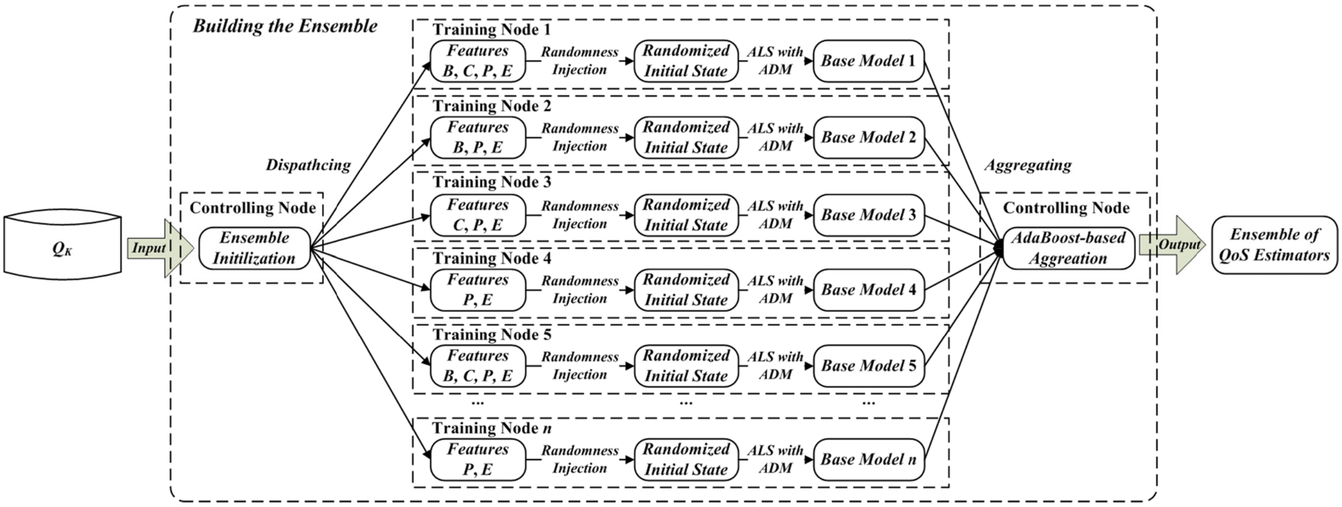


Fig. 2. The processing flow of the proposed scheme.

With the diversified base models, we finally aggregate their estimates for unknown QoS data in Q_U as follows,

$$q_{v,r} \in Q_U : \hat{q}_{v,r} = \frac{\sum_{i=1}^{n} w_i \cdot \hat{q}_{v,r}^{i}}{\sum_{i=1}^{n} w_i}, \tag{15}$$

where $\hat{q}_{v,r}^{i}$ denotes the estimate by the ith base model, n denotes the number of base models in the ensemble, and w_i denotes the model weight of the ith base model, respectively. Note that the model weights can be obtained through several model aggregating approaches [27-37]. Here we choose the AdaBoost approach [20, 30-37] to compute them. As proven in [20, 32, 35-37], this approach is effective in aggregating MF-based models for missing-data-estimation. In general, with AdaBoost the weight of each base model is computed according to its training error and the training weights assigned to each training instance by previous-trained base models. For more details of this process, please refer to [20, 32].

3.3 Parallelization

Note that an ensemble of MF-based QoS-estimators usually contains dozens of base models. If we build these base models sequentially, the computing time is much more than that of building a single MF-based estimator. To reduce the computing time, it is necessary to parallelize an ensemble building process. Since the training process of each base model is independent of that of the others, it is easy to implement such parallelization by training different base models on different computational nodes. From this point of view, we give the processing flow of the parallelized ensemble in Fig. 2.

As depicted in Fig. 2, the ensemble building process includes a) with the input QoS data, the controlling node dispatches the building task of each base model to each training node; b) each training node randomly initializes the features of each base model, and executes the corresponding feature training process based on ALS with ADM; and c) the resulting base models are sent back to the controlling node to implement the AdaBoost-based aggregation, resulting in the desired ensemble of QoS-estimators. With sophisticated parallel computing frameworks like map-reduce, this process can be implemented conveniently. We name the proposed scheme in Fig. 2 as the <u>E</u>nsemble of <u>A</u>lternating-direction-method-based <u>M</u>atrix <u>F</u>actorization <u>M</u>odels (EAMFM). Next we analyze its computational and storage complexity.

