Hindawi
Journal of Electrical and Computer Engineering
Volume 2017, Article ID 9547869, 10 pages
https://doi.org/10.1155/2017/9547869

Research Article

Dynamically Predicting the Quality of Service: Batch, Online, and Hybrid Algorithms

Ya Chen and Zhong-an Jiang

University of Science and Technology, Beijing 100080, China

Correspondence should be addressed to Zhong-an Jiang; jza1963@263.net

Received 10 August 2016; Accepted 14 February 2017; Published 6 March 2017

Academic Editor: Jar Ferr Yang

This paper studies the problem of dynamically modeling the quality of web service. The philosophy of designing practical web service recommender systems is delivered in this paper. A general system architecture for such systems continuously collects the user-service invocation records and includes both an online training module and an offline training module for quality prediction. In addition, we introduce matrix factorization-based online and offline training algorithms based on the gradient descent algorithms and demonstrate the fitness of this online/offline algorithm framework to the proposed architecture. The superiority of the proposed model is confirmed by empirical studies on a real-life quality of web service data set and comparisons with existing web service recommendation algorithms.

1. Introduction

The quality of service or QoS has been exploited by many application domains as an importance metric for users to evaluate the quality of provided services. Especially when many providers offer similar services at the same time, this metric can be taken to identify reliable ones. The quality of service can be collected after user using a service and a user-service QoS matrix can be generated with these values. Once we have such a matrix, existing recommendation approaches [1–4] can be exploited to generate predictions for the missing values of the user-service QoS matrix. Because the QoS value is always related to the temporal/spatial information [5–7], other contextual information [8] has been used for improving the accuracy of QoS prediction. As the user-service QoS matrix is dynamic and increases with the usage of service, the model learned before may not be suitable for the current state. This fact requires us to rebuild a model after an increasing of data for a better model. Usually, it takes a long time when the user-service QoS matrix is huge. A dynamic model is needed for handling dynamics of available data.

As a concrete example, we will focus on one type of QoS prediction problem, the quality of web service prediction, to demonstrate the dynamic model of QoS prediction. Web service and APIs are very useful resources when developing Mashup applications. For example, ProgrammableWeb (ProgrammableWeb can be accessed at https://www.programmableweb.com/) provides over 11,000 collected web services and 12,000 open APIs [9]. The increasing number of services and APIs makes the effort of finding reliable services more difficult.

Since it is impossible for service users to make selection decisions without prior knowledge about service candidates, it is vital for researchers to develop quality of service (QoS) prediction approaches [10, 11] or service recommender systems [12, 13] to assist users or programs when dynamically choosing their services. In general, the problem of predicting the quality of web services may be abstracted in terms of a collection of services, a collection of users (programs), and each user's experiences (quality of services) on a subset of services. The objective for developing a QoS predicting system is to decide what level of quality a service can bring to a particular user or to a typical user.

Existing recommendation approaches have been applied in the web service discovering domain, for example, user-based collaborative filtering (CF) [14], item-base CF [15], and hybrid CF [16]. These web service and API prediction systems assume that all user-service invocation records are collected and the trained model will not change with the accumulation

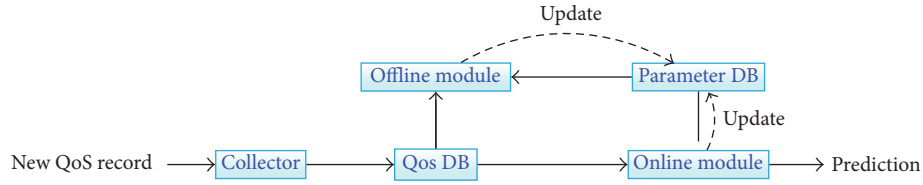

Figure 1: QoS prediction system architecture.

of new QoS value arriving to the system. In practice, this assumption is not always true and the QoS pattern may vary. A robust web service QoS prediction system should handle the dynamics. The training set for a batch algorithm or an offline algorithm must be available *a priori* and an online algorithm is required to update the learned model instead of retraining the whole model from scratch.

As mentioned before, context information is one of the most important information for QoS prediction, in web service or API recommendation scenarios; context-aware recommendation [17–20] is used to characterize the correlation between users' dynamic preferences and their contexts. Promising performance improvement has been achieved by these context-aware systems. However, these models can only handle the static data and the pattern dynamics are not able to be captured by these approaches.

Recently, with the success of matrix factorization and tensor factorization techniques on recommender systems [21–23], they have also been introduced to the web service QoS prediction domain [24]. The basic idea is to use a tensor factorization-based approach to evaluate time-aware personalized QoS values. These studies achieve performance improvements in comparison with traditional models; however, the temporal dynamics of user-service invocation records is not considered. We advocate that a 3-way tensor does not explicitly carry the necessary dynamics of user-side and service-side properties that may impact the web service quality. Therefore, we propose a timeSVD [25] based approach to characterize the dynamics of property change over time.

In this paper, we propose a general architecture and dynamic model which can update the learned model only based on the change of QoS values. Specifically, we design a hybrid model which jointly combines the online and offline algorithms. Such system is capable of monitoring the change of QoS values and adapting itself in real-time to the new arriving data. We conduct experiments on some real-life data and the empirical study confirms the effectiveness and efficiency of the proposed model.

The remainder of this paper is organized as follows. Section 2 gives the architecture of QoS prediction systems monitoring user-service invocation streams and incrementally predicting the quality of services. Section 3 presents a dynamic quality of service model and derives online, offline, and hybrid predictors resulting from a stochastic gradient descent algorithm. Section 4 presents an experimental evaluation of our approach. This paper ends with some discussion and brief conclusions.

