

International Journal of Grid and Utility Computing

ISSN online: 1741-8488 - ISSN print: 1741-847X

<https://www.inderscience.com/ijguc>

Quality of service prediction model in cloud computing using adaptive dynamic programming parameter tuner

Monika, Om Prakash Sangwan

DOI: [10.1504/IJGUC.2023.10054820](https://doi.org/10.1504/IJGUC.2023.10054820)

Article History:

Received:	25 June 2020
Last revised:	11 October 2020
Accepted:	29 October 2020
Published online:	21 March 2023

Quality of service prediction model in cloud computing using adaptive dynamic programming parameter tuner

Monika* and Om Prakash Sangwan

Department of Computer Science and Engineering,
Guru Jambheshwar University of Science and Technology,
Hisar, Haryana, India
Email: monikard31@hotmail.com
Email: sangwan00863@gmail.com
*Corresponding author

Abstract: With the continuous proliferation of cloud services, the recommendation of optimal cloud service according to user requirements has become a critical issue and makes it highly infeasible for a single user to find a specific application with QoS requirements and thus, depends on other users' collected information about various cloud services. These collected QoS values are highly non-linear, complex, and uncertain. To deal with the given scenario, it is required to develop a recommender system for the prediction of unknown QoS values using some optimisation techniques. Therefore, we have employed a novel backpropagation-based ADP parameter tuning strategy with two basic prediction techniques for developing the self-adaptive intelligent system to provide an automatic parameter tuning capability to these techniques. To evaluate the proposed approach, we have done a simulation of the approach on a real QoS data set and experimental results show better prediction accuracy compared with other traditional approaches.

Keywords: cloud computing; QoS prediction; ADP parameter tuner; fuzzy C-means clustering; matrix factorisation; back propagation neural network.

Reference to this paper should be made as follows: Monika and Sangwan, O.P. (2023) 'Quality of service prediction model in cloud computing using adaptive dynamic programming parameter tuner', *Int. J. Grid and Utility Computing*, Vol. 14, No. 1, pp.1–14.

Biographical notes: Monika is currently pursuing her PhD degree in Computer Science and Engineering from Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India. Her current research interests are within cloud computing topics and include quality models, service selection, computational intelligence and soft computing.

Om Prakash Sangwan received his MTech degree (Hons) and PhD degree in Computer Science and Engineering from the Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India, where he is currently a Professor in the Department of Computer Science and Engineering. He was an Assistant Professor in the Department of Computer Science and Engineering, School of Information and Communication Technology, Gautam Buddha University, Greater Noida, Uttar Pradesh. He is also a CISCO Certified Network Associate (CCNA) and a CISCO Certified Academic Instructor (CCAI). His research interests include software engineering focusing on planning, designing, testing, metrics and application of neural networks and fuzzy logic and neuro-fuzzy. He is a Member of Computer Science Teacher Association (CSTA), New York, USA, International Association of Computer Science and Information Technology (IACIST), USA, a Professional Member of Association of Computing Machinery, IEEE and a Life Member of Computer Society of India, India.

1 Introduction

In the technological era, there exist a lot of cloud services with similar functionality that make it difficult for cloud users to select accurate cloud service to produce high-quality applications. To handle such issues, the non-functional quality values (Monika and Sangwan, 2019) of different cloud services must be taken into account. Non-functional

characteristics of cloud services with similar functionality play a crucial role in the selection of cloud services. However, the values of these non-functional characteristics of each cloud service are not available due to many following reasons. 1) Unpredictable internet environment 2) Sparsity of available historical QoS data because the user invokes some of the cloud services out of numerous at a time. 3) Evaluating all cloud services is time-consuming and expensive. 4) The

difference in QoS values at the user and server-side. 5) Different service users may experience different QoS for the same cloud due to environmental conditions (Zhang et al., 2010).

There is a high range of data sparsity and uncertainty available in collected data (i.e. QoS values) due to problems mentioned earlier in the above paragraph and it leads to inaccurate cloud QoS prediction which results in large deviation in cloud service selection. Consequently, how to help users in cloud service selection according to their requirements has become one of the most critical issues to solve. Analysing the collected historical data for the prediction of unknown QoS values of cloud service can be an effective way to solve this problem. Therefore, QoS prediction models were proposed for analyzing and predicting of QoS values and gained attention from researchers. The task of predicting the missing QoS value can be considered as filling the blank position in the original database (user-item matrix) in such a way that the predicted values must be consistent with the existing QoS values in the matrix. Initially, in QoS prediction research, basic prediction approaches have been proposed like User-based Collaborative Filtering (CF), user and Item-based CF approaches, etc. Although these techniques are useful for QoS prediction of cloud services but are not capable enough to handle the unpredictable network environment conditions and these complications motivated us to work on some novel techniques for prediction of QoS value of cloud services.