3.4 Complexity Analysis

The computational complexity of EAMFM depends on that of its base models. From (6)–(8), (13), (14) and Table I, we see that its base models employ the ALS-based iterative parameter training scheme with the ADM principle, where each training iteration is decomposed into f~(f+2) sub tasks. Note that processing each subtask requires a single traverse on Q_K for computing the update gain of corresponding features. Hence, the computational cost of each iteration is $\Theta(|Q_K|\times f)$. Assuming that T iterations are required to make each base model coverage, we have the training cost $\Theta(|Q_K|\times f\times T)$.

After finishing training the base models, the ensemble is built based on AdaBoost. As described in [20, 32], such a process requires testing the estimation error of each base model on Q_K for deciding their weights in the ensemble. To test the estimation error of a base model requires traversing on Q_K and computing the estimate $\hat{q}_{u,s}$ for $\forall q_{u,s} \in Q_K$, thereby leading to the cost at $\Theta(|Q_K|\times f)$. Hence, given that an EAMFM model consists of n base models, its computational complexity comes to $\Theta(|Q_K|\times f\times T\times n)$. In theory, this computational complexity is roughly n times that of building a single MF-based QoS-estimator. However, as shown in Fig. 2 and later in the experiments, with parallelization the computing time of building an EAMFM is comparable with that of building a single QoS-estimator.

In terms of the storage complexity, each base model needs to cache corresponding features as depicted in Table I and Fig. 2. Depending on their estimation rules and objective functions, their storage complexity ranges from $\Theta((|U|+|S|)\times f)$ to $\Theta((|U|+|S|)\times(f+1))$. By reasonably omitting the lower-order terms, the storage complexity of n base models in an EAMFM model is $\Theta((|U|+|S|)\times f\times n)$. Meanwhile, it is necessary to cache Q_K for taking the training process. Hence, the total storage cost of an EAMFM model is $\Theta((|U|+|S|)\times f\times n+|Q_K|)$. Given that f<<min{$|U|,|S|$}, such storage complexity can be easily handled in industrial applications.

4 EXPERIMENTS AND RESULTS

4.1 General Settings

Datasets. Two datasets are involved in the experiments; both are collected by the WS-Dream system [3, 5-6, 8, 10, 11]. The first one, i.e., D1, is the response-time dataset, which contains 1,873,838 response-time data by 339 users on 5,825 real-world Web-services. The second dataset, i.e., D2, consists of 1,831,253 throughput data by 339 users on 5,825 real-world Web-services. Note that these two WS-Dream datasets are the largest QoS datasets publicly available, and are commonly employed in prior research regarding QoS estimation [3, 5-11, 20].

TABLE II
DETAILED SETTINGS OF THE TEST CASES

Dataset	No.	Train:Test	Training data	Testing data
D1	D1.1	5%:95%	93,692	1,780,146
	D1.2	10%:90%	187,384	1,686,454
	D1.3	15%:85%	281,076	1,592,762
	D1.4	20%:80%	374,768	1,499,070
D2	D2.1	5%:95%	91,563	1,739,690
	D2.2	10%:90%	183,125	1,648,128
	D2.3	15%:85%	274,688	1,556,565
	D2.4	20%:80%	366,251	1,465,002

On both datasets, we have designed different test cases for validating the performance of each tested model under different data densities, as shown in Table II. The column 'Train: Test' means the ratio of training data to testing ones; e.g., 5%:95% denotes that 5% of the given data are chosen randomly and employed as training data, to predict the remaining 95% of the given data. This process is repeated for 10 times to obtain more objective results. Naturally, all the testing data are not involved in the training process.

Evaluation Metric. In this work, we focus on the accuracy of generated QoS-predictions, since it can directly reflect whether or not the model has captured the essential characteristics of given data. Hence, we employ the commonly accepted evaluation metrics mean absolute error (MAE) [39, 40] as the evaluation metric, which is formulated by

$$MAE=\sum_{(v,r)\in\Gamma}\left|q_{v,r}-\hat{q}_{v,r}\right|\Big/|\Gamma|, \tag{16}$$

where Γ denotes the testing dataset, and naturally $Q_K\cap\Gamma=\phi$. For a specified model, low MAE stands for high prediction accuracy. Note that during our experiment, we have recorded them both with 4 significant digits.