2. Architecture

For a system which is capable of characterizing the dynamics of monitored services, one needs to instantaneously update the system parameters based on the newly available data. This fact indicates the most important feature of this system is the computing complexity. With the increasing of available data, it is impossible for a system to learn the model from all the collected historical data. This requires a system structure to simultaneously contain both offline and online training functions. The offline training module trains the model based on all available data. So the time cost of executing this part is relatively high. The offline training model updates the previously trained model parameters. Although training an online model is more efficient than the offline training module, the model accuracy may be lower than the offline model. When we collect enough data or the system is not busy, we can execute the batch algorithm or the offline model to improve the model accuracy of the online model. Note that we only keep one set of model parameters in the system; both online algorithm and offline algorithm will modify the same model. This requires us to design such an algorithm that supports this requirement. Figure 1 illustrates the architecture design of our system. Five components are involved in this architecture. Collector receives the new arrival QoS record and stores the incoming message to QoS DB. Then the online module, usually implemented by a light-weight algorithm, takes the existing model parameters from parameter DB and updates the parameters based on online algorithms. The updated parameters are then transferred to the parameter DB. The parameter DB stores the model parameters of the running systems. Online trainer and offline trainer will all start their process based on the model parameters saved in the parameter DB. If new prediction request comes, the online module will be executed to generate predictions based on the most recent learned model parameters. The offline module, which implements time costly training algorithms, is only invoked when the system is free loaded or has sufficient computation resources. The training of offline module also starts from the most recent model parameters and the updated ones are also saved in the parameter DB.

It can be seen that this architecture design is generic and can be suitable for many other QoS prediction scenarios based on user experiences. This fact shows the applicability of our proposed architecture.

From the architecture design, we can find that a profound algorithm framework is demanded to support the online and offline module together. That means that the online training

algorithm is, on the one hand, capable of performing training "incrementally" based on the existing training results and, on the other hand, does so at a quick rate. In the next section, we introduce an algorithm framework based on stochastic gradient descent which can be transferred into online and also offline way.

3. Time-Dependent Quality of Web Service Prediction Model

In this section, we propose the formulation of the time-dependent quality of web service problem and a unified QoS prediction model.

3.1. Problem Statement. We use $\mathcal{I}$ and $\mathcal{U}$ to denote, respectively, the collection of all observed web services and the set of all users. We denote by $r_{u,i}$ the quality of web service i experienced by user u and by $r_{u,i}$ the estimate of this quantity is denoted by $\widehat{r}_{u,i}$. In order to capture the temporal dynamics of the quality of a web service, time is slotted into intervals, and we use variable t to index the time interval in which the service is invoked. Then the quantities $r_{u,i}$ and $\widehat{r}_{u,i}$ are augmented to the form of $r_{u,i,t}$ and $\widehat{r}_{u,i,t}$ correspondingly. Let $\mathcal{T}$ denote the time index set $\{1, 2, \ldots, T\}$ for some positive integer T. The set of all observed $\{r_{u,i,t}\}$ across all users $\mathcal{U}$, all services $\mathcal{I}$, and all time-slots $\mathcal{T}$ is denoted by $\mathcal{R}$. We note that it is not the case that, for every $(u, i, t) \in \mathcal{U} \times \mathcal{I} \times \mathcal{T}$, service quality $r_{u,i,t}$ is observed; in fact for a quite significant fraction of $\mathcal{U} \times \mathcal{I} \times \mathcal{T}$, service quality is not observed. We denote by $\mathcal{K}$ the set of all (u, i, t) triples for which $r_{u,i,t}$ is not observed.

Let $\mathcal{H} \subset \mathcal{U} \times \mathcal{I} \times \mathcal{T}$ be a subset of (u, i, t) triples for which $r_{u,i,t}$ is not observed. The problem of predicting the quality of web services can then be phrased as for each $(u, i, t) \in \mathcal{H}$ obtaining an estimate $\widehat{r}_{u,i,t}$ of $r_{u,i,t}$ based on $\mathcal{R}$. We note that $\mathcal{H}$ does not necessarily contain all unobserved (u, i, t) triples. The typical scenario is that it contains only the ones of practical interest. For example, if T indexes the present time-slot, $\mathcal{H}$ is the set of all (u, i, T) triples for which $r_{u,i,T}$ is not observed, or, alternatively, $\mathcal{H}$ could also be the set of all $(u, i, T + 1)$ triples. The two cases here correspond to predicting the unobserved service qualities in the current time-slot or predicting the service qualities in next time-slot.

3.2. Matrix Factorization Model and Its Variants. The matrix factorization model and its various modifications can be used to solve the service quality prediction problem of our interest. The key idea of matrix factorization is assuming a latent low-dimensional space $\mathbb{R}^D$ on which, for each user u, a user feature p_u is defined and for each item (i.e., service in the context of this paper) i, an item feature is defined. That is, p_u and q_i both belong to $\mathbb{R}^D$, and the estimated rating $\widehat{r}_{u,i}$ is defined by the inner product of these two vectors: namely,

$$\widehat{r}_{u,i} = q_i^T p_u. \tag{1}$$

We note that, in the original matrix factorization setting, the problem considered does not have temporal dynamic, or, equivalently phrased, time index set $\mathcal{T}$ contains only a single time index, and as a consequence, index t is disregarded.

Representing the collection of q_i's as a $D \times |\mathcal{I}|$ matrix Q and the collection of $\{p_u\}$ as a $D \times |\mathcal{U}|$ matrix P, the estimation problem of interest then reduces to solve the following minimization problem: find (Q, P) that minimizes

$$\sum_{(u,i)\in\mathcal{K}} \left\| r_{u,i} - \widehat{r}_{u,i} \right\|^2 + \lambda_Q \|Q\|^2 + \lambda_P \|P\|^2 \tag{2}$$

for some given positive values of λ_Q and λ_P. The notation $\|\cdot\|$ denotes either vector l-2 norm or matrix Frobenius norm, which should be clear from the context.