To address the limitation of existing QoS prediction approaches, we have proposed two models i) Optimised Matrix Factorisation Prediction Model ii) Optimised Fuzzy C-Means prediction model. Matrix factorisation and Fuzzy C-Means clustering are some basic traditional techniques used with static model parameters for the prediction of missing QoS values. But these techniques with static parameters are not able to handle the significant changes under the unpredictable internet conditions and sparsity of available historical QoS information. Therefore, in our work, we have used a novel backpropagation-based (Adaptive Dynamic Programming (ADP) parameter tuning strategy with these two basic prediction techniques. Backpropagation is used for improving the accuracy of predictions in neural networks and combined with ADP parameter tuner for developing the self-adaptive intelligent system in proposed QoS prediction models which composed these models with automatic parameter tuning capability.

In summary, the contributions of the paper are as follows:

- Identification of the research problem of QoS prediction in cloud computing and proposed two novel approaches i.e. i) Optimised Matrix Factorisation Prediction Model ii) Optimised Fuzzy C-Means prediction model with backpropagation-based ADP parameter tuning strategy. These prediction models learn from historical data from similar users to achieve high QoS value prediction accuracy.

- Performed some experiments based on real data set to calculate the effectiveness of the proposed approaches. We also compared the proposed approach with other traditional QoS prediction approaches and found the proposed approach to be more effective and accurate.

The rest of the paper is organised as follows: Section 2 provides the related work about QoS prediction. Section 3 presents problem definition and motivation for our work. Section 4 presents the detailed description and working of MF and FCM-based proposed models. In Section 5, we have discussed the ADP parameter tuner in detail and how it helps in parameter optimisation of proposed models. Section 6 shows the experiment results based on the proposed approaches. Section 7 concludes the paper with potential future directions.

2 Related work

With the dynamic nature of services provided by different clouds, predicting QoS value of a cloud service accurately becomes one of the main challenges faced by cloud users. To solve this issue of accurate prediction of QoS attributes (Sangwan, 2018) value, many researchers have worked in various directions of quality prediction and experimented various existing and new methods and their combinations.

Collaborative Filtering (CF) is a very common technique used by many researchers. It predicts the value based on the publicly available contextual metadata and its similarity with the active user. There are two types of collaborative filtering i.e. memory-based CF and model-based CF (Liu and Chen, 2019).

Memory-based CF consists of two types: User-Based and Item-based CF. User-based collaborative filtering is an effective way to recommend useful content by looking for users who share the same rating patterns with the active user pattern and combine the rating of like-minded users with the help of some supervised learning algorithms like user-based nearest neighbour algorithm, etc. (Mansur et al., 2017). Item-based collaborative filtering follows the same pattern with the user-based collaborative filtering except that it predicts the item by finding the similarity between items and the current item associated with active user (Resnick et al., 1994). The drawback of QoS values is that they are sensitive to internet conditions and thus instantly vary over time. So to overcome this problem, Li et al. (2018) proposed a time-aware matrix factorisation model using a temporal smoothing method to perform the time-varying QoS prediction. Jin et al. (2019a) proposed a two-phase approach for time-aware quality prediction in which historical time slices and current time slices are used for QoS prediction. Zhou et al. (2019) proposed spatio-temporal context-aware collaborative neural model by considering multiple spatial features of users and services and temporal feature simultaneously.

Model-based collaborative filtering uses different techniques like neural network, clustering, association technique, etc. to predict the rating of unknown items.

Jin et al. (2019b) proposed a neighbourhood-aware deep learning model using the non-linear relationship between users and user neighbours through a multi-layer perceptron. Chaudhuri (2017) also used the SVM model-based collaborative filtering approach in the field of networking. Somu et al. (2020) proposed a QoS prediction technique by combining enhanced cosine similarity measure with service-user-based historical QoS values. Liu and Chen (2019) developed a model in which clustering is performing using both explicit and implicit context information and then a trust-aware CF approach is used to optimise the trust network of the above-clustered users.