Meanwhile, we are also concerned with the computational efficiency of the proposed model, which is directly reflected by the consumed time to train a tested model. Hence, we have recorded it during our experiments.

Experimental Process. The experiments include four parts:

A. Accuracy test of base models with different combinations of linear-bias-factors, which demonstrates that the diversified base models are able to generate accurate QoS estimates;

B. Accuracy test of EAMFM with the number of base models increasing from 1~20, indicating that EAMFM is effective in achieving higher accuracy than its component models;

C. Computing time test of EAMFM with the number of computing nodes increasing from 1~20, indicating that efficient parallelization can drastically reduce the computing time of EAMFM; and

D. The comparison between EAMFM and three state-of-the-art QoS estimators to show the effectiveness of the proposed scheme.

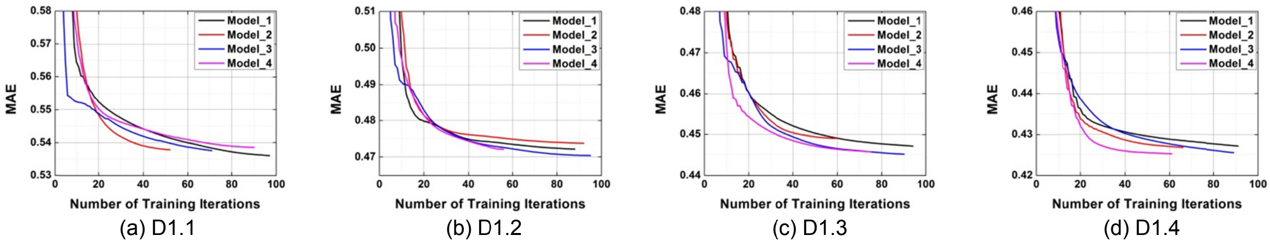
To present a concise report, for Experiment Parts A-C we only present the results on D1; however, similar results are obtained on D2 in our tests.

All experiments are conducted on a cluster with 32 computing nodes, where each node is equipped with a 3.0GHz CPU and 4GB memory. All tested models are implemented in JAVA SE7U60.

4.2 Part A

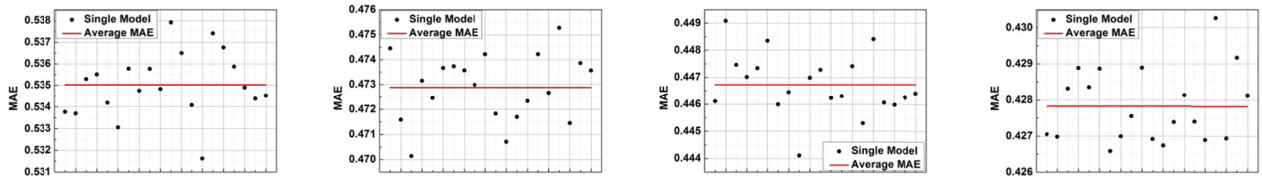
As mentioned before, the effectiveness of an ensemble depends on the diversity and accuracy of its base models, Section III states how we diversify the base models in an EAMFM model in detail. However, it is necessary to check whether or not these diversified models have closely high prediction accuracy, -if not, they cannot be used to build an effective ensemble.

We first test the accuracy of four different base models, which are marked as Models 1-4 corresponding to the estimation rules and objective functions 1-4 in Table I, respectively. For all models, the regularizing parameters are set as $\lambda_B=\lambda_C=\lambda_P=\lambda_E=0.1$. To make a fair comparison,



