24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Temporal Web Service QoS Prediction via Kalman Filter-Incorporated Latent Factor Analysis

Ye Yuan[1, 2] and **Mingsheng Shang**[1] and **Xin Luo**[1, 3]

Abstract. With the rapid development of services computing in the past decade, automatic selection of QoS-aware Web service out from numerous candidates with similar functions is becoming a hot yet thorny issue. Conducting warming up tests on numerous candidate services for quality evaluation is extremely time-consuming and expensive, making it vital to generate highly accurate predictions for missing QoS data based on known ones. Since QoS data are time-dependent, it is vital to consider the temporal dynamic patterns hidden in historical ones when building a QoS-estimator. For addressing this issue, this study invents a Kalman filter-incorporated Latent Factor Analysis (KLFA)-based QoS-estimator, which precisely models the temporal patterns hidden in dynamic QoS data. Experimental results based on large-scale and real-world Web service QoS data demonstrate that compared with state-of-the-art temporal-aware QoS-estimators, a KLFA-based one achieves significantly higher prediction accuracy for missing QoS data.

1 INTRODUCTION

Cloud computing facilities are indispensable for big data-related industrial applications [1], with web services being fundamental components for implementing service-oriented architecture-based systems and softwares. They are so desired by most industrial Web applications, that numerous Web-services with similar functions are available online. Consequently, how to implement efficient service selection from a large candidate set becomes a major yet thorny issue in the area of cloud computing [2, 3].

Most non-functionality of a Web-service is reflected by its Quality-of-service (QoS), making efficient QoS-based approaches for service selection [4, 5] highly feasible, which take QoS data as the fundamental data source. Commonly, QoS data are retrieved from both servers and users. Server-side data like price and invoking protocol are provided by service providers. On the other hand, user-side data like response-time and throughput are highly related to many user-dependent factors. Although user-side QoS data can be obtained by real-world service invocations, but it suffers the following defects: a) such ‘warming up’ tests lead to high expenses because most online services are not free to invoke; and b) with a drastically increasing number of candidate services, it costs long to evaluate all of them. Due to these defects mentioned above, it becomes infeasible to acquire all QoS data through real invocations, yielding the critical problem of missing QoS data estimation, i.e., how to predict unknown QoS data precisely based on historical QoS invocations?

Various approaches are proposed to address this issue, where latent factor analysis (LFA)-based approaches [6-9, 13, 14] are getting increasingly attractive. An LFA-based QoS estimator takes a user-service QoS matrix as the fundamental data source, where each row denotes a specified user, each column denotes a specified service, and each element denotes the QoS record of a certain type experienced by a specified user on a specified service. Obviously, a user cannot invoke all candidate services or a service cannot be touched by all users, making this matrix highly sparse with numerous unknown yet desired data. Hence, an LFA-based estimator tries to handle this user-service matrix, extracting its essential factors for predicting its unobserved data accurately.

To do so, an LFA-based estimator maps both the users and services into the same low-dimensional latent factor (LF) space to build a low-rank approximation to the target user-service matrix for predicting its missing data. Such a process is defined on the known data of the target matrix only and implemented through various optimization techniques [10-12]. The resultant approximation matrix can be full where the missing data are estimated based on the information hidden in the known ones. With a well-established model and carefully-chosen learning algorithms, an LFA-based estimator can predict missing QoS data well with high computational efficiency. Recently, researchers have paid a lot of efforts on investigating LFA-based QoS-estimators, thereby achieving several sophisticated models.

Related works. Lo et al. [15] propose to extend an LFA-based QoS-estimator via incorporating the location information in each historical QoS record. It can outperform an LFA-based QoS estimator with additional data sources. Zheng et al. [11] propose an LFA-based QoS-estimator which considers the neighborhood information. Nonetheless, it costs much memory to store the pair-wise neighborhood factor of each user/service in real applications. Luo et al. [8] propose to ensemble a set of diversified non-negative latent factor analysis-based QoS-estimators to achieve a highly accurate one. For improving the time efficiency of the resultant model, Luo et al. [12] further propose an ensemble of alternating direction method-based LFA models, where the training process of

[1] Chongqing Engineering Research Center of Big Data Application for Smart Cities, and Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China, e-mail: {yuanye, msshang, luoxin21}@cigit.ac.cn (X. Luo is the corresponding author)

[2] University of Chinese Academy of Sciences, Beijing 100049, China

[3] Hengrui (Chongqing) Artificial Intelligence Research Center, Department of Big Data Analyses Techniques, Cloudwalk, Chongqing 401331, China

each single model is accelerated following the principle of an alternating direction method.

In spite of their efficiency, the aforementioned LFA-based QoS-estimators are static models without considering the temporal dynamic patterns hidden in QoS data. Note that in real applications, the QoS data retrieved from user-side are highly sensitive to many user-dependent factors (e.g., networking environment and geographic location), which results in the temporal dynamics. Hence, it becomes a vital task to explore LFA-based QoS-estimators which are aware of temporal dynamics hidden in the QoS data. Koren [17] propose the timeSVD++ model, which models the temporal effects as additional LFs based on the time slot of given data. However, it can only account for drift, meaning that it can move from a central point. In fact, this constraint results in overfitting an incorrect model. Sahoo et al. [18] propose to model the dynamic transition of target data with a hidden Markov model (HMM) approach. These models can incorporate dynamic information into existing LF models, but it is not suitable for long-term prediction and hard to completely describe the temporal patterns in target data. Zhang et al. [16] employ tensor factorization (TF) technique with average QoS value constraints to improve the prediction accuracy for missing QoS data. Zhang et al. [19] propose a non-negative tensor factorization (NTF)-based QoS-estimator. An NTF-based model can extract temporal patterns from dynamic QoS data. However, it suffers accuracy loss when the temporal patterns fluctuate frequently because it is in nature a pure optimization-based approach without specific design for describing the temporal patterns.

Contributions. This paper incorporates the principle of Kalman filter [20] into an LFA-based model, thereby implement an LFA-based QoS estimator. Given a sparse user-service-time tensor, we first adopt a Kalman filter to track the temporal variations of user LFs, thereby obtain filter-user LFs which incorporate the temporal patterns. And then we train service LFs based on filter-user LFs with alternating least squares (ALS). Through such a design, we are able to precisely model the temporal patterns hidden in dynamic QoS data, thereby achieving high prediction accuracy for missing QoS data. The main contributions of this paper include:

a) A KLFA-based QoS-estimator, which precisely models the temporal patterns hidden in dynamic QoS data;
b) Algorithm design and analysis for a KLFA-based QoS-estimator;
c) Empirical studies conducted on the WS-Dream datasets [21, 22] where several state-of-the-art temporal-aware QoS-estimators are included for demonstrating the efficiency of the proposed KLFA-based QoS-estimator.