An extension of matrix factorization is regularized SVD with bias [26], which formulates the estimated $\widehat{r}_{u,i}$ as

$$\widehat{r}_{u,i} = \mu + b_i + b_u + q_i^T p_u, \tag{3}$$

where μ is the average of all QoS values on $\mathcal{K}$ and b_u and b_i are, respectively, user bias and item bias. Denote by $B_{\mathcal{U}}$ the collection of all b_u's and by $B_{\mathcal{I}}$ the collection of all b_i's. The estimation problem then reduces to the following minimization problem: find $(\mu, B_{\mathcal{U}}, B_{\mathcal{I}}, Q, P)$ that minimizes

$$\begin{aligned} &\sum_{(u,i)\in\mathcal{K}} \left\| r_{u,i} - \widehat{r}_{u,i} \right\|^2 \\ &\quad + \lambda \left(\|Q\|^2 + \|P\|^2 + \left\| B_{\mathcal{U}} \right\|^2 + \left\| B_{\mathcal{I}} \right\|^2 \right). \end{aligned} \tag{4}$$

The optimization problems as stated in (2) and (4) can both be solved using gradient descent or stochastic gradient descent algorithms.

A further extension of the above SVD model is the so-called SVD++ model [25], in which the total number of entries in $\mathcal{R}$ that correspond to each user u is incorporated in defining $\widehat{r}_{u,i}$. This leads to a revised form of the cost function in the optimization problem and further improved performance is obtained. Recently, [25] has introduced a timeSVD++, which builds upon SVD++ and considers incorporating time information in the definition of $\widehat{r}_{u,i}$, that is, letting $\widehat{r}_{u,i}$ vary with time.

We now restate the timeSVD++ model in the context of service quality prediction. It is worth noting that, in this problem, the number of times that a user invokes web services appears irrelevant, unlike in the typical application of SVD++ or timeSVD++, say Netflix movie-rating prediction problem, where the total number of times a user rates movies provides implicit information about their interest in the movie. For this reason, the version of timeSVD++ we present here drops the term relating the service invocation frequency of the users.

Moving time index t from subscript to a variable, the estimated quality is now defined as a function of time t as follows:

$$\widehat{r}_{u,i}(t) = \mu + b_i(t) + b_u(t) + q_i(t)^T p_u(t), \tag{5}$$

where b_u, b_i, p_u, and p_i are all made time-dependent.

For further specification, let

$$\begin{aligned}
\mathrm{dev}_u(t) &= (t - t_u) \cdot |t - t_u|^{\beta}, \\
\mathrm{dev}_i(t) &= (t - t_i) \cdot |t - t_i|^{\beta}, \\
b_u(t) &= (b_u + \alpha_1 \mathrm{dev}_u(t) + b_{u,t})\, c_i(t), \\
b_i(t) &= b_i + \alpha_2 \mathrm{dev}_i(t) + b_{i,t}, \\
p_u(t) &= p_u + \alpha_3 \mathrm{dev}_u(t), \\
q_i(t) &= q_i + \alpha_4 \mathrm{dev}_i(t).
\end{aligned} \tag{6}$$

Here user-centric bias $b_u(t)$ is modeled as a linear function of user-centric deviation $\mathrm{dev}_u(t)$ at time t multiplied with $c_i(t)$, which models the time-varying rating scale service i receives at time t. The service-centric bias $b_i(t)$ is modeled as a linear function of service-centric deviation $\mathrm{dev}_i(t)$ at time t. In the equations for $\mathrm{dev}_u(t)$ and $\mathrm{dev}_i(t)$, t_i is the time-slot in which the quality of service i is for the first time measured, and t_u is the time-slot in which user u invokes a service.

Similar to the previous settings, this model gives rise to a minimization problem with the following objective function:

$$\mathscr{L} = \sum_{(u,i,t)\in\mathscr{K}} \left\| r_{u,i,t} - \widehat{r}_{u,i,t} \right\|^2 \tag{7}$$

$$+\lambda \left(\sum_u b_u^2 + \sum_i b_i^2 + \sum_u \left\| p_u(t) \right\|^2 + \sum_i \left\| q_i(t) \right\|^2 \right). \tag{8}$$

The parameter set to minimize over in this optimization problem is the set of $\Theta = \alpha_1, \alpha_2, \alpha_3, \alpha_4, b_u, b_i, \{b_{u,t}\}_t, \{b_{i,t}\}_t, p_u, q_i, \{c_i(t)\}_t$ and finding the parameter configurations that minimizes the objective function $\mathscr{L}$ defined above will allow us to predict service quality for unobserved (u, i, t) triples. This set of parameters to be optimized is denoted by θ in the following context.

At this point we have not only arrived at a sensible and well-defined notion of quality of web services, we also have translated the problem of QoS prediction to an optimization problem. In the remainder of this paper, we present a prediction algorithm based on stochastic grade descent algorithm [27].

4. Algorithm

Overall we take a stochastic gradient-based approach to minimize the objective function. Denoting $r_{u,i,t} - \widehat{r}_{u,i,t}$ by $err_{u,i,t}$, we have

$$\begin{aligned}
\frac{\partial_L}{\partial_{b_u}} &= \sum_{u,i,t} -err_{u,i,t} c_i(t) + \lambda b_u, \\
\frac{\partial_L}{\partial_{b_i}} &= \sum_{u,i,t} -err_{u,i,t} + \lambda b_i, \\
\frac{\partial_L}{\partial_{p_u}} &= \sum_{u,i,t} -err_{u,i,t} \left(q_i + \alpha_4 \mathrm{dev}_i(t)\right) + \lambda p_u, \\
\frac{\partial_L}{\partial_{q_i}} &= \sum_{u,i,t} -err_{u,i,t} \left(p_u + \alpha_3 \mathrm{dev}_u(t)\right) + \lambda q_i, \\
\frac{\partial_L}{\partial_{b_{u,t}}} &= \sum_{u,i,t} -err_{u,i,t} c_i(t) + \lambda \left(b_{u,t}\right), \\
\frac{\partial_L}{\partial_{b_{i,t}}} &= \sum_{u,i,t} -err_{u,i,t} + \lambda b_{i,t}, \\
\frac{\partial_L}{\partial_{\alpha_1}} &= \sum_{u,i,t} -err_{u,i,t} \mathrm{dev}_u(t)\, c_i(t) + \lambda \alpha_1, \\
\frac{\partial_L}{\partial_{\alpha_2}} &= \sum_{u,i,t} -err_{u,i} \mathrm{dev}_i(t) + \lambda \alpha_2, \\
\frac{\partial_L}{\partial_{\alpha_3}} &= \sum_{u,i,t} -err_{u,i,t} \mathrm{dev}_u(t) \left(q_i + \alpha_4 \mathrm{dev}_i(t)\right) + \lambda \alpha_3, \\
\frac{\partial_L}{\partial_{\alpha_4}} &= \sum_{u,i,t} -err_{u,i,t} \mathrm{dev}_i(t) \left(p_u + \alpha_3 \mathrm{dev}_u(t)\right) + \lambda \alpha_4.
\end{aligned} \tag{9}$$