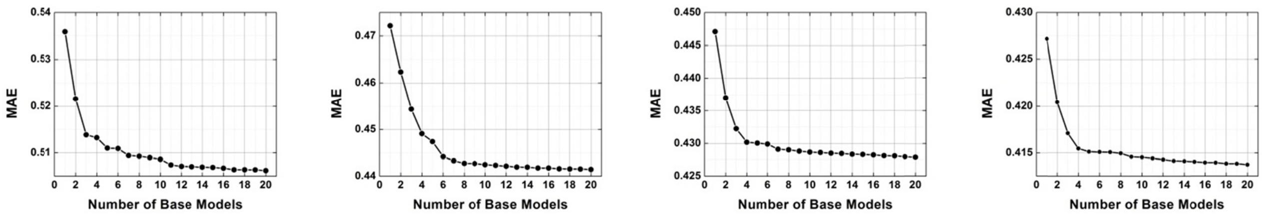
(a) D1.1 (b) D1.2 (c) D1.3 (d) D1.4

Fig. 3. Typical training process of base models with different estimation rules and objective functions. In the legends, Models 1-4 correspond to estimation rules and objective functions 1-4 in Table I.



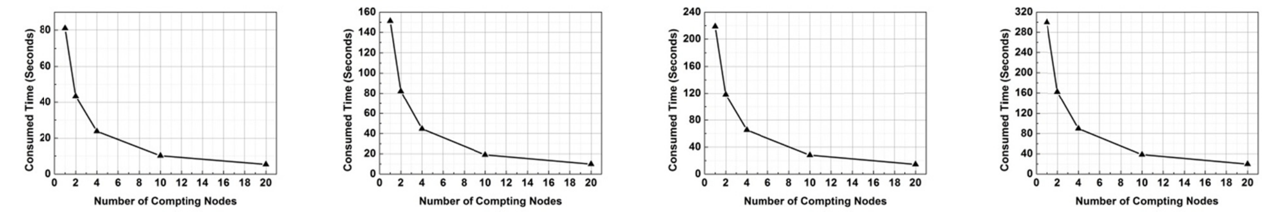
(a) D1.1 (b) D1.2 (c) D1.3 (d) D1.4

Fig. 4. MAE of base models with different initial hypothesis. All tested models adopt the first estimation rule and objective function in Table I.



(a) D1.1 (b) D1.2 (c) D1.3 (d) D1.4

Fig. 5. MAE of EAMFM with the number of base models increasing.



(a) D1.1 (b) D1.2 (c) D1.3 (d) D1.4

Fig. 6. Computing time of building a 20-model EAMFM with the number of computing nodes increasing.

their features are initialized with the same arrays whose elements are randomly generated in the scale of (0,0.04). For all models, the training process terminates when the training error arises or the decrease in training error between two consecutive iterations is less than 10^{-4}.

The results are depicted in Fig. 3. From them, we see that with different estimation rules and objective functions, the accuracy of base models varies; however, the difference is small. For instance, on testing case D1.1, Model 1 achieves the lowest MAE at 0.5359. In contrast, the MAE of Model 4 is the highest at 0.5385. The accuracy gap between them is 0.48% only.

Another interesting phenomenon is that with the training data increasing, the accuracy rank of base models with different estimation rules also varies, as depicted in Fig. 3. For instance, the MAE of Model 4 is the highest among all tested base models on testing case D1.1, but the lowest on testing case D1.4. This phenomenon in fact indicates the necessity to build an ensemble like our proposed one, which takes the effect by different estimation rules and objective functions into full consideration.

Afterwards, we have tested the effect of randomness injection on the accuracy of base models. For each testing case, the MAE of a base model with randomly initialized features in the scale of (0,0.004) in 20 independent runs is recorded. Fig. 4 depicts the MAE of Model 1 on D1 in such a test. Note that similar situations can be found in the same test for Model 2-4.

From Fig. 4, we see that the accuracy gap between models with different initial hypothesis is a bit larger than that caused by difference in training processes. For instance, on testing case D1.1, with different initial hypothesis, the highest MAE of Model 1 is 0.5379, and the lowest is 0.5316, indicating the accuracy gap at 1.17%. With more training data, this accuracy gap is shrunk. On testing case D1.4, the highest and lowest MAE of Model 1 are respectively 0.4302 and 0.4265, indicating a gap at 0.86% that is smaller than that on D1.1.

To summarize, diversifying mechanism adopted by EAMFM do not lead to loss in prediction accuracy of its base models.