According to the authors' best knowledge, such efforts have been never seen in any previous work.

The rest of this paper is organized as follows. Section 2 gives the problem statement. Section 3 presents a KLFA-based QoS-estimator. Section 4 gives the experimental results. Section 5 is the discussion. Finally, Section 6 concludes the paper.

2 PRELIMINARIES

In this paper we consider a dynamic scene as depicted in Figure 1. As the invoking time varies, the QoS data experienced by a specified user on a specified service is temporal dynamics. Thus, we can naturally adopt a three-dimensional tensor to describe the relationship among users, services and time, which is the fundamental data source for our problem, as depicted in Figure 2.

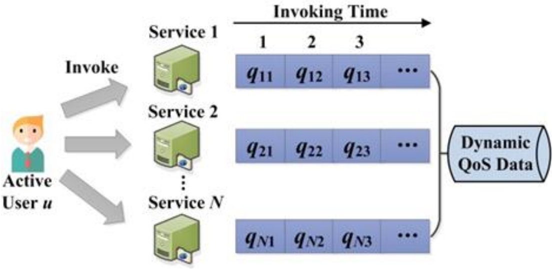


Figure 1. Collecting dynamic QoS data.

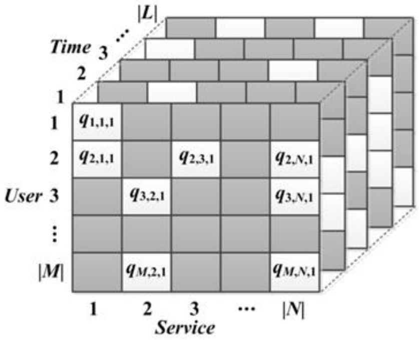


Figure 2. User-service-time tensor.

Given a user set M, service set N and time set L, we define this user-service-time tensor as follows:

***Definition* 1. User-service-time Tensor *Q*.** Q is a $|M|\times|N|\times|L|$ tensor where each element $q_{i,j,t}$ denotes the service QoS of the specified characteristic on service j experienced by user i at time interval t, where user $i\in\{1, 2, \ldots |M|\}$, service $j\in\{1, 2, \ldots |N|\}$, and time interval $t\in\{1, 2, \ldots |L|\}$.

As mentioned before, at each time interval, a service cannot be touched by all users, and a user cannot invoke all candidate services. Therefore, Q is a High Dimensional and Sparse (HiDS) tensor [28] with most QoS data unknown, as depicted in Figure 2. Let Q_K and Q_T denote its known and missing entry sets, respectively and we have $|Q_K|\ll|Q_T|$.

***Definition* 2. Temporal-aware QoS-estimator $\hat{Q}$.** Given Q, $\hat{Q}$ is built on Q_K. For each missing QoS record $q_{i,j,t}$, it can generate a prediction $\hat{q}_{i,j,t}$ based on information in Q_K and $\sum_{i,j,t\in Q_K}\left|q_{i,j,t}-\hat{q}_{i,j,t}\right|$ is minimized.

3 KLFA-BASED QOS ESTIMATOR

Table 1. Notations and Descriptions.

Notations	Descriptions
s_j	LFs of service j
$u_{i,t}$	LFs of user i at time interval t
$F_{i,t}$	transition process matrix of user i at time interval t
$H_{i,t}$	observation process matrix of user i at time interval t
$w_{i,t}$	transition Gaussian noise vector of user i at time interval t
$v_{i,t}$	observation Gaussian noise vector of user i at time interval t
$q_{i,t}$	The set of QoS values observed by user i at time interval t
$P_{i,t}$	covariance matrix of $u_{i,t}$
$K_{i,t}$	Kalman gain matrix of user i at time interval t
U_0	initial user LF matrix
U	filter-user LF matrix
S	service LF matrix

Kalman filter is a recursive estimation algorithm in control theory

[20], which is widely used in various fields for dynamic estimation. It is able to estimate the current state based on the previous state accurately.

As mentioned before, although state-of-the-art temporal-aware QoS-estimators have been shown to be successful in practice, they are limited in extracting temporal patterns from dynamic QoS data. For addressing this issue, we introduce a KLFA-based QoS-estimator, which precisely models the temporal patterns hidden in dynamic QoS data without considering the data's variation rule and frequency. Before clarifying the KLFA-based QoS-estimator, some basic notations are introduced at first in Table 1.

For each user $i\in\{1, 2, \ldots |M|\}$, the proposed KLFA-based QoS-estimator obtains the filter-user LFs $u_{i,|L|}$ through $|L|$ times Kalman filter. For each service $j\in\{1, 2, \ldots |N|\}$, we train service LFs s_j based on filter-user LFs $u_{i,|L|}$ with alternating least squares (ALS).

Moreover, let R is a $|M|\times|N|$ HiDS matrix which denotes a time slice of Q. The rank-f estimate to R is able to denoted by $\hat{R}=US$ where $U=[u_{1,|L|}, \ldots, u_{|M|,|L|}]^{\mathrm{T}}$ and $S=[s_1, \ldots, s_{|N|}]$. Note that $U_{|L|}$ has dimension $|M|\times f$, S has dimension $f\times|N|$ and $f\ll\min\{|M|,|N|\}$. From the above analysis, the prediction is only for a time slice of Q. However, the obtained filter-user LFs $u_{i,|L|}$ incorporate the temporal patterns hidden in dynamic QoS data over the whole time frame. Hence, we can extend the matrix $\hat{R}$ to the tensor $\hat{Q}$, where each slice of $\hat{Q}$ is $\hat{R}$ and $\hat{q}_{i,j,t}=u_{i,|L|}s_j$.

Therefore, to represent the given user-service-time tensor Q, KLFA-based QoS-estimator builds its estimator $\hat{Q}$. Since Q is an HiDS tensor, U and S are built based on Q_K only. To do so, the objective function with Euclidean distance is formulated as follows:

$$\underset{U,S}{\arg\min}\,\varepsilon_{KLFA}\;U,S\;=\sum_{i,j,t\in Q_K}\left(C\;q_{i,j,t}-u_{i,|L|}s_j\;^2+\left\|u_{i,|L|}\right\|_F^2+\left\|s_j\right\|_F^2\right)\qquad(1)$$

where $\|\cdot\|_F$ computes the Frobenius norm, the constant C controls the regularization effect [23].

Note that the objective function (1) is generally the same as one in a static LFA-based QoS-estimator [10-12]. However, our model is sharply different from a static LFA-based QoS-estimator in the following aspects:

a) The fundamental data source of KLFA is an HiDS tensor, not an HiDS matrix.

b) The user LFs in KLFA incorporate the temporal patterns hidden in dynamic QoS data, which are not standard static LFs.