To provide QoS information of cloud services to users, Xu et al. (2016) proposed an approach named online learning-based matrix factorisation. Feng and Huang (2018) presented a neighbourhood enhanced matrix factorisation approach in which the entropy method is used to calculate personal weights for different users or services and then neighbourhood and geographical information of users are integrated into matrix factorisation for prediction of missing values. Model-based CF provides more accurate results than other techniques so here, in this paper, we have also considered a model-based CF approach i.e. Matrix Factorisation for QoS prediction.

To solve the problem of data sparsity clustering is also used by many researchers. Liu and Chen (2019) proposed a model for QoS prediction, in which K -medoids clustering is used for calculation of a set of similar users, and then trust-aware CF combines the trust values of clustered users. Ding et al. (2020) developed a prediction model based on user and service clustering (k -means++) by extracting context features of services according to WSDL files of web services and of users according to users' QoS history service request. Katarya and Verma (2017) applied K -means clustering using cuckoo search to recommend QoS values to a new user based on similar users in clustering. Wu et al. (2015) used the two phase K -Means clustering to handle the issue of untrustworthy users in QoS prediction. Lu et al. (2016) proposed a new prediction model based on probabilistic latent features such as network performance, user context, etc. Keshavarzi et al. (2019) proposed a hybrid approach in which missing data is filled with statistically semi-real data. Then, a modified version of k -medoids is used for the clustering of larger time-series data sets, and a lazy learning algorithm is used to train the model to predict the QoS values.

Dynamic programming is a useful approach to solve the optimisation and control parameter problem by using the principle of optimality in many fields. There are many variations of dynamic programming available i.e. Adaptive Dynamic Programming (ADP), Heuristic dynamic Programming (HDP), Dual Heuristic Programming (DHP), etc. There is a little difference among these dynamic programming approaches. Lu et al. (2014) used the neural dynamic programming approach to improve the dynamic performance of PSO. Luo et al. (2015a) used the Takagi-Sugano fuzzy inference system for the implementation of the ADP parameter tuner and also provided the convergence results to guarantee the scalability of the proposed system. In this paper, we have used the ADP with the implementation of

backpropagation neural network to optimise the parameters of our proposed models.

3 Problem definition

Many issues that can occur in the collection of QoS values of different cloud services have already been discussed in the introduction section. These issues lead to a high range of data sparsity and uncertainty in QoS collected data. This uncertainty in data may result in inaccurate cloud service QoS prediction and a large deviation in service selection. The collection of QoS values collected from different users can be described by a user-item matrix as shown in Table 1. The following are some of the notations that are used in this table.

Table 1 User-item matrix (R)

Users/Items	I1	I2	I3	I4
U1	.25	1.56	.97	-
U2	-	.67	-	.09
U3	.89	-	.20	.77
U4	.69	.91	-	.54

$U = \{u_1, u_2, \dots, u_m\}$ represents the cloud services users, where m represents the total number of users. $I = \{i_1, i_2, \dots, i_n\}$ represents the cloud services items with similar functionality, where n represents the total number of service items.

Let R of size $m \times n$ be the matrix and each existing entry in this matrix represents a certain QoS value of a cloud service invoked by the service user. Each user provided a set of QoS invocation data i.e. user u_i provides $R_i = \{r_{i1}, r_{i2}, \dots, r_{in}\}$ as the i -th pattern of user-item matrix. So, $R = \{R_1, R_2, R_3, \dots, R_m\}$ is a set of m pattern provided by different m users and each pattern is n -dimensional pattern corresponding to n services. The example of a user-item rating matrix used in this paper is shown in Table 1 where, missing values are represented by -(Hyphen).

The objective of this paper is to predict missing QoS values in the user-item matrix by using proposed QoS prediction models that may help cloud users to use these predicted QoS values in the selection of suitable cloud services according to their requirements.