4.3 Part B

This part aims to evaluate the estimation results of EAMFM as the number of base models increases from 1 to 20, to demonstrate its effectiveness in boosting the estimation accuracy. For each base model, the regularizing parameters are set as $\lambda_B=\lambda_C=\lambda_P=\lambda_E=0.1$. The results are depicted in Fig. 5. From these results, we have the following findings:

a) EAMFM has advantage in estimation accuracy over its base models. As depicted in Fig. 5, its MAE decreases with the number of base model increasing on all testing cases. For instance, on D1.1, EAMFM achieves the MAE at 0.5078 with 20 base models. Compared with that of a single base model at 0.5359, the improvement is 5.24%, which is quite significant.

b) The improvement in estimation accuracy by EAMFM is related with the amount of training data. In Table III, we have summarized the improvement in estimation accuracy by EAMFM on all testing cases. From Table III, we see that with more and more training data, the improvement generally gets smaller and smaller. On possible reason for this phenomenon is that with more training data, the resulting model generates its estimates nearer to the true values. Consequently, the accuracy gain by an ensemble decreases.

TABLE III
MAE OF EAMFM (WITH 20 BASE MODELS) AND AMF

Testing Case	EAMFM (20 models)	Single Base Model	Accuracy Gain by Ensemble
D1.1	0.5062	0.5359	5.54%
D1.2	0.4414	0.4722	6.52%
D1.3	0.4279	0.4471	4.29%
D1.4	0.4137	0.4272	3.16%

In summary, QoS estimates by EAMFM are more accurate than those by a single base model.

4.4 Part C

In this part we test the efficiency of parallelism in EAMFM. On each testing case of D1, we build an EAMFM model consisting of 20 base models, and record the consumed time. This process is repeated with the number of computational nodes increasing from 1 to 20. The corresponding results are depicted in Fig. 6.

It can be clearly seen from Fig. 6 that with the number of computing nodes increasing, the consumed time by building a 20-model EAMFM drastically decreases. For instance, on testing case D1.1, the consumed time is 101.7 seconds with 1 computing node, and 6.8 seconds with 20 computing nodes. However, the time is not linear with respect to the number of employed computing nodes. This is because when building an EAMFM model, the model aggregation via an AdaBoost-based algorithm cannot be parallelized, as depicted in Fig. 2.

4.5 Part D

This part compares the performance of the proposed EAMFM against three QoS estimators that are among the most effective ones to the best knowledge of the authors. They are summarized in Table IV. The experiment settings of all tested models are as follows:

a) For M1, the number of the nearest neighbors and the balancing coefficient directly decide the performance. We tune them together on all cases following the instructions in [5, 6];

b) For all MF-based QoS estimator, the dimension of the latent factor space, i.e., f, is set at 20. Meanwhile, to eliminate the effect of the random initial guess, we initialize each model with the same randomly-generated arrays;

c) For M2, the momentum is set at 0.9, and we tuned its step-size following the work in [15];

d) For M3, the regularizing coefficient is tuned for each case by following [19, 20]; and

e) For EAMFM, the regularizing parameters in its base models are set as $\lambda_B=\lambda_C=\lambda_P=\lambda_E=0.1$ on D1, and $\lambda_B=\lambda_C=\lambda_P=\lambda_E=6$ on D2. The number of base models n is set at 20.

TABLE IV
COMPARED MODELS IN EXPERIMENT PART III

Models	Description
M1	The QoS estimator proposed in [5, 6] that combines user-oriented and item-oriented S-KNN models.
M2	The QoS estimator based on probabilistic matrix factorization proposed in [15]. It is a sophisticated and highly accurate model in CF community.
M3	The QoS estimator based on regularized single-element-dependent non-negative matrix factorization proposed in [19, 20]. It takes the non-negative nature of QoS data into consideration.

The comparison results are recorded in Tables V and VI. From them, we conclude:

a) When compared with the other models, EAMFM is able to gain its significant advantage in estimation accuracy. On D1.1, its MAE is 0.5062, about 25.14% lower than 0.6762 by M1, 9.32% lower than 0.5582 by M2, and 6.91% lower than 0.5438 by M3. On D1.4, with denser training data the MAE of M1-M3 and EAMFM are 0.4589 and 0.4473, 0.4361 and 0.4137, respectively. The accuracy improvement to M1-M3 by EAMFM is 9.85%, 7.51%, and 5.14%, respectively. Similar results are also obtained in the experiments on D2, as recorded in Table V.