4.1. Offline/Batch Training Algorithm. Equation (8) allows a stochastic gradient ascent algorithm to optimize the objective function, in which the value of the objective function can be incrementally increased via updating the configuration of parameters. Referring to a generic element in the parameter set Θ by θ, for each $\theta \in \Theta$, we update each θ according to

$$\theta^{t+1} := \theta^t - \eta \frac{\partial_L}{\partial_\theta}, \tag{10}$$

where η is a choice of step size. This update rule is applied iteratively in offline training, until convergence or upon reaching a prescribed number of iterations. As is well known, such gradient ascent algorithm monotonically increases the value of the objective functions over iterations.

We define a convergence condition so that if the gradient of the objective function becomes small, the parameter is then close to the optima. The algorithm is summarized as in Algorithm 1.

4.2. Online Training Algorithm. When the user-service invocation experiences arrive one at a time in sequence, the parameter update function of the previous stochastic gradient ascent algorithm is essentially our online training algorithm. More precisely, for each new invocation record $r_{u,i,t}^{\mathrm{new}}$ that has just arrived, the online algorithm instantaneously updates θ according to (10). Such an update is only performed *once* in the online algorithm for each newly arrived invocation record. The simplicity of this computation, depending only on the current setting of θ, the incoming record is certainly remarkable.

Input: invocation matrix $\mathcal{R}$, regularization parameter λ, learning rate η, number of latent factors d, max iterations itmax
Output: parameter set Θ
Initialize θ with random value
for $it = 1, it <= it\text{max}, it++$ **do**
  for $u = 1, u <= |\mathcal{U}|, u++$ **do**
    for $s = 1, s <= |\mathcal{S}|, s++$ **do**
      for $t = 1, t <= |\mathcal{T}|, t++$ **do**
        update θ according to Eq. (9)
      end for
    end for
  end for
end for
return Θ;

Algorithm 1: Offline algorithm.

Input: The set of new available invocation records $\mathcal{R}^{\text{new}}_{u,i,t}$, regularization parameter λ, learning rate η, number of latent factors d, and the latest Θ^{old}
Output: parameter set Θ^{new}
for $r^{\text{new}}_{u,i,t} \in \mathcal{R}^{\text{new}}_{u,i,t}$ **do**
  update θ according to Eq. (9)
end for
return parameter set Θ^{new}

Algorithm 2: Online algorithm.

4.3. Hybrid Algorithm. As the online approach only takes into account current incoming invocation record, it is necessary to intermittently apply a batch algorithm within the online algorithm to collectively infer over the existing available data. As such, the convergence speed and the prediction precision are both increased. We introduce a hybrid approach that takes the best of Algorithms 1 and 2 and works in this way: at the beginning of each time step t, the online algorithm will process the incoming records and corresponding users and services between time step $(t-1)$ and t. If a predefined number of data has arrived or if other criteria, such as additional computation resources, are met, a batch algorithm will be activated over all of the collected records during time step 0 and time step t. The batch phase can catch more detailed statistics of the overall QoS prediction model and improve the performance of the online prediction.

Note that when we take all available invocation records at a specific time t, denoted by $\mathcal{R}^t$ as the input of Algorithm 1, then this leads to a batch update rule which iteratively updates Θ until a predefined condition is achieved.

We articulate this algorithm as in Algorithm 3, where RBAC stands for run batch algorithm criteria.

It can be seen that this hybrid algorithm simulates the model updating process of our proposed architecture in Section 3.

5. Experimental Evaluation

In this section, we present the experimental evaluation of the time-dependent quality of web services prediction models.

5.1. Data Set. The dataset, real-world QoS evaluation results (response time and throughput), used for the experimentation is downloaded from http://www.wsdream.net/. This dataset includes 142 users on 4532 web services at 64 continuous time-slots. In this study, we choose the response time of user-service invocation to evaluate the performance of the algorithms. The provider of this dataset proposed a tensor factorization-based prediction approach in [24]. This method, referred as TF in the following context, is one of the baseline approaches compared in this study. Other baseline algorithms include traditional tensor factorization and traditional matrix factorization (MF) by compressing (averaging) the observed user-service pairs over different time periods.

As in practice, user-service invocation is typically quite sparse. Therefore, to evaluate the prediction performance over various data sparsity levels, we randomly choose 10%, 30%, and 50% of the data as the observed data to make the QoS prediction.

We examine the prediction precision of our proposed batch algorithm, online algorithm, and hybrid algorithm in our experiments. To validate the performance of our

Input: the set of new available invocation records $\mathscr{R}^{\text{new}}$, all available invocation records $\mathscr{R}^t$
regularization parameter λ, learning rate η, number of latent factors d, max iterations itmax, and the latest Θ^{old}
Output: parameter set Θ
while R^{new}! = NULL **do**
Update parameters by Algorithm 1, given $\mathscr{R}^{\text{new}}$, λ, η, d, and Θ^{old} as the input parameters;
if RBAC == true **then**
Update parameters by Algorithm 1, given $\mathscr{R}^t$, λ, η, d, itmax, and Θ^{old} as the input parameters;
end if
end while

Algorithm 3: Hybrid algorithm.

proposed models, we also consider TF [24, 28] and MF [26] for comparison.