4 Proposed models

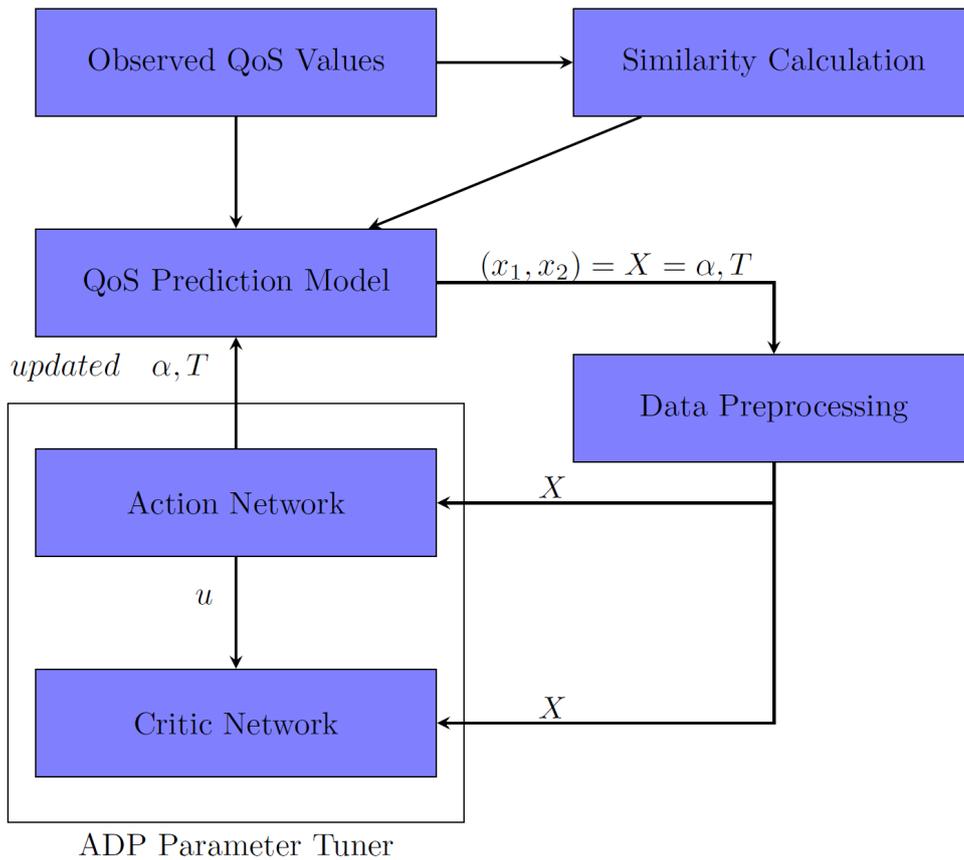
It is difficult for an individual user to try all the cloud services, to know the QoS values of different cloud services, the user has to rely on the information (QoS values) collected by other cloud users but this collected information is not complete. Therefore, the average of all the QoS values collected from other users about unknown cloud service is used by researchers in basic prediction method although this method has ignored the computing background environment such as internet conditions, locations, service features, etc.

In our research, we have proposed two QoS prediction models named Optimised Matrix Factorisation prediction model and Optimised Fuzzy C-Mean Clustering prediction model. In the Optimised Matrix Factorisation prediction model, PCC is used to calculate the similarity between users and then Matrix Factorisation (MF) is used for prediction of missing QoS values by making use of the calculated similarity values. MF is considered as the most successful technique to build a prediction model and it has been studied in electronic commerce for many years. It recommends services according to similar users with an active user or similar item with the item rated by the active user and we have used the PCC here to calculate the similarity between users or items. There are also some other techniques to calculate similarity like spearman coefficient, Jaccard's similarity coefficient, cosine coefficient, etc. but these similarity measures are not enough to find the effective similar users, especially those users who have only rated a small number of items. PCC utilises the ratings in cold user conditions and provides better performance combined with MF than other similarity measures in terms of RMSE.

In the Optimised Fuzzy C-Mean Cluster prediction model, we have used the improved version of fuzzy clustering i.e. FCM clustering. The benefit of FCM is that it provides the membership value for each data point in each cluster. Here, we combined the advantages of PCC with FCM to increase the accuracy of prediction. This model has one advantage over MF model i.e. if a new user has no invocation experience with any cloud service then according to MF based model, new user has zero similar users and correspondingly zero similarity value with other users but FCM model calculates its similarity with the average value of all cluster centres to complete the recommendation which is consistent with the new user situation.

Here, we have used the backpropagation-based ADP parameter tuner to optimise the values of parameters used in proposed models. In backpropagation neural network, weights are updated backward from output to input and due to this property, it provides accurate results compared to other neural networks. Figure 1 shows the flow graph of the proposed QoS prediction model with the ADP parameter tuner to optimise its parameter values.

Figure 1 Flow graph of proposed approach



4.1 Optimised matrix factorisation (Opt-MF) prediction model

Here, the matrix factorisation technique is used for the calculation of QoS values. It uses the method of gradient descent to approximate the factorised matrices and also considered the similarity value between users for accurate prediction. It has been successfully applied in spectral data analysis and text mining. It is the most accurate approach to reduce the problem from a high level of sparsity in large databases and this is the main reason to select this model in our research because as discussed in Section 1 (Introduction), these QoS prediction models have a large amount of data sparsity as their main problem. The only problem with MF is that it is slow and has a computationally expensive nature.