Moreover, KLFA is also essentially different from tensor factorization approaches [19, 33] which directly decompose the target QoS tensor into a set of matrices: user LFs, service LFs, and time LFs matrices, respectively. For KLFA, we just incorporate temporal patterns into user LFs without extra time LFs matrix.

Next, we show how to build a KLFA-based QoS-estimator.

3.1 Coupled Dynamical System

To model the temporal patterns hidden in dynamic QoS data, a coupled dynamical system of user LFs needs to be proposed first, whose posteriori estimate can be obtained using Kalman filter. This system can be represented by two stochastic equations (2) and (3).

First, we assume that user LFs are function of time and make $u_{i,t}$ denote the LFs of user i at time interval t. For each user, the initial user LFs matrix $U_0=[u_{1,0}, \ldots, u_{|M|,0}]^{\mathrm{T}}$ is randomly. The user LFs evolution is according to the generally non-stationary transition process matrix $F_{i,t}$ and transition noise $w_{i,t}$ to capture dynamic variations along the time frame. Taken together, the dynamic variation process is described as follows:

$$u_{i,t}=F_{i,t}u_{i,t-1}+w_{i,t}\qquad(2)$$

where equation (2) is called a state process equation and represents the internal state of the system. The transition noise $w_{i,t}$ is independent, zero-mean, Gaussian noise with distributions: $p(w_{i,t})\sim N(0, W_{i,t})$ and $W_{i,t}$ is the covariance matrix. Note that in real application, although the QoS data is always in a state of dynamic change, there is no sudden change in magnitude. Hence, in this context, we can set $F_{i,t}$ to be a diagonal matrix which diagonal elements are random values close to 1 even equal to 1.

Meanwhile, we assume that the service LFs evolve very slowly, which can be considered constant over the time frame that user LFs are collected. Also, due to a limited QoS values of Web services number are observed by each user, we incorporate the service LFs through a non-stationary observation process matrix $H_{i,t}$, which is made up of subsets of rows of the service LF matrix S observed at time t by user i, e.g., if user i observes service 1, 3 and 5 at time t, so we select s_1, s_3 and s_5 to compose the observation matrix $H_{i,t}$. Note that all $H_{i,t}$ are subsets of the same fixed S and are coupled in this way. We also include observation noise and the overall observation equation is:

$$q_{i,t}=H_{i,t}u_{i,t}+v_{i,t}\qquad(3)$$

where equation (3) is called an observation process equation and outputs observation on the basis of the state of the system. The observation process matrix $H_{i,t}$ maps user LFs $u_{i,t}$ to observed QoS values $q_{i,t}$. Moreover, $v_{i,t}$ is independent, zero-mean, Gaussian noise with distributions: $p(v_{i,t})\sim N(0, V_{i,t})$ and $V_{i,t}$ is the covariance matrix. The product $H_{i,t}u_{i,t}$ in (3) parallels the product $<u_i, s_j>$ in equation (1). We use $q_{i,t}$ to denote the QoS values observed by user i at time interval t, corresponding to the know entries of Q_K.

3.2 U Step via Kalman Filter

According to the nature of Kalman filter [20], we run M independent Kalman filter to exploit dynamic variations of user LFs for time interval $t\in\{1, 2, \ldots |L|\}$. The Kalman filter performs two steps: the time prediction step and the feedback update step.

Time prediction step. First, we need to use the state process equation (2) to predict the priori estimate for user LFs at time interval t from the posteriori estimate for user LFs at time interval t-1. That means it is utilized to project forward both the current posteriori estimate to acquire the priori estimate for the next time interval. The time prediction step can be formulated as follows:

$$u_{i,t}^{-}=F_{i,t}u_{i,t-1}\qquad(4)$$

$$P_{i,t}^{-}=F_{i,t}P_{i,t-1}F_{i,t}^{\mathrm{T}}+W_{i,t}\qquad(5)$$

where T denotes the matrix transpose. $u_{i,t}^{-}$ represents a priori user LFs at time interval t and its priori covariance matrix is $P_{i,t}^{-}$. $u_{i,t}$ is a posteriori user LFs at time interval t and its posteriori covariance matrix is $P_{i,t}$.

Feedback update step. Now, we obtain the priori estimate for the time interval t according to time prediction step. And then, we rectify the priori estimate based on the observation process equation (3) to get the posteriori estimate for the time interval t. That means the feedback update step can be seen as a modification of priori estimate based on observed values. The feedback update step can be formulated as follows:

$$e_{i,t}=q_{i,t}-H_{i,t}u_{i,t}^{-}\qquad(6)$$

$$E_{i,t}=V_{i,t}+H_{i,t}P_{i,t}^{-}H_{i,t}^{\mathrm{T}} \tag{7}$$

$$K_{i,t}=P_{i,t}^{-}H_{i,t}^{\mathrm{T}}E_{i,t}^{-1} \tag{8}$$

$$P_{i,t}=\ I-K_{i,t}H_{i,t}\ \ P_{i,t}^{-} \tag{9}$$

$$u_{i,t}=u_{i,t}^{-}+K_{i,t}e_{i,t} \tag{10}$$

where I is the unit matrix, $e_{i,t}$ and $E_{i,t}$ represent observation residuals and covariance of the observation residuals, respectively. Based on (6)-(10), we obtain the posteriori estimate $u_{i,t}$ and $P_{i,t}$ at time interval t. $u_{i,t}$ is the optimal estimate LFs for user i at time t. The posteriori estimate can be used not only as the optimal estimate of this time interval, but also as the input of the next time interval to keep the Kalman Filter running along the time.

For each user $i\in\{1, 2, \ldots |M|\}$, the filter-user LFs $u_{i,|L|}$ are obtained through $|L|$ times Kalman filter. Hence, the filter-user LFs $u_{i,|L|}$ incorporate the temporal patterns hidden in dynamic QoS data over the whole time frame. The process of modeling the temporal patterns via Kalman filter is depicted in Figure 3.

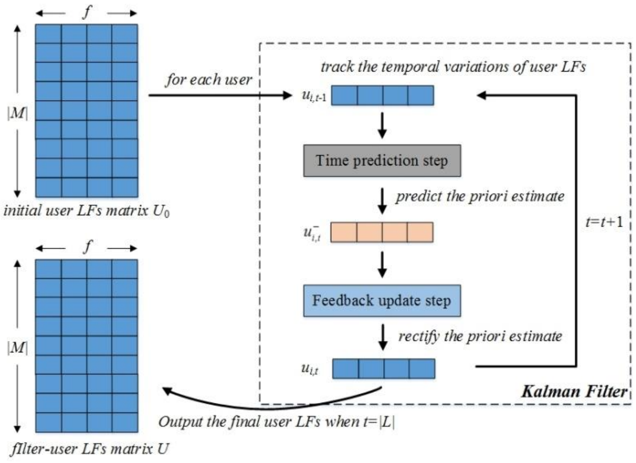

f
|M|
initial user LFs matrix U_0
for each user
track the temporal variations of user LFs
$u_{i,t-1}$
Time prediction step
predict the priori estimate
$u_{i,t}^{-}$
Feedback update step
rectify the priori estimate
$u_{i,t}$
t=t+1
Kalman Filter
f
|M|
Output the final user LFs when t=|L|
fIlter-user LFs matrix U

Figure 3. Process of modeling the temporal patterns via Kalman filter.