As the traditional matrix factorization only predicts the quality of service for a specific time-slot, as in the scenario of time-dependent QoS prediction, the quality of a web service i can be characterized by the mean of $r_{u,i,\tau}$, where $r_{u,i,\tau}$ is the set of predicted QoS value for the observed time-slots. This measure, which we call the *average quality of web services* of service i, is formally defined follows:

$$r_{u,i} = \frac{\sum_{t\in\tau} r_{u,i,t}}{|\tau|}, \tag{11}$$

where $|\cdot|$ denotes the cardinality of a set. It is clear that this value is not able to characterize the dependencies and network performances over different time intervals.

5.2. Performance Evaluation. In this section, we present the experimental evaluation of the online incremental QoS prediction models in terms of RMSE, the prediction precision.

5.2.1. Evaluation Metric. As discussed in Section 3, the objective of our approach is to find a set of parameters to minimize the loss function of observing the web service invocation performance. Therefore, the loss function, RMSE, can naturally be used as the evaluation metric to compare the overall performance for the resulting model.

The definition of RMSE for the specific time-slot t is given by the following equation:

$$\text{RMSE} = \sqrt{\frac{\sum_{r_{i,j,t}\in\mathscr{T}} \sum_{u\in\mathscr{U}'} \left(\widehat{r}_{i,j,t} - r_{i,j,t}\right)^2}{|T|}}, \tag{12}$$

where $r_{i,j,t}$ is the observed QoS of service i invoked by user u at time-slot t, $\widehat{r}_{i,j,t}$ is the predicted corresponding QoS value, and T is the testing set.

5.3. Experimental Results. For comparison purpose, we also evaluate the performance of our proposed algorithm (TSVD) (a pure online version, a pure offline versio,n and the hybrid version the algorithm), MF, and TF on the collected data sets. We performed the following steps to demonstrate the performance of online algorithm and hybrid algorithm. At the beginning, the experiments have no knowledge of user-service invocation records. At this point, the online algorithm operates every time an invocation record (or several records) arrives. We randomly choose one user-service invocation record and the performance of the algorithms is evaluated after each record arrives. After the incoming data arrives the parameters of online algorithm are updated and the performance on the testing set is evaluated. We also investigate the performance of the offline algorithm and other compared approaches with all the invocation records available. The results are constant when compared with the dynamic results from online algorithm and hybrid algorithm.

5.3.1. Batch Algorithm Performance. To show the impact of iteration rounds on the performance of our system, we evaluate the RMSE of the training set and the testing set with the increasing of the number of iterations. We first randomly choose 10% of the 64th time-slot as the testing set. To evaluate how training data sparseness affects algorithm performance, we then randomly choose 10%, 30%, and 50% of each time-slot of the original user-service invocation matrix as the training data, respectively. Figure 2 shows this impact at 10%, 30%, and 50%. We observe that our algorithm converges at around 100 to 300 iterations. In addition, the RMSE of the training set decreases when increasing the percentage of the training data. The RMSE of the testing set almost remains at the same level for all levels of sparseness. These two facts confirm the stability of our approach.

We investigate the effect of time-slots of data on the performance of different methods and the results are shown in Figure 3. When the sparsity of the user-service invocation matrix is great, 10% in this experiment, the performance of all three approaches increases when the number of observed time-slots increases. Our offline TSVD achieves the best performance when we choose 40 time-slot previous invocation records as the training set ($x = 40$). However, when the sparsity of the user-service invocation matrix decreases, 30% and 50%, the RMSE of the testing set becomes worse as the observed number of time-slots increases. For MF and TF, the RMSE increases with an increase of the observed time-slots of data. Also, when increasing the sparsity of the observed user-service invocation matrix, the performance of all three compared algorithms is increased in terms of RMSE.