4.1.1 Pearson coefficient correlation (PCC)

PCC is the most widely used similarity measure in collaborative filtering. PCC uses the following formula to compute the similarity between two users i.e. $PCC(u, v)$ (Sedgwick, 2012):

$$\frac{\sum_{i \in I_v \cap I_u} (R(u, i) - \bar{R}(u)) (R(v, i) - \bar{R}(v))}{\sqrt{\sum_{i \in I_v \cap I_u} (R(u, i) - \bar{R}(u))^2} \sqrt{\sum_{i \in I_v \cap I_u} (R(v, i) - \bar{R}(v))^2}} \quad (1)$$

where,

R : user-item matrix (see Table 1)

$I_v \cap I_u$ represent the set of common rating items by both users v and u .

$\bar{R}(u)$ represent the average QoS values of all the services used by the user u .

$\bar{R}(v)$ represent the average QoS values of all the services used by user v .

$R(u, i)$ represent the rating of item i by user u .

The value computed by PCC lies within the range of $[-1, 1]$. The larger value of similarity measure represent that the two users are more similar to each other and smaller value of similarity measure represent less or no similarity between two users. In PCC the accuracy of score increases when data is not normalised and it calculates similarity as the covariance of two user's rating divided by their standard deviation based on co-related items (Tan and He, 2017).

Hence by PCC measure, we can find out some user's pair that have more or less similarity to rating score on the same items based on the assumption that users with higher similarity on different types of items help recommender system to predict accurate results with higher probability. In PCC all neighbour ratings might not be equally valuable. We need to add some randomness by assigning weights to items that have high variance.

4.1.2 Matrix factorisation (MF)

Recommender systems frequently use matrix factorisation models to generate personalised recommendations for users. Matrix factorisation are effective techniques that allow us to discover the latent features on basis of which different items are rated by users. These techniques are popularly used to reduce the dimensionality of data by factorisation. The most common MF technique will project the user-item matrix $R_{m \times n}$ into low rank approximation by reducing down to two smaller matrices i.e. $X = A^T B$ where, $A \in R^{m \times l}$ is a matrix of $m \times l$, $B \in R^{l \times n}$ is a matrix of $l \times n$. l represent the rank of approximation matrix X and ($l=3$) in our research work.

Each row of A would represent the strength of association between a user and the features. Similarly each row of B represents the strength of association between an item and the features. Our main objective here is to minimise the error between the observed value and predicted value that can be calculated by using the following sum of squared equation for each user-item pair (Li and Lin, 2020):

$$\begin{aligned} \text{Min}(A, B) = e_{xy} = & \sum_{x=1}^m \sum_{y=1}^n R(x, y) - \left[\alpha (A_x^T B_y) \right. \\ & \left. + (1 - \alpha) \sum_{k \in T(x)} \text{Sim}_{xk} A_k^T B_y \right] \\ & + \frac{\beta}{2} \sum_{l=1}^l (\|A\|^2 + \|B\|^2) \end{aligned} \quad (2)$$

where,

α : tell us about how much data of itself and similar users, the active user will consider during the prediction of an unknown service item i .

β : is a regularisation factor that uses additional information to prevent the problem of over-fitting ($\beta=0.03$ in this paper).

T : defines the number of top similar users considered during the prediction of QoS value.

There are many techniques used to find low ranked matrix A and B . In this paper, we have first initialised these two matrices with random values and calculate the difference of the product of these matrix with original matrix and try to minimise this error iteratively. This method of iteration is called the gradient descent method whose main aim is to find a local minimum of the differences. The following weight updation equations are used for approximation of low rank matrices A and B that also minimise error.

$$A_{xk} = A_{xk} + (2 * \text{learningrate} * e_{xy} * B_{ky}) \quad (3)$$

$$B_{ky} = B_{ky} + (2 * \text{learningrate} * e_{xy} * A_{xk}) \quad (4)$$

Here, the learning rate is a constant and determines the length of steps in the direction of minimum. We have chosen a small value for the learning rate i.e. 0.002 because the large value of the learning rate means a larger length of steps that may have

a higher probability of missing the minimum value. By using the above updation rule, we can iteratively perform the operation until the error converges to minimum.