a) Due to its need for building nultiple base models, EAMFM indeed consumes much more time than its peers. However, owing to its ease of parallelization, its computing time can be reduced to be comparable with that of the other sophisticated QoS estimators. As recorded in Table VI, with only one computing node, to build a 20-model EAMFM consumes 81.0 seconds. This is 2.36, 19.76 and 13.28 times that of the time by UIPCC, PMF and RSNMF, respectively. However, with 20 computing nodes its time is drastically reduced to 5.4 seconds, which is clearly comparable with 4.1 and 6.1 seconds needed by PMF and RSNMF, and much lower than that 34.3 seconds needed by UIPCC. This indicates that the practicability of EAMFM can be greatly improved via its efficient parallelization.

TABLE V
MAE OF TESTED MODELS ON ALL CASES

Testing Case	UIPCC	PMF	RSNMF	EAMFM
D1.1	0.6762	0.5582	0.5438	0.5062
D1.2	0.5412	0.4931	0.4868	0.4414
D1.3	0.4993	0.4647	0.4593	0.4279
D1.4	0.4589	0.4473	0.4361	0.4137
D2.1	28.70	22.56	21.44	17.30
D2.2	24.35	17.64	17.21	15.04
D2.3	21.82	15.58	15.07	13.69
D2.4	20.34	14.76	14.33	13.35

TABLE VI
CONSUMED TIME CORRESPONDING TO TABLE IV (SECONDS)

Testing Case	UIPCC	PMF	RSNMF	EAMFM (1 node)	EAMFM (20 nodes)
D1.1	34.3	4.1	6.1	81.0	5.4
D1.2	71.2	5.2	10.7	151.2	9.9
D1.3	125.1	5.7	12.6	218.8	14.4
D1.4	153.9	6.3	15.7	300.2	19.8
D2.1	38.5	5.4	6.1	92.2	6.4
D2.2	86.7	7.2	12.4	174.7	11.3
D2.3	139.4	8,5	18.0	251. 3	16.5
D2.4	167.5	9.6	23.5	336. 8	21.9

Based on the above results and discussions, we summarize that EAMFM is able to provide highly accurate estimates to unknown QoS data based on historical ones. Its time cost is high due to the need of multiple base models; yet, it can be drastically reduced with parallel computing. Hence, it provides an effective scheme for industrial applications seeking for highly-accurate QoS estimates with available parallel computing facilities, e.g., cluster and cloud.

5 CONCLUSIONS

This work aims at designing an effective scheme for QoS estimation with a) high prediction accuracy, b) low time cost, and c) dependence on QoS data only. To do so, we firstly investigate the alternating least squares-based parameter training process for a matrix-factorization-based QoS estimator. By dividing this process into single-element-dependent subtasks, we subsequently integrate the principles of alternating direction method (ADM) into it, thereby obtaining the ADM-based parameter training scheme. The resulting model is further diversified and aggregated to achieve an Ensemble of ADM-based Matrix Factorization Model (EAMFM), which is able to provide highly accurate estimates to unknown QoS data. Through parallelization, its computiational time is drastically reduced to achieve the comparable speed as the best single-model QoS estimators. In the future work, we plan to apply the idea of EAMFM to other big data-related areas, including complex network analysis [41-43] and bioinformatics [42, 44].

REFERENCES

[1] L. Zeng, T. Watson, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, *et al.*, "QoS-aware Middleware for Web Services Composition," *IEEE Trans. on Software Engineering*, vol. 30, pp. 311-327, 2004.

[2] Y. Liu, A. Ngu, and L. Z. Zeng, "QoS Computation and Policing in Dynamic Web Service Selection," *in Proc. of the 13th Int. Conf. on World Wide Web*, New York, USA, 2004.

[3] Z.-B. Zheng, Y.-L. Zhang, and M. R. Lyu, "Distributed QoS Evaluation for Real-World Web Services," *in Proc. of the 17th IEEE Int. Conf. on Web Services*, Miami, Florida, USA, 2010, pp. 83-90.

[4] L.-S. Shao, J. Zhang, Y. Wei, J.-F. Zhao, B. Xie, and H. Mei, "Personalized QoS estimate for Web Services via Collaborative Filtering," *in Proc. of the 14th IEEE Int. Conf. on Web Services*, Salt Lake City, USA, 2007, pp. 439-446.