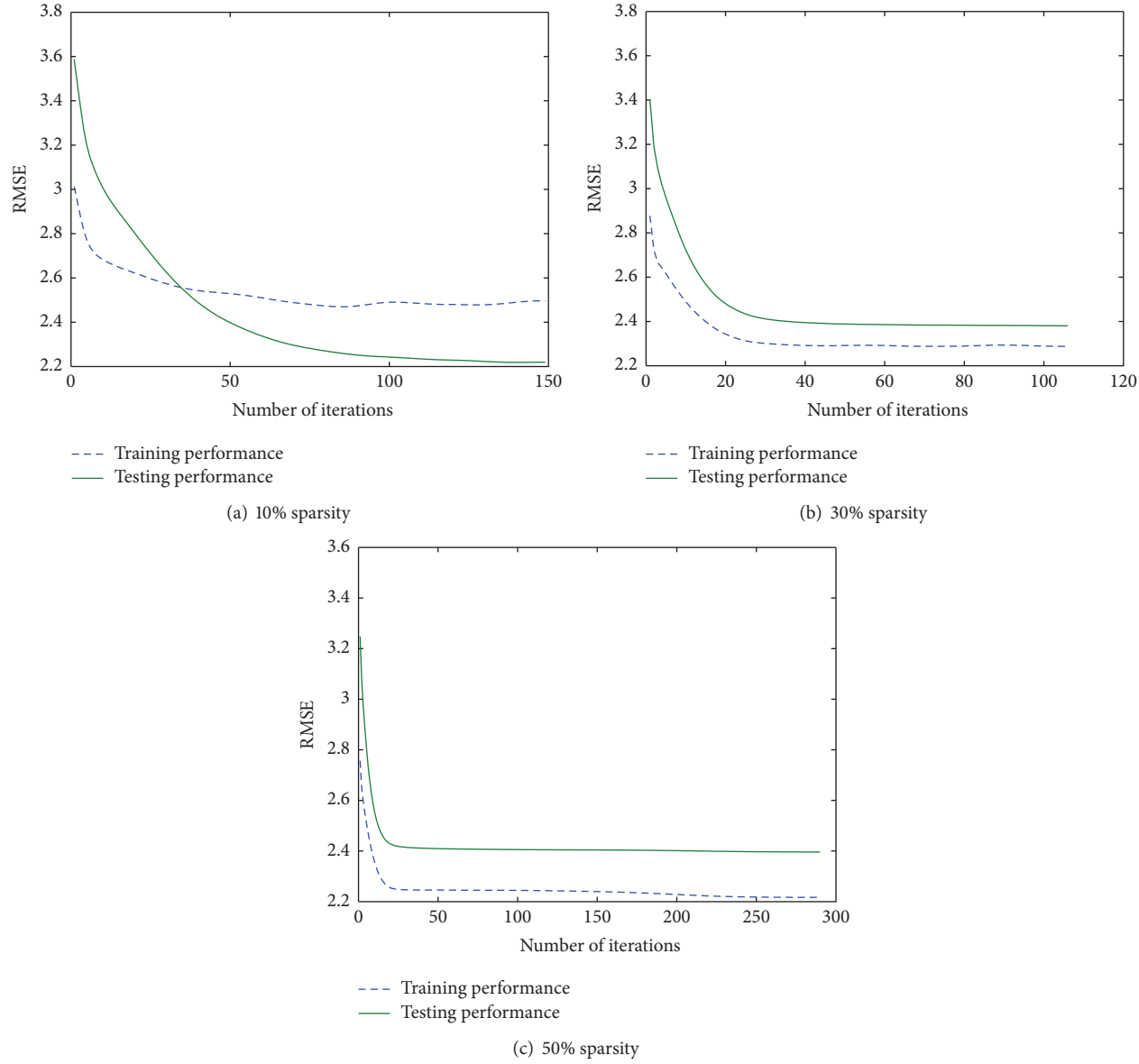

Figure 2: Performance dynamics of the offline TSVD over iteration rounds.

5.3.2. Online and Hybrid Algorithm Performance. We analyze the performance obtained from the offline algorithm, the online algorithm, and the hybrid algorithm to verify the superiority of our approach. To simulate the user-service invocation record arriving scenario, we randomly choose 10% of each time-slot records as the observed invocation record stream and take another 5% of each time-slot as the testing set. This experimental setup means that, for each 15 minutes' time-slot, there are about 20,000 invocation records arriving to the system. We evaluate the online and hybrid algorithms after the arrival of each 100 invocation records by calculating the RMSE of the testing set of the corresponding time-slot.

For the online algorithm, we merely keep updating the parameters with the incremental of records, for example, Algorithm 2. In the hybrid approach, the parameter update follows the same manner as the online algorithm except at the end of each time-slot where a batch algorithm is executed on all the newly arrived data at the current time-slot.

Figure 4 demonstrates the performance of the online algorithm and the hybrid algorithm as user-service invocation records incrementally arrive. The RMSE performance of the testing set continues to increase as new data arrives. After accumulating enough invocation records, the online algorithm and the hybrid algorithm reach the similar level as the batch algorithm.

Moreover, as shown in Figure 4, the performance of the hybrid algorithm is always better than the online algorithm as the sequence of invocation records become available for the response time. At the end of the invocation record stream, the hybrid algorithm can perform the same level of performance as the offline algorithm.