4.2 Optimised fuzzy C-mean clustering (Opt-FCM) prediction model

In this model, Fuzzy C-Mean clustering is used to predict the QoS value and ADP Parameter tuner is used to optimise the parameters in this prediction model. Some recent studies also show that the use of fuzzy logic in recommendation systems can provide much better results.

4.2.1 Fuzzy C-mean clustering (FCM)

Clustering is a way of modelling the concise representation of system behaviour by identifying the natural clustering of data from a large data set. Fuzzy C-mean clustering is a powerful tool of clustering for analysis of data and construction of models where each data point in the data set belongs to every cluster either with higher degree or lower degree of membership according to the closeness of data point with centre of cluster (Suganya and Shanthy, 2012).

The membership function (μ) in the FCM have the following constraints (Zhang et al., 2012):

- $\sum_{i=1}^{\text{No. of Clusters}} \mu_{ij} = 1 \quad \forall j = 1, 2, \dots, m$
- $0 < \sum_{j=1}^{\text{No. of elements in cluster}} \mu_{ij} < n$
 $\forall i = 1, 2, \dots, \text{No. of Clusters}$
- $0 < \mu_{ij} < 1$, where $1 < i < m, i < j < \text{No. of Clusters}$

The working of FCM is as follows: Each cluster is randomly initialised as their cluster centre point, which is used to represent the mean of that cluster and each data point is assigned a membership value for each cluster by the FCM. FCM repeats this process by updating the cluster centre and membership value of each data point until cluster centres don't take their right position in the data set. At each iteration, the membership value and cluster centre are calculated by the following equations (5) and (6), respectively (Zhang et al., 2012).

$$\mu_{ij} = \frac{1}{\sum_{c=1}^C \left(\frac{d_{ij}}{d_{ic}} \right)^{\left(\frac{2}{m-1} \right)}} \quad (5)$$

$$v_j = \frac{\sum_{i=1}^n (\mu_{ij}^m * r_i)}{\sum_{i=1}^n (\mu_{ij}^m)} \quad \forall j = 1, 2, \dots, C, \quad 1 \leq m < \infty \quad (6)$$

where

n = No. of data points in data sets

v_j = j -th cluster centre

C = No. of Clusters

μ_{ij} = membership of i -th data to j -th cluster centre

The main aim of FCM Clustering is to minimise the following objective function (Zhang et al., 2012):

$$J(U, V) = \sum_{i=1}^n \sum_{j=1}^C (\mu_{ij})^m * (d_{ij}^2) \quad (7)$$

where d_{ij} is a Euclidean distance function used to calculate the difference between i -th data and j -th cluster centre i.e. $d_{ij} = \|x_i - v_j\|$ where $\|*\|$ is any norm measure the similarity between any measured data x_i and the cluster centre v_j .

The pseudo code of FCM shown as follow:

FCM Clustering

Input: Data Set, No. of Clusters (C), No. of Iterations (z), Threshold Value θ

Output: Centre Matrix (V), Membership Matrix (U).

1. Choose the initial cluster centre V randomly.
 2. Initialise the membership value matrix U using the distance between each data point and cluster centre matrix V using equation (5).
 3. Update centre matrix V based on membership matrix U using equation (6).
 4. Update membership value matrix using equation (5).
Repeat steps 4 and 5 until minimum J (objective function) is achieved.
 5. Return updated matrices U and V .
-

4.2.2 Prediction model

The missing QoS values using fuzzy C-means clustering can be predicted as follows:

$$R(x, y) = \alpha \left(\sum_{c=1}^C \mu_{cx} * v(c, y) \right) + (1 - \alpha) \left(\frac{\sum_{k \in T(x)} \text{sim}(x, k) * (R((k, y) - \bar{R}_k))}{\sum_{k \in T(x)} \text{sim}(x, k)} \right) \quad (8)$$

where,

μ_{cx} : represent the membership value for user x in the c_{th} cluster.

v_{cy} : represent the centre of c_{th} cluster for service y .

$\text{sim}(x, k)$: represent the similarity between user x and user k .

$R(x, y)$: represent original QoS value for service y rated by user k .

\bar{R}_k : represent the mean value of all QoS rating given by user k .