3.3 S Step via Alternating Least Squares

Based on the Kalman filter, the filter-user LFs with temporal patterns are estimated. Note that objective function (1) relies on Q_K along with related row and column vectors from U and S, respectively. Hence, we solve (1) depending on each column vector of S by denoting $S=[s_1, \ldots, s_{|N|}]$. Naturally, since each $u_{i,|L|}$ is obtained through Kalman filter, equation (1) is analytically solvable with respect to each single service LFs s_j with $j\in\{1, 2, \ldots |N|\}$. The derivative of ε_{KLFA} with respect to s_j is:

$$\frac{\partial\varepsilon_{KLFA}}{\partial s_j}=2s_j\left|Q_K(j)\right|-2C\sum_{i,j,t\ \in Q_K\ \ j}u_{i,|L|}^{\mathrm{T}}q_{i,j,t}+2C\sum_{i\in M\ \ j}u_{i,|L|}^{\mathrm{T}}u_{i,|L|}s_j \tag{11}$$

Note that in (11), $M(j)$ denotes the subset of M where each user $i\in M(j)$ corresponds to a unique instance $q_{i,j,t}\in Q_K(j)$. To achieve a local minimum of (1), we set (11) at zero to achieve the following expression:

$$\sum_{i,j,t\ \in Q_K\ \ j}u_{i,|L|}^{\mathrm{T}}q_{i,j,t}=\left(\frac{\left|Q_K(j)\right|}{C}I+\sum_{i\in M\ \ j}u_{i,|L|}^{T}u_{i,|L|}\right)s_j \tag{12}$$

As discussed in the previous work [23, 24], the condition $f\ll\min\{|M|, |N|\}$ is always fulfilled. Hence, $\sum_{i\in M\ \ j}u_{i,|L|}^{T}u_{i,|L|}$ is full-rank, making the following solution to (12) exist:

$$s_j=\left(\frac{\left|Q_K(j)\right|}{C}I+\sum_{i\in M\ \ j}u_{i,|L|}^{\mathrm{T}}u_{i,|L|}\right)^{-1}\sum_{i,j,t\ \in Q_K\ \ j}u_{i,|L|}^{\mathrm{T}}q_{i,j,t} \tag{13}$$

with (13), we solve S based on U.

3.4 Algorithm Design and Analysis

Algorithm: KLFA

```
Input: $Q_K$, $M$, $N$, $L$, $f$, $C$
Operation                                                           Cost
Initialize: $U_0=[u_{1,0},..,u_{|M|,0}]^{\mathrm{T}}\in R^{|M|\times f}$ randomly    $\Theta(|M|\times f)$
Initialize: $S=[s_1,..,s_{|N|}]\in R^{f\times|N|}$ randomly                $\Theta(|N|\times f)$
Initialize: $ROUND$ = max_round， $n = 0$                             $\Theta(2)$
while not converge && $n\le ROUND$ do                                $\times n$
  for $i$=1 to $|M|$                                                  $\times|M|$
    initialize $P_{i,0}\in R^{f\times f}$ all zeros                       $\Theta(f^2)$
    for $t$=1 to $|L|$                                                $\times|L|$
      set $W_{i,t}\in R^{f\times f}$ known                                $\Theta(f^2)$
      set $V_{i,t}\in R^{|N(i,t)|\times|N(i,t)|*}$ known                    $\Theta(|N(i,t)|^2)$
      set $E_{i,t}\in R^{|N(i,t)|\times|N(i,t)|}$ zeros                     $\Theta(|N(i,t)|^2)$
      set $F_{i,t}\in R^{f\times f}$ diagonal matrix                       $\Theta(f^2)$
      set $H_{i,t}\in R^{|N(i,t)|\times f}$ subsets of fixed $S$             $\Theta(|N(i,t)|\times f)$
      set $K_{i,t}\in R^{f\times|N(i,t)|}$ zeros                          $\Theta(|N(i,t)|\times f)$
      set $e_{i,t}\in R^{|N(i,t)|}$ zeros                                 $\Theta(|N(i,t)|)$
      $u_{i,t}^{-}=F_{i,t}u_{i,t-1}$                                      $\Theta(f^2)$
      $P_{i,t}^{-}=F_{i,t}P_{i,t-1}F_{i,t}^{\mathrm{T}}+W_{i,t}$                    $\Theta(2\times f^3)$
      $e_{i,t}=q_{i,t}-H_{i,t}u_{i,t}^{-}$                                 $\Theta(|N(i,t)|\times f)$
      $E_{i,t}=V_{i,t}+H_{i,t}P_{i,t}^{-}H_{i,t}^{\mathrm{T}}$                     $\Theta(|N(i,t)|^2\times f+$
                                                                    $|N(i,t)|\times f^2)$
      $K_{i,t}=P_{i,t}^{-}H_{i,t}^{\mathrm{T}}S_{i,t}^{-1}$                        $\Theta(|N(i,t)|^2\times f+$
                                                                    $|N(i,t)|\times f^2+|N(i,t)|^3)$
      $u_{i,t}=u_{i,t}^{-}+K_{i,t}e_{i,t}$                                 $\Theta(|N(i,t)|\times f)$
      $P_{i,t}=\ I-K_{i,t}H_{i,t}\ \ P_{i,t}^{-}$                             $\Theta(|N(i,t)|\times f^2+f^3)$
    end for                                                          --
  end for                                                            --
  for $j$=1 to $|N|$                                                  $\times|N|$
    set $C\in R^{f\times f}$ zeros                                       $\Theta(f^2)$
    set $d\in R^{f}$ zeros                                              $\Theta(f)$
    for $I\in M(j)^{**}$                                               $\times|Q_K(j)|$
      $C = C+u_{i,|L|}^{\mathrm{T}}u_{i,|L|}$                                 $\Theta(f^2)$
      $d = d+u_{i,|L|}^{\mathrm{T}}q_{i,j,t}$                                 $\Theta(f)$
    end for                                                          --
    $s_j=\left(\frac{|Q_K(j)|}{C}I+C\right)^{-1}d$                          $\Theta(f^3+f^2)$
  end for                                                            --
  $n = n+1$                                                          $\Theta(1)$
  set $U_0=[u_{1,|L|},..,u_{|M|,|L|}]^{\mathrm{T}}$                        $\Theta(|M|\times f)$
```

```
end while                                          --
Output: filter-user LFs and service LFs
```

* $|N(i, t)|$ represents the number of Qos values observed by user i on time t.