[5] Z.-B. Zheng, H. Ma, M. R. Lyu, and I. King, "WsRec: A Collaborative Filtering Based Web Service Recommender System," *in Proc. of the 16th IEEE Int. Conf. on Web Services*, Los Angeles, USA, 2009, pp. 437-444.

[6] Z.-B. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-Aware Web Service Recommendation by Collaborative Filtering," *IEEE Trans. on Services Computing*, vol. 4, pp. 140-152, 2011.

[7] J. Cao, Z.-A. Wu, Y.-Q. Wang, and Y. Zhuang, "Hybrid Collaborative Filtering Algorithm for Bidirectional Web Service Recommendation," *Knowledge and Information Systems*, pp. 1-21, 2012.

[8] Y.-L. Zhang, Z.-B. Zheng, and M. R. Lyu, "WSPred: A Time-Aware Personalized QoS estimate Framework for Web Services," in *Proc. of the 22nd IEEE Int. Symposium on Software Reliability Engineering*, 2011, pp. 210-219.

[9] W. Lo, J.-W. Yin, S.-G. Deng, Y. Li, and Z. Wu, "Collaborative Web Service QoS estimate with Location-Based Regularization," in *Proc. of the 19th IEEE Int. Conf. on Web Services*, 2012, pp. 464-471.

[10] Z.-B. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative Web Service QoS estimate via Neighborhood Integrated Matrix Factorization," *IEEE Trans. on Services Computing*, vol. 6, pp. 289-299, 2013.

[11] Y.-L. Zhang, Z.-B. Zheng, and M. R. Lyu, "Exploring Latent Features for Memory-Based QoS estimate in Cloud Computing," in *Proc. of the 30th IEEE Int. Symposium on Reliable Distributed Systems*, Madrid, Spain, 2011, pp. 1-10.

[12] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-art and Possible Extensions," *IEEE Trans. on Knowledge and Data Engineering*, vol. 17, pp. 734-749, 2005.

[13] Y. Koren and R. Bell, "Advances in Collaborative Filtering," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds., New York: Springer, 2011, pp. 145-186.

[14] M. Kurucz, A. Benczúr, and K. Csalogány, "Methods for Large Scale SVD with Missing Values," in *Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, California, USA, 2007, pp. 7-14.

[15] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1257–1264, 2008.

[16] G. Takács, I. Pilászy, Bottyán Németh, and D. Tikky, "Scalable Collaborative Filtering Approaches for Large Recommender Systems," *Journal of Machine Learning Research*, vol. 10, pp. 623-656, 2009.

[17] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *IEEE Computer*, vol. 42, pp. 30-37, 2009.

[18] X. Luo, Y.-N. Xia, and Q.-S. Zhu, "Incremental Collaborative Filtering Recommender Based on Regularized Matrix Factorization," *Knowledge Based Systems*, vol. 27, pp. 271-280, 2012.

[19] X. Luo, M.-C. Zhou, Y.-N. Xia, and Q.-S. Zhu, "An Efficient Non-negative Matrix-factorization-based Approach to Collaborative-filtering," *IEEE Trans. on Industrial Informatics*, vol. 10, pp. 1273 - 1284, 2014.

[20] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating Highly Accurate Predictions for Missing QoS Data via Aggregating Nonnegative Latent Factor Models," *IEEE Trans. on Neural Networks and Learning Systems*, DOI:10.1109/tnnls.2015.2412037.

[21] B. Stephen, P. Neal, C. Eric, P. Borja, and E. Jonathan, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, vol. 3, pp. 1-122.

[22] X. Luo, M.-C. Zhou, S. Li, Z.-H. You, Y.-N. Xia, and Q.-S. Zhu, "A Non-negative Latent Factor Model for Large-scale Sparse Matrices in Rec-

ommender Systems via Alternating Direction Method," *IEEE Trans. on Neural Networks and Learning Systems*, DOI:10.1109/TNNLS.2015.2415257.

[23] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-Scale Parallel Collaborative Filtering for the Netflix Prize," *Algorithmic Aspects in Information and Management*, Lecture Notes in Computer Science R. Fleischer and J. Xu, eds., pp. 337-348: Springer Berlin / Heidelberg, 2008.