The above equation consists of two parts: Part 1 is obtained by multiplication of membership value by cluster centre

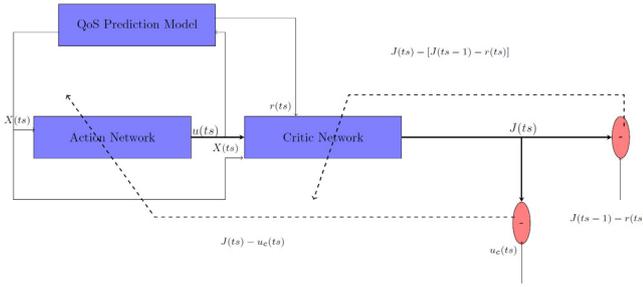
values and part 2 is obtained by considering the QoS values of similar users. The value of α decides that how much percentage value of part 1 and part 2 must be considered for accurate prediction of missing QoS values.

5 ADP parameter tuner

The existing prediction approaches use the static set of values for parameters and these static values don't explore the entire search space of parameters and lack adaptability, due to which performance will degrade and it also provide the unsatisfactory predicted QoS values. To solve these issues, we have used ADP parameter tuner to optimise the parameter values of these QoS prediction approaches.

ADP has the capability to learn and optimise the control variable over time. In our research, we have used the model based architecture of ADP, where networks are used to predict the future system variables and function. The detailed implementation of ADP parameter tuner with action-critic network architecture is shown in Figure 2.

Figure 2 Detailed Implementation of ADP parameter tuner



There are two networks in ADP parameter tuner i.e. Action and Critic neural network. The solid lines in tuner represent signal flow and dashed lines are the Back Propagation paths for parameter tuning of the network. $X(ts)$ is the model output which is taken as input by both action network and critic network. $r(ts)$ is a reinforcement signal from the external environment, it is a function to measure the distance of predicted result based on parameter $u(ts)$ from original result. It tells us the effectiveness of control variable $u(ts)$ and controls the convergence speed of network. The critic network takes the values of $X(ts)$ and $u(ts)$ as input to approximate the critic network output with the objective function $R(ts)$. The critic network is determined by Bellman's error equation, so the critic network is trained to achieve the approximate function $J(ts)$ and to balance the error Bellman equation (Lu et al., 2014) i.e.

$$e_{Bellman}(ts) = (r(ts) + J(ts) - J(ts-1)) \quad (9)$$

The error between the function $R(ts)$ and critic network output $J(ts)$ is used to train the action neural network. The

output of action network i.e. $u(ts)$ is used to make the critic output as close to U_c as possible, where U_c is the desired objective value of $R(ts)$, and set to 0 without loss of generality in this paper i.e. $J(ts) - R(ts) = 0$.

Here $R(ts)$ is the cost function, which is defined based on the reinforcement signal $r(ts)$ (Luo et al., 2015b) i.e.

$$R(ts) = \sum_{i=1}^{\infty} \gamma^i * r(ts+i) \quad (10)$$

and the main aim of our tuning problem is to minimise this cost function where γ is a constant factor and lies between 0 and 1 i.e. $\gamma \in [0,1]$. The pseudo code of ADP Parameter Tuner is shown as follow:

ADP Parameter Tuner

Action Network:

$$u = ActNet(X, w_a)$$

Where Act Net: Action Network

$X \rightarrow$ Input Vector

$w_a \rightarrow$ Weight of Act Net

$u \rightarrow$ Output of Act Net

$J = CritNet(X, u, r, w_c)$

Where Crit Net: Critic Network

$w_c \rightarrow$ Weight of Critic network

$u \rightarrow$ Output of Act Net

$r \rightarrow$ Reinforcement Signal

$X \rightarrow$ Input Vector

1. **Initialise** $X(0) = (\alpha, T)$
2. Randomly Initialise Weights i.e. $w_a(0), w_c(0)$.
3. $u(0) \leftarrow ActNet(X(0), w_a(0))$
4. $J(0) \leftarrow CritNet(X(0), u(0), r(0), w_c(0))$
5. **for** $ts = 1$ to max number of steps
6. $X(ts) = X(ts-1)$ (Obtained state value from QoS prediction Model)
7. $r(ts) = r(ts-1)$
8. $w_a(ts) = w_a(ts-1)$
9. $w_c(ts) = w_c(ts-1)$
10. $u(ts) = ActNet(X(ts), w_a(ts))$
11. $J(ts) = CritNet(X(ts), u(ts), r(ts), w_c(ts))$
- Weight Updation Phase of Critic Network
12. $E_c(ts) = \frac{1}{2} (J(ts) - (J(ts-1) - r(ts)))^2$
13. cyc=0;