** $M(j)$ represents the set of users who observe service j over the time frame.

Based on the above inference, we design Algorithm KLFA. As discussed in Sections 3.2 and 3.3, each iteration of KLFA-based QoS-estimator consists of two phases: a) obtain the filter-user LFs via Kalman filter to explore the dynamics of QoS attributes and model the temporal patterns; b) executing the S step to solve service LFs based on user LFs and (13).

As given in Algorithm KLFA, its computational cost consists of the following parts:

$$\begin{cases} \text{(1). Initialization} \approx \Theta\ 2|M|\times f\ +\Theta\ |N|\times f \\ \text{(2). } U \text{ step} \approx n\cdot\Theta\left(\ |M|\times|L|\ \times\left(\begin{matrix} 3f^3+2|N(i,t)|^2\times f \\ +3|N(i,t)|\times f^2+|N(i,t)|^3 \end{matrix}\right)\right) \\ \text{(3). } S \text{ step} \approx n\cdot\Theta\ \ f^2+f\ \times\ |Q_K|+|N|\times f \end{cases} \tag{14}$$

Based on (14), the computational cost of KLFA is:

$$T \approx n\cdot\Theta\left(\begin{matrix} |M|\times|L|\ \times\left(\begin{matrix} 3f^3+2|N(i,t)|^2\times f \\ +3|N(i,t)|\times f^2+|N(i,t)|^3 \end{matrix}\right) \\ +\ f^2+f\ \times\ |Q_K|+|N|\times f \end{matrix}\right) \approx \Theta\left(n\times\frac{|Q_K|^3}{|M|\times|L|}\right) \tag{15}$$

Note that in (15), we take advantage of the inference that $|M|\times|L|\times|N(i,t)|=Q_K$. In real applications, f is a positive constant and can be set small without impairing an LF model's performance. Therefore, we usually have $f \ll |N(i,t)|$. Moreover, we reasonably omit the lower-order-terms to achieve the final result.

4 EXPERIMENTAL RESULTS

4.1 General Settings

Evaluation Protocol. We focus on the accuracy of QoS prediction, since it directly reflects whether or not the model has captured the essential characteristics of given data. Hence, we adopt it as the evaluation protocol in our experiments. A model's prediction accuracy is measured by root mean squared error (RMSE) [30]:

$$RMSE=\sqrt{\left(\sum_{(i,j,t)\in Q_T}\left(q_{i,j,t}-\hat{q}_{i,j,t}\right)^2\right)\Big/|Q_T|} \tag{16}$$

where Q_T denotes the test set and is disjoint with Q_K, $q_{i,j,t}$ is the QoS value of Web service j observed by user i at time interval t, $\hat{q}_{i,j,t}$ denotes the prediction for the testing instance, and $|\cdot|_{abs}$ computes the absolute value of a given number, respectively. All experiments are conducted on a Tablet with a 2.5 GHz E5-2680 V4 CPU and 500 GB RAM. The programming language is JAVA SE 7U60.

Table 2. Details of datasets.

Dataset	D1	D2
Type	Response-time	Throughput
Num. of Users	142	142
Num. of Web services	4,500	4,500
Num. of time intervals	64	64
$\|Q_K\|$	30,171,491	30,286,687

Datasets. To evaluate the KLFA-based QoS-estimator, we do the experiments on two large-scale real-world Web service QoS datasets collected by the WS-Dream system [21, 22]. These two datasets consist of the response-time and throughput of the invocations on 4,500 real-world WSs by 142 users during 64 time intervals. Their details are summarized in Table 2.

We randomly select the service 1 observed by user 5 and service 7 observed by user 6 from the Throughput dataset. As shown in Figure 4, a user observes different QoS values on the same Web service during different time intervals. This observation indicates the temporal of QoS values.

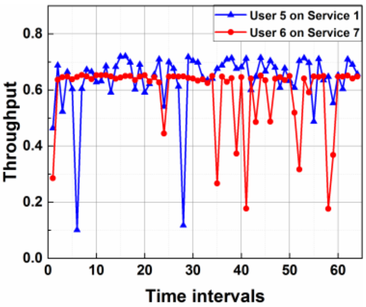


Figure 4. Throughput of Two Pairs of User-Service.

For D1 and D2, we design different testing cases under different data ratios, as shown in Table 3. The column Train: Test means the ratio of training to testing data; e.g., 10%:90% denotes 10% of given data is chosen randomly as training data to predict the remaining 90% data. Naturally, testing data are not involved in the training process. Cross-validation techniques are employed to obtain more objective results.

Table 3. Testing cases.

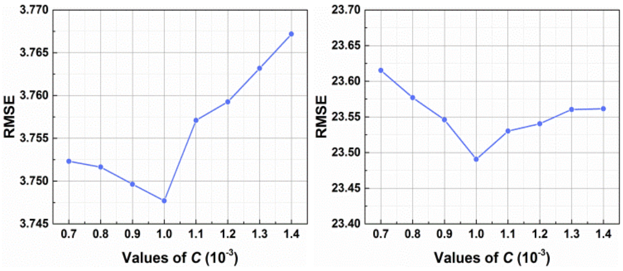
Dataset	No.	Train:Test	Training data	Testing data
D1	D11	5%:95%	1,508,574	28,662,917
	D12	10%:90%	3,017,149	27,154,342
	D13	20%:80%	6,034,298	24,137,193
	D14	40%:60%	12,068,596	18,102,895
	D15	50%:50%	15,085,745	15,085,746
D2	D21	5%:95%	1,514,334	28,772,353
	D22	10%:90%	3,028,668	27,258,019
	D23	20%:80%	6,057,337	24,229,350
	D24	40%:60%	12,114,674	18,172,013
	D25	50%:50%	15,143,343	15,143,344

4.2 Parameter-Sensitive Tests

From Section 3, we can see that the performance of KLFA depends on the parameters C and f, which decide the regularization effect and LF dimension, respectively. Moreover, the covariance matrix $W_{i,t}$ of transition noise $w_{i,t}$ and the covariance matrix $V_{i,t}$ of observation noise $v_{i,t}$ in Kalman filter are known normally [29]. Learning the $W_{i,t}$ and $V_{i,t}$ is difficult in practice from such limited observations, but simplifications to process models yield tractable closed-form solutions. These simplifications are that $W_{i,t}$ is $\sigma_w I$, $V_{i,t}$ is $\sigma_v I$. Therefore, it is essential to investigate KLFA's performance as C, f, σ_w and σ_v vary. Figures 5-8 depict the results measured by RMSE as parameters vary on D11 and D21. The similar conclusions can be reached on other testing cases. From them, we have the following findings:

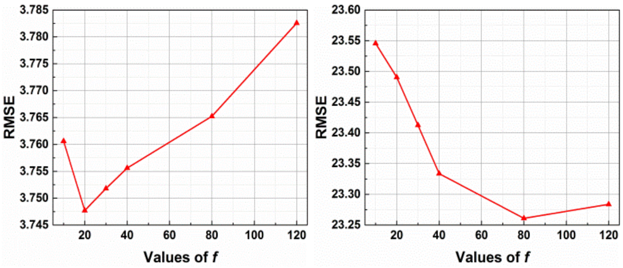
a) First, we fix f, σ_w and σ_v to investigate KLFA's performance as C varies. As shown in Figure 5(a), on D11, the lowest RMSE is 3.7477 with C=1.0×10^{-3}, 0.12% lower than 3.7523 with

$C=0.7\times10^{-3}$, 0.11% lower than 3.7518 with $C=0.8\times10^{-3}$, 0.05% lower than 3.7496 with $C=0.9\times10^{-3}$, 0.25% lower than 3.7571 with $C=1.1\times10^{-3}$, 0.31% lower than 3.7592 with $C=1.2\times10^{-3}$, 0.41% lower than 3.7631 with $C=1.3\times10^{-3}$, and 0.51% lower than 3.7671 with $C=1.4\times10^{-3}$. Similar situations can also be found on D21. It demonstrates that regularization effect is important for KLFA to achieve high-prediction accuracy.



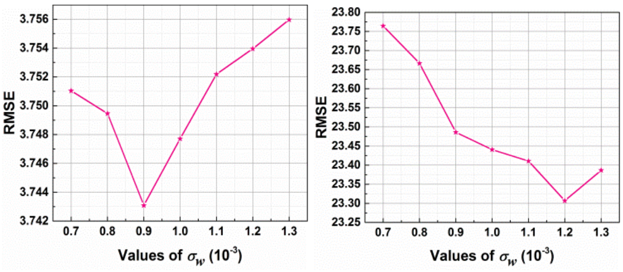
(a) RMSE on D11 (b) RMSE on D21

Figure 5. The KLFA's prediction accuracy as C varies.



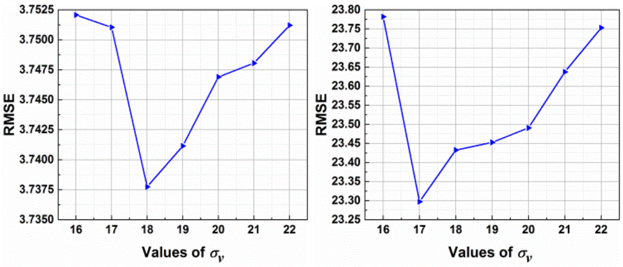
(a) RMSE on D11 (b) RMSE on D21

Figure 6. The KLFA's prediction accuracy as f varies.



(a) RMSE on D11 (b) RMSE on D21

Figure 7. The KLFA's prediction accuracy as σ_w varies.



(a) RMSE on D11 (b) RMSE on D21

Figure 8. The KLFA's prediction accuracy as σ_v varies.

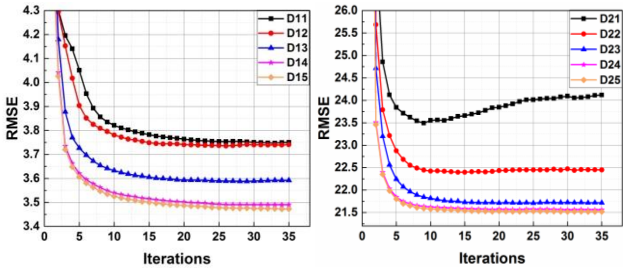
b) Second, we fix C, σ_w and σ_v to investigate KLFA's performance as f varies. As shown in Figure 6(a), on D11, the lowest RMSE is 3.7477 with f=20 and the highest RMSE is 3.7826 with f=120. The largest gap of RMSE is 0.92%. On D21, the lowest and highest RMSE are 23.2610 with f=80 and 23.5456 with f=10, respectively. The gap reaches 1.21%. Note that KLFA's prediction accuracy also relies on the LF dimension. However, in the previous studies [10, 23], the prediction accuracy of LFA-based approaches for missing data usually increases as f increases. Nonetheless, this rule does not hold for KLFA.

c) Finally, we investigate the effects of transition noise and observation noise. As depicted in Figures 7 and 8, KLFA's prediction accuracy is always closely connected with σ_w and σ_v: 1) fix other parameters to investigate KLFA's performance as σ_w varies. For instance, as shown in Figure 7(a), on D11, the lowest RMSE is 3.7431 with $\sigma_w=0.9\times10^{-3}$ and the highest RMSE is 3.7560 with $\sigma_w=1.3\times10^{-3}$. The gap is 0.34%; 2) fix other parameters to investigate KLFA's performance as σ_v varies. For instance, on D21, as shown in Figure 8(b), the lowest RMSE is 23.2972 with σ_v=17, 0.48% higher than 23.7818 with σ_v=16.

Based on these results, it is necessary to tune these parameters carefully to ensure high prediction accuracy. Note that the tuning process is a tedious task. This problem can be addressed via offline tuning setting, or adopting empirical values. Without loss of generality, we simply fix $C=1.0\times10^{-3}$, f=20, $\sigma_w=1.0\times10^{-3}$, and σ_v=20 in the following experiments.

4.3 Convergence Process

In this part, we aim to validate the convergence of KLFA. The training process with different testing cases are shown in Figure 9. We have some meaningful findings from these results:



(a) RMSE on D1 (b) RMSE on D2

Figure 9. Training process with different test cases.

a) Figure 9 depicts that the prediction accuracy is connecting with data ratios. When the ratio is small (e.g., ratio is 5% or 10%), the RMSE and MAE are higher. For instance, as shown in Figure 9(a), on D1, the lowest RMSE and MAE are 3.7477 and 1.8675, 3.7354 and 1.7545, 3.5881 and 1.7545, 3.4902 and 1.6583, 3.4724 and 1.6504 when the ratio of training data are 5%, 10%, 20%, 40% and 50%, respectively. The RMSE decreases as the ratio increases. Similar results can also be found on D2.

b) It is different on D1 and D2 on regarding to the prediction accuracy's change rate as data ratio varies. As shown in Figure 9, when the ratio of training data is increased from 5% to 10%, the improvement of prediction accuracy is insignificant on D1, but significant on D2. For instance, the lowest RMSE is 3.7477 with the testing case D11, only 0.33% higher than 3.7354 with the testing case D12. However, on D2, the lowest RMSE is 23.4906 with the testing case D21, 4.66% higher than 22.3959 with the testing case D22. The gap is quite obvious. When the ratio of training data is further increased from 20% to 40%, the lowest RMSE are 3.5881 and 3.4902 on D1. The gap reaches 2.7%. However, on D2, the RMSE is 21.7122 and 21.5637, respectively. The gap is only 0.68%.