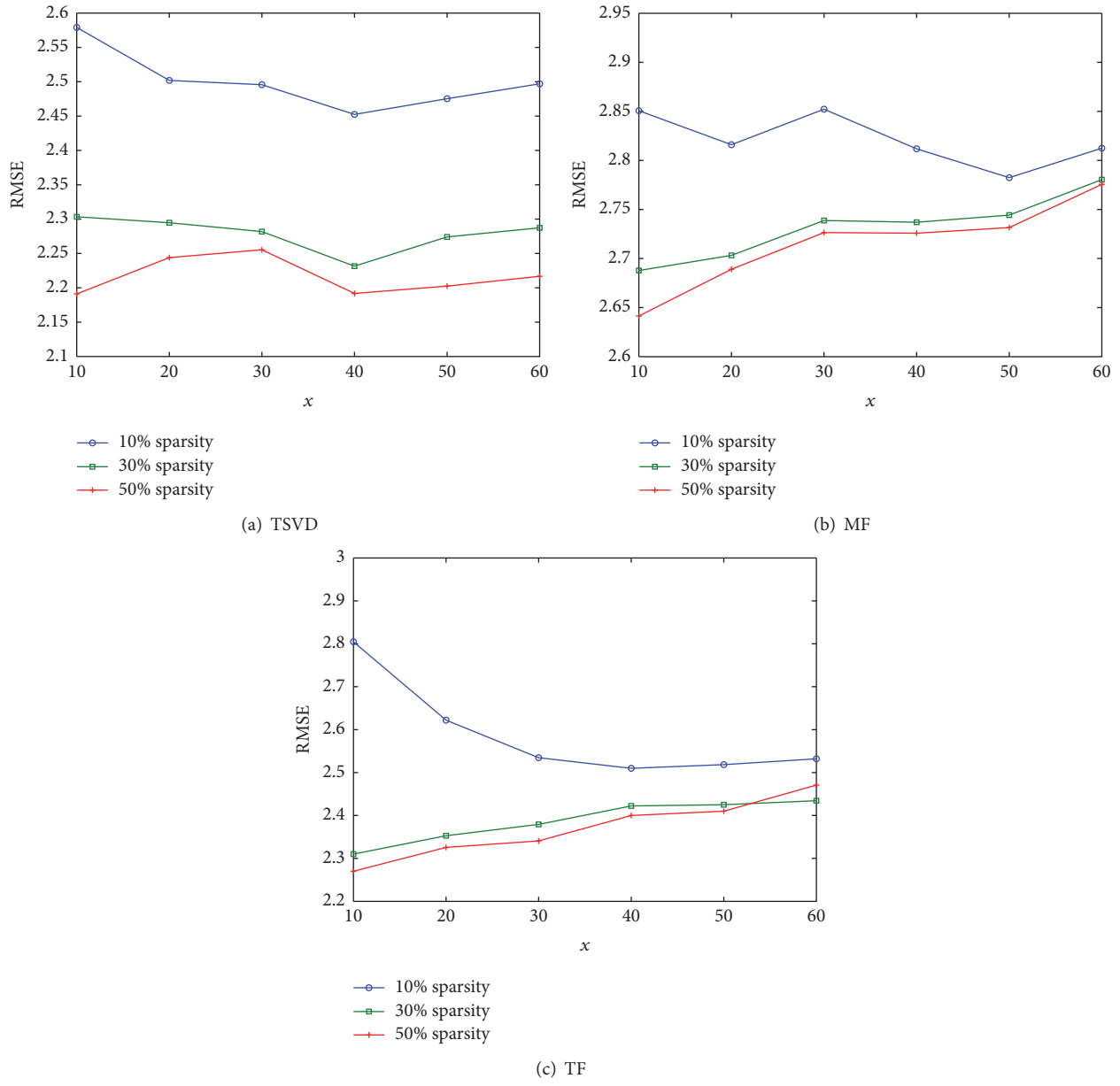

Figure 3: Performance comparison of the offline TSVD, MF, and TF.

This fact confirms that introducing a batch phase in the online algorithm improves the convergence rate of the online algorithm.

The user-service invocation records incrementally become available as shown in Figure 4. We see that the performance of our hybrid approach is close to the batch algorithm after observing the invocation records of the first few time-slots. This confirms the advantages of our quality of web service modeling approach.

Figure 4 also shows that the prediction performances of the hybrid algorithm are better than the online algorithm in terms of RMSE and reach the performance level of the offline algorithm much faster than the online algorithm. This is because the batch phase of the hybrid algorithm increases the prediction performance of the online time SVD. We can also observe that, as the observed data increases, the performance of both the online and the hybrid algorithms is improved.

Based on the above experiments, we have observed an improvement for the quality of web service prediction problem with both online and offline approaches. We conclude that our proposed model can effectively determine the quality of web services. In practice, the hybrid algorithm can begin by collecting a small amount of service invocation records and the parameters of the model can be initialized by the batch algorithm, so that the performance will be further improved.

6. Conclusion

In this paper, we propose a hybrid system designing framework for dynamic service quality prediction, which takes

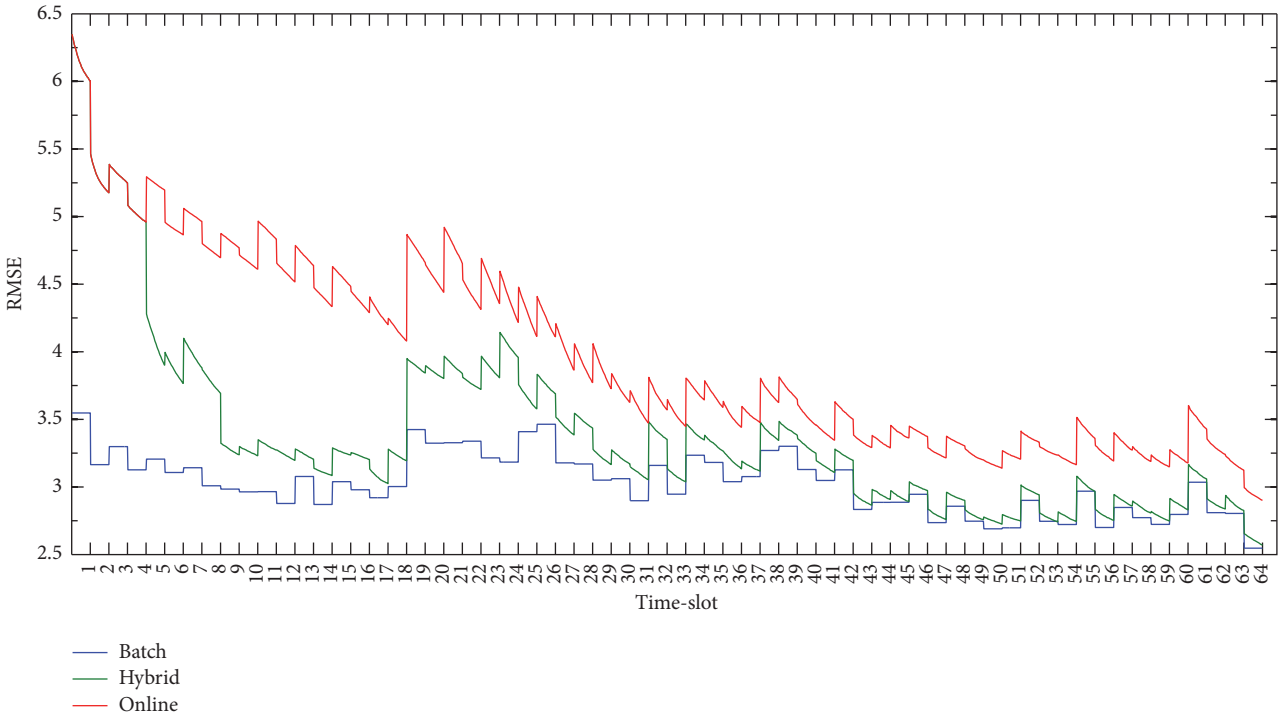

Figure 4: Performance comparison.

the best of both online and offline algorithms, gives a good tradeoff for the effectiveness of the offline algorithm, and is applicable to most real-life scenarios. We have also developed a quality of service predicting system that fits the proposed architecture and have introduced a factorization approach to formulate the quality of services which is a more generative model for quality assessment. An online algorithm and an offline algorithm have been developed under this generative model. Furthermore, we have designed a practical hybrid algorithm to simulate the process of the proposed architecture, which begins working with little or no knowledge of web services and user experiences. The model is refined as it goes along. Empirical studies have been conducted on a real web service invocation data set and experimental results show that the quality of services can be precisely predicted.