14. **while** ($E_c(ts) > T_c$ & $cyc > N_c$)
15. Update Weights of Critic Network
Using equations (32) to (35).
16. $J(ts) = \text{Crit Net}(X(ts), u(ts), r(ts), w_c(ts))$
17. $E_c(ts) = \frac{1}{2} (J(ts) - (J(ts-1) - r(ts)))^2$
18. $cyc = cyc + 1$;
19. **end while** (Training Phase of Critic Network)
Weight Updation Phase of Action Network
20. $E_a(ts) = \frac{1}{2} (J(ts) + u(ts) - U_c)^2$
21. $cyc = 0$;
22. **while** ($E_a(ts) > T_a$ & $cyc > N_a$)
23. Update Weight of Action Network
Using equations (20) to (23).
24. $u(ts) = \text{Act Net}(X(ts), w_a(ts))$
25. $J(ts) = \text{Crit Net}(X(ts), u(ts), r(ts), w_c(ts))$
26. $E_a(ts) = \frac{1}{2} (J(ts) + u(ts) - U_c)^2$
27. $cyc = cyc + 1$;
28. **end while** (Training Phase of Action Network)
(Send $u(ts)$ to QoS prediction Model)
29. **end while** (Loop for max number of steps)

5.1 The neural network

The multilayered back propagation neural network is used to implement both action and critic network. The sigmoidal function $\theta(x)$ maps the neural network results in between 0 and 1 and it is used by both hidden and output layers of critic and action network in ADP tuner. The sigmoidal function for neural networks is defined as:

$$\theta(x) = \frac{1 - \exp^{-x}}{1 + \exp^{+x}} \quad (11)$$

5.2 Action network

The architecture of action network have $2 \times 6 \times 2$ units of neurons in input, hidden and output layer, respectively. In this, $[x_1(ts), x_2(ts)] = [\alpha(ts), T(ts)]$ are the inputs to action network and $u_1(ts), u_2(ts)$ are the corresponding output of action network.

5.2.1 Objective function

The error function of action network is defined as follow (Zhang et al., 2012):

$$e_a(ts) = J(ts) - R(ts) \quad (12)$$

and the objective function to be minimised is defined as follows (Zhang et al., 2012):

$$E_a(ts) = \frac{1}{2} * e_a(ts) \quad (13)$$

when the objective function is at optimal stage, the squared prediction error of the action network must be at its minimum level.

5.2.2 Weight updation rule

The weight learning rule for action network is designed as follows (Lu et al., 2014):

$$w_a(ts+1) = w_a(ts) + \Delta w_a(ts) \quad (14)$$

$$\Delta w_a(ts) = \left[-\frac{\partial E_a(ts)}{\partial w_a(ts)} \right] \quad (15)$$

$$\frac{\partial E_a(ts)}{\partial w_a(ts)} = \frac{\partial E_a(ts)}{\partial J(ts)} * \frac{\partial J(ts)}{\partial w_a(ts)} \quad (16)$$

where

- $w_a(ts)$: represent the weight parameter in the action neural network.

5.2.3 Learning in action network

The action network is trained until the critic output value i.e. $J(ts)$ approximates the objective function $R(ts)$. The output of action network is calculated as follow (Lu et al., 2014):

$$u_k(ts) = \frac{1 - \exp^{-\sum_{i=1}^{H_{na}} w_{aki}^{(2)} * g_i(ts)}}{1 + \exp^{-\sum_{i=1}^{H_{na}} w_{aki}^{(2)} * g_i(ts)}} \quad k = 1, 2 (\text{no. of output neurons}) \quad (17)$$

$$g_i(ts) = \frac{1 - \exp^{-h_i(ts)}}{1 + \exp^{-h_i(ts)}} \quad i = 1, 2, \dots, H_{na} \quad (18)$$

$$h_i(ts) = \sum_{j=1}^{I_{na}} w_{aj}^{(1)} * X_j(ts) \quad (19)$$

where

$u_k(ts)$: output of k -th neuron of output layer in action network.

$h_i(ts), g_i(th)$: Input and output of the i -th hidden node.

$X_j(ts)$: Input to the j -th node.

H_{na} and I_{na} : Number of neurons in hidden and input layer, respectively.

$w_{aj}^{(1)}$: weight parameter between j -th input neuron and i -th hidden neuron of action network.

$w_{aki}^{(2)}$: weight parameter between i -th hidden neuron and k -th output neuron of action network

The procedure of back propagation rule applied to action network is as follow (Lu et al., 2014):

- $\Delta w_{ai}^{(2)}$: Action network weight adjustment from hidden layer to output layer.