4.4 Comparison with State-of-the-art Models

In this part of experiments, we compare a KLFA-based QoS-estimator with the widely used state-of-the-art temporal-aware QoS-estimators in terms of prediction accuracy. Moreover, we add a widely used static LFA-based QoS-estimator [23] as the baseline, which can be used to illustrate the importance of considering the temporal patterns hidden in QoS data. The details of compared models are listed below:

M1: TimeSVD++, it is a dynamic collaborative recommendation model [17], which models the temporal effects as additional LFs.

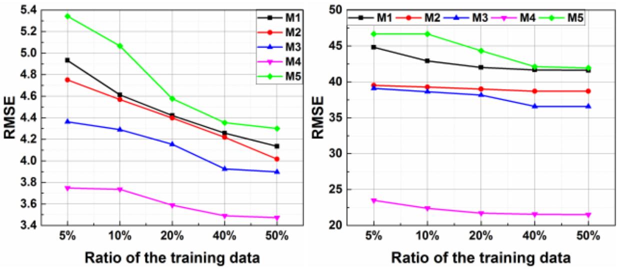
M2: TF, it is a tensor factorization-based model [25], which applies tensor factorization on an user-service-time tensor to extract user-based, service-based and time-based factors directly.

M3: WSPred, it is a tensor factorization-based prediction model with average QoS value constraints [16].

M4: KLFA, it is proposed in this paper, which incorporates Kalman filter to model the temporal patterns.

M5 (baseline): Static LFA, This method considers the user-service-time tensor as a set of user-service matrix slices in terms of time. For each slice, the prediction method proposed by Koren and Bell [24] is employed.

In terms of the hyperparameters in each model, to ensure a fair comparison, we adopt the following settings: 1) For all compared models, we compare their prediction accuracy as the ratio of training data increases from 5% to 50%; 2) M1-M3 and M5 depend on learning rate and regularization coefficient. Without lost of generality, we search their optimal values by performing cross-validation on the observed QoS data under the same latent factor space *f*=20. The results are shown in Figure 10 and Table 4. From them, we have the following findings:



(a) RMSE on D1 (b) RMSE on D2

Figure 10. Prediction accuracy comparison.

a) M4 proposed in this paper outperforms its peers on prediction accuracy with all the testing cases. For instance, as shown in Figure 10(a) and Table 4, on D11, the lowest RMSE achieved by M4 is 3.7477, about 24.01% lower than 4.9322 by M1, 21.11% lower than 4.7508 by M2, 14.09% lower than 4.3626 by M3, and 29.84% lower than 5.3423 by M5. On D21, the lowest RMSE achieved by M4 is 23.4906, about 47.57% lower than 44.8051 by M1, 40.55% lower than 39.5133 by M2, 39.92% lower than 39.0972 by M3, and 49.67% lower than 46.6735 by M5. The improvement of prediction accuracy is significant.

b) M1-M5's RMSE decrease as the ratio of training data increases, that means the RMSE and MAE of dense tensors are lower than those of sparse ones. For instance, on D1, M4 achieves the lowest RMSE at 3.4724 on D15, 7.34% lower than 3.7477 on D11, 7.04% lower than 3.7354 on D12, 3.22% lower than 3.5881 on D13, and 0.51% lower than 3.4902 on D14. These results indicate that a denser tensor provides more information for predicting the missing values.

c) Note that all the compared models except M5 take into account the temporal dynamics patterns hidden in the QoS data. As shown in Table 4, M5 is outperformed by its peers since it does not consider the temporal dynamic patterns hidden in QoS data. For M1-M4, M1 adopts a heuristic function to model the temporal patterns hidden in dynamic data, which may result in inaccurate representation of these temporal patterns; M2 and M3 perform better than M1 since they extract temporal patterns from dynamic QoS data directly through tensor factorization. Moreover, M3 incorporates the average QoS value constraints into M2 to improve the performance on prediction accuracy; However, M2 and M3 suffer accuracy loss because they are tensor factorization approaches, which are in nature a pure optimization-based approach without specific design for describing the temporal patterns. Owing to the nature of Kalman-filter, M4 is able to naturally model such temporal dynamics with state-transition functions, thereby completely describe the temporal patterns hidden in target data.

Table 4. RMSE of compared models.

Dataset	No.	M1	M2	M3	M4	M5
D1	D11	4.9322	4.7508	4.3626	**3.7477**	5.3423
	D12	4.6126	4.5696	4.2892	**3.7354**	5.0667
	D13	4.4212	4.3977	4.1535	**3.5881**	4.5772
	D14	4.2571	4.2191	3.9246	**3.4902**	4.3541
	D15	4.1347	4.0169	3.8971	**3.4724**	4.2996
D2	D21	44.8051	39.5133	39.0972	**23.4906**	46.6735
	D22	42.9100	39.2792	38.6271	**22.3959**	46.6656
	D23	42.0124	38.9987	38.1601	**21.7122**	44.3223
	D24	41.6670	38.6981	36.5811	**21.5637**	42.1177
	D25	41.6047	38.6972	36.5731	**21.5063**	41.9407

5 DISCUSSION

For implementing KLFA, we assume the service LFs evolve very slowly and can be considered constant over the time frame that user LFs are collected by a Kalman filter. Actually it can be named User-based KLFA (U-KLFA). In addition, we try to obtain the filter-service LFs by Kalman filter for modeling the temporal dynamic patterns, which can be named Service-based KLFA (S-KLFA). The details are recorded in the Supplementary File[3].

6 CONCLUSIONS

This paper proposes a KLFA-based QoS-estimator, which considers the temporal dynamics of QoS attributes. It estimates the current LFs based on the previous ones through Kalman filter. The Extensive experiments results based on large-scale real-world Web service QoS datasets show when compared with state-of-the-art temporal-aware QoS-estimators, KLFA is able to achieve significantly higher prediction accuracy for missing QoS data.

However, there are several directions of future work for improving KLFA: a) according to (15), the main drawback of KLFA is the high computational complexity due to matrix inversions. Therefore, parallel computing [26] can be considered to improve the computational efficiency; b) Section 4.2 shows that the tuning process of hyperparameters is a tedious task. Naturally, the ideal way is to implement the parameter's self-adaptive without

[3] https://pan.baidu.com/s/1JyllVuVV9L1ZKGMaDaqXiA

introducing any sizable computational overhead. According to prior research [27], an evolutionary-computation-based algorithm like particle swarm optimization can be useful in implementing efficient adaptations of parameters in our scene. It will be interesting to develop KLFA extensions under such circumstances.