Although this paper provides a unified modeling approach for the quality of service prediction, the general factorization methodology presented is applicable to the algorithmic engines of other predictors from user experiences. We also note that the factorization modeling approaches allow the use of more advanced features, possibly encoded by tensor factorization, and other well-principled algorithms to develop both online and offline trainers for the proposed generic predictor architecture.

Competing Interests

The authors declare that they have no competing interests.

References

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Itembased collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, pp. 285–295, ACM, Hong Kong, May 2001.

[2] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[3] J. Wang, A. P. De Vries, and M. J. T. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 501–508, ACM, Seattle, Wash, USA, August 2006.

[4] M. Balabanović and Y. Shoham, "Fab: content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[5] J. Liu, M. Tang, Z. Zheng, X. Liu, and S. Lyu, "Location-aware and personalized collaborative filtering for web service recommendation," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 686–699, 2016.

[6] Z. Li, J. Cao, and Q. Gu, "Temporal-aware QoS-based service recommendation using tensor decomposition," *International Journal of Web Services Research*, vol. 12, no. 1, pp. 62–74, 2015.

[7] S. Meng, Z. Zhou, T. Huang et al., "A Temporal-Aware Hybrid Collaborative Recommendation Method for Cloud Service," in *Proceedings of the IEEE International Conference on Web Services (ICWS '16)*, pp. 252–259, San Francisco, Calif, USA, June 2016.

[8] F. Sardis, G. Mapp, J. Loo, M. Aiash, and A. Vinel, "Investigating a mobility-aware qos model for multimedia streaming rate adaptation," *Journal of Electrical and Computer Engineering*, vol. 2015, Article ID 548638, 7 pages, 2015.

[9] P. C. Evans and R. C. Basole, "Revealing the API ecosystem and enterprise strategy via visual analytics," *Communications of the ACM*, vol. 59, no. 2, pp. 26–28, 2016.

[10] L. Li, M. Rong, and G. Zhang, "A web service QoS prediction approach based on multi-dimension QoS," in *Proceedings of the 6th International Conference on Computer Science & Education (ICCSE '11)*, pp. 1319–1322, IEEE, Singapore, August 2011.

[11] L. Chen, Y. Feng, J. Wu, and Z. Zheng, "An enhanced QoS prediction approach for service selection," in *Proceedings of the IEEE International Conference on Services Computing (SCC '11)*, pp. 727–728, July 2011.

[12] Y. Jiang, J. Liu, M. Tang, and X. Liu, "An effective Web service recommendation method based on personalized collaborative filtering," in *Proceedings of the IEEE 9th International Conference on Web Services (ICWS '11)*, pp. 211–218, July 2011.

[13] X. Chen, X. Liu, Z. Huang, and H. Sun, "RegionKNN: a scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *Proceedings of the IEEE 8th International Conference on Web Services (ICWS '10)*, pp. 9–16, IEEE, July 2010.

[14] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS prediction forweb services via collaborative filtering," in *Proceedings of the IEEE International Conference on Web Services (ICWS '07)*, pp. 439–446, IEEE, Salt Lake City, Utah, USA, July 2007.

[15] Q. Zhang, C. Ding, and C.-H. Chi, "Collaborative filtering based service ranking using invocation histories," in *Proceedings of the IEEE 9th International Conference on Web Services (ICWS '11)*, pp. 195–202, July 2011.

[16] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "WSRec: a collaborative filtering based web service recommender system," in *Proceedings of the IEEE International Conference on Web Services (ICWS '09)*, pp. 437–444, IEEE, July 2009.

[17] A. Schmidt, M. Beigl, and H.-W. Gellersen, "There is more to context than location," *Computers and Graphics (Pergamon)*, vol. 23, no. 6, pp. 893–901, 1999.

[18] A. Karatzoglou, L. Baltrunas, K. Church, and M. Böhmer, "Climbing the app wall: enabling mobile app discovery through context-aware recommendations," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*, pp. 2527–2530, Maui, Hawaii, USA, November 2012.

[19] Y. Xu, J. Yin, W. Lo, and Z. Wu, "Personalized location-aware QoS prediction for web services using probabilistic matrix factorization," in *Web Information Systems Engineering—WISE 2013*, vol. 8180 of *Lecture Notes in Computer Science*, pp. 229–242, Springer, Berlin, Germany, 2013.

[20] X. Fan, Y. Hu, R. Zhang, W. Chen, and P. Brezillon, "Modeling Temporal effectiveness for context-aware web services recommendation," in *Proceedings of the IEEE International Conference on Web Services (ICWS '15)*, pp. 225–232, IEEE, July 2015.

[21] B. Hidasi and D. Tikk, "Fast als-based tensor factorization for context-aware recommendation from implicit feedback," in *Machine Learning and Knowledge Discovery in Databases*, pp. 67–82, Springer, Berlin, Germany, 2012.

[22] H. Wermser, A. Rettinger, and V. Tresp, "Modeling and learning context-aware recommendation scenarios using tensor decomposition," in *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM '11)*, pp. 137–144, IEEE, Kaohsiung, Taiwan, July 2011.

[23] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver, "TFMAP: optimizing MAP for top-n context-aware recommendation," in *Proceedings of the 35th Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*, pp. 155–164, Portland, Ore, USA, August 2012.

[24] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: a time-aware personalized QoS prediction framework for Web services," in *Proceedings of the 22nd IEEE International Symposium on Software Reliability Engineering (ISSRE '11)*, pp. 210–219, IEEE, December 2011.

[25] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pp. 447–456, Paris, France, June 2009.

[26] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pp. 426–434, Las Vegas, Nev, USA, August 2008.

[27] L. Bottou, "Stochastic learning," in *Advanced Lectures on Machine Learning*, O. Bousquet and U. von Luxburg, Eds., vol. LNAI 3176 of *Lecture Notes in Artificial Intelligence*, pp. 146–168, Springer, Berlin, Germany, 2004.

[28] M. Welling and M. Weber, "Positive tensor factorization," *Pattern Recognition Letters*, vol. 22, no. 12, pp. 1255–1261, 2001.