[24] Y.-F. Hu, Y. Koren, and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets," in Proc. of the 8th IEEE Int. Conf. on Data Mining, Pisa, Italy, 2008, pp. 263-272.

[25] P. Paatero, and U. Tapper, "Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values," *Environmetrics,* vol. 5, no. 2, pp. 111-126, 1994.

[26] C.-J. Lin, "Projected Gradient Methods for Nonnegative Matrix Factorization," *Neural Computation,* vol. 19, no. 10, pp. 2756-2779, 2007.

[27] R. Maclin and J. Shavlik, "Combining the estimates of multiple classifiers: using competitive learning to initialize neural networks," in Proc. of the*14th Int. Joint Conf. on Artificial Intelligence*, Montreal, Quebec, Canada, 1995.

[28] Y. Freund, and R. Schapire, "A Decision-theoretic Generalization of On-line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, 1997.

[29] T. Dietterich, "Ensemble Methods in Machine Learning," in *Multiple Classifier Systems*. vol. 1857, ed: Springer Berlin/Heidelberg, 2000, pp. 1-15.

[30] Y. Freund, R. Iyer, R. Schapire, and Y. Singer, "An Efficient Boosting Algorithm for Combining Preferences," *Journal of Machine Learning Research,* vol. 4, pp. 933-969, 2003.

[31] D. Solomatine, and D. Shrestha, "AdaBoost.RT: a boosting algorithm for regression problems," *in Proc. of the 2004 IEEE Int. Joint Conf. on Neural Networks*, Budapest, Jul. 2004, pp. 1163-1168.

[32] A. Schclar, A. Tsikinovsky, L. Rokach, A. Meisels, and L. Antwarg, "Ensemble Methods for Improving the Performance of Neighborhood-based Collaborative Filtering," *in Proc. of the 3rd ACM Conf. on Recommender Systems*, New York, NY, Oct. 2009, pp. 261-264.

[33] X.-Y. Su and T. M. Khoshgoftaar, "A Survey Of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1-19, 2009.

[34] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review,* vol. 33, pp. 1-39, 2010.

[35] A. Bar, L. Rokach, G. Shani, B. Shapira, and A. Schclar, "Improving Simple Collaborative Filtering Models Using Ensemble Methods," in *Proc. of the 11th Int. Workshop on Multiple Classifier Systems*, Nanjing, JS, May. 2013, pp. 1-12.

[36] X.-T. Jiang, Z.-D. Niu, J.-M. Guo, G. Mustafa, Z.-H. Lin, B.-M. Chen, and Q. Zhou, "Novel Boosting Frameworks to Improve the Performance of Collaborative Filtering," *JMLR: Workshop and Conference Proceedings,* vol. 29, pp. 87-99, 2013.

[37] Y.-H. Wang, H.-L. Sun, and R.-C. Zhang, "AdaMF: Adaptive Boosting Matrix Factorization for Recommender System," in *Proc. of the 15th Int. Conf. on Web-Age Information Management*, Macau, Jun. 2014, pp. 43-54.

[38] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. on Information Systems,* vol. 22, pp. 5-53, 2004.

[39] G. Shani and A. Gunawardana, "Evaluating Recommendation Systems," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds., ed New York: Springer US, 2011, pp. 257-297.

[40] M. Yu, and M.-C. Zhou, "A Performance Modeling Scheme for Multi-stage Switch Networks With Phase-Type and Bursty Traffic," *IEEE/ACM Transactions on Networking,* vol. 18, no. 4, pp. 1091-1104, 2010.

[41] X. Cao, X. Wang, D. Jin, Y. Cao, and D. He, "Identifying overlapping communities as well as hubs and outliers via nonnegative matrix factorization," *Scientific Reports*, vol. 3, pp. 2993, 2013.

[42] C.-J. Jiang, H.-C. Sun, Z.-J. Ding, P.-W. Wang, and M.-C. Zhou, "An Indexing Network: Model and Applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 44, no. 12, pp. 1633-1648, 2014.

[43] X. Luo, Z. You, M. Zhou, S. Li, H. Leung, Y. Xia, and Q. Zhu, "A Highly Efficient Approach to Protein Interactome Mapping Based on Collaborative Filtering Framework," *Scientific Reports,* vol. 5, pp. 7702, 2015.