ACKNOWLEDGEMENTS

This work is supported in part by the National Natural Science Foundation of China under grants 61772493, 61902370, 61602434 and 61702475, in part by the Natural Science Foundation of Chongqing (China) under grants cstc2019jcyj-msxmX0578 and cstc2019jcyjjqX0013, and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences.

REFERENCES

[1] M. Bahrami and M. Singhal, ‘The role of cloud computing architecture in big data’, *Springer International Publishing*, (2015).

[2] H. Artail, M. Saghir, and M. Sharafeddin, ‘Speedy cloud: Cloud computing with support for hardware acceleration services’, *IEEE Transactions on Cloud Computing*, 7(3), 850-865, (2019).

[3] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, ‘Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services’, *IEEE Transactions on Services Computing*, 3(3), 223-235, (2010).

[4] P. Wang, A. K. Kalia, and M. P. Singh, ‘A collaborative approach to predicting service price for QoS-aware service selection’, In *Proceedings of the IEEE International Conference on Web Services*, 33-40, (2015).

[5] K. Liang, A. K. Qin, K. Tang, and K. C. Tan, ‘QoS-Aware Web Service Selection with Internal Complementarity’, *IEEE Transactions on Services Computing*, 12(2), 276-289, (2019).

[6] J. E. Haddad, M. Manouvrier, and M. Rukoz, ‘TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition’, *IEEE Transactions on Services Computing*, 3(1), 73-85, (2010).

[7] X. Luo, M. Zhou, Z. Wang, Y. Xia, and Q. Zhu, ‘An effective scheme for QoS estimation via alternating direction method-based matrix factorization’, *IEEE Transactions on Services Computing*, 12(4), 503-518, (2019).

[8] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, ‘Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models’, *IEEE Transactions on neural networks and learning systems*, 27(3), 524-537, (2016).

[9] D. Yu, Y. Liu, Y. S. Xu, and Y. Y. Yin, ‘Personalized QoS Prediction for Web Services Using Latent Factor Models’, In *Proceedings of the IEEE International Conference on Services Computing*, 107-114, (2014).

[10] X. Luo, M. Zhou, S. Li, Y. Xia, Z. You, Q. Zhu, and H. Leung, ‘Incorporation of Efficient Second-Order Solvers Into Latent Factor Models for Accurate Prediction of Missing QoS Data’, *IEEE Transactions on Cybernetics*, 48(4), 1216-1228, (2017).

[11] Z. Zheng, H. Ma, M. R. Lyu, and I. King, ‘Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization’, *IEEE Transactions on Services Computing*, 6(3), 289-299, (2013).

[12] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, ‘A Nonnegative Latent Factor Model for Large-Scale Sparse Matrices in Recommender Systems via Alternating Direction Method’, *IEEE Transactions on Neural Networks and Learning Systems*, 27(3), 579-592, (2016).

[13] Q. Wang, M. Chen, and M. Shang, ‘A momentum-incorporated latent factorization of tensors model for temporal-aware QoS missing data prediction’, *Neurocomputing*, 367, 299-307, (2019).

[14] R. Xiong, J. Wang, and Z. Li, ‘Personalized LSTM Based Matrix Factorization for Online QoS Prediction’, In *Proceedings of the IEEE International Conference on Services Computing*, 34-41, (2018).

[15] W. Lo, J. Yin, and S. Deng S, ‘Collaborative Web Service QoS Prediction with Location-Based Regularization. In *Proceedings of 19th IEEE International Conference on Web Services*, 464-471, (2012).

[16] Y. Zhang, Z. Zheng, and R. L. Michael, ‘WSPred: A Time-Aware Personalized QoS Prediction Framework for Web Services’, In *Proceedings of the 22nd IEEE International Symposium on Software Reliability Engineering*, 210-219, (2011).

[17] Y. Koren, ‘Collaborative filtering with temporal dynamics’, In *Proceedings of the 15th SIGKDD International Conference on Knowledge discovery and data mining*, 447-456, (2009).

[18] N. Sahoo, P. V. Singh, and T. Mukhopadhyay, ‘A hidden Markov model for collaborative filtering’, *Mis Quarterly*, 36(4), 1329-1356, (2012).

[19] W. Zhang, H. Sun, X. Liu, and X. Guo, ‘Temporal QoS-aware Web Service recommendation via Non-negative Tensor Factorization’, In *Proceedings of the 23rd International Conference on World wide web*, 585-596, (2014).

[20] G. Welch and G. Bishop, ‘An introduction to the Kalman filter’, *Technical Report*, (1995).

[21] Z. Zheng, Y. Zhang, and M. R. Lyu, ‘Distributed QoS evaluation for real-world Web services’, In *Proceedings of the IEEE International Conference on Web Services*, 83-90, (2010).

[22] Z. Zheng, Y. Zhang, and M. R. Lyu, ‘Exploring Latent Features for Memory-Based QoS Prediction in Cloud Computing’, In *Proceedings of the 30th IEEE International Symposium on Reliable Distributed Systems*, 1-10, (2011).

[23] Y. Hu, Y. Koren, and C. Volinsky, ‘Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, 263–272, (2009).

[24] Y. Koren, R. Bell, and C. Volinsky, ‘Matrix-factorization techniques for recommender systems’, *IEEE Computer*, 42(8), 30-37, (2009).

[25] S. Rendle and L. S. Thieme, ‘Pairwise interaction tensor factorization for personalized tag recommendation’, In *Proceedings of the 3rd International Conference on Web search and data mining*, 81-90, (2010).

[26] O. Beaumont, B. A. Becker, A. M. DeFlumere, and A. L. Lastovetsky, ‘Recent Advances in Matrix Partitioning for Parallel Computing on Heterogeneous Platforms’, *IEEE Transactions on Parallel and Distributed Systems*, 30(1), 218-229, (2019).

[27] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Paster, ‘Particle swarm optimization for hyper-parameter selection in deep neural networks’, *in Proceedings of the ACM International Conference on Genetic and Evolutionary Computation*, 481-488, (2017).

[28] X. Luo, H. Wu, M. Zhou, and H. Yuan, ‘Temporal Pattern-aware QoS Prediction via Biased Non-negative Latent Factorization of Tensors’, *IEEE Transactions on Cybernetics*, 10.1109/TCYB.2019.2903736, (2019).

[29] S. Benferhat, D. Dubios, and H. Prade, ‘Kalman-like Filtering in a Possibilistic Setting’, In *Proceedings of the 14th European Conference on Artificial Intelligence*, 8-12, (2000).

[30] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl, ‘Evaluating collaborative filtering recommender systems’, *ACM Trans. on Information Systems*, 22(1), 5-53, (2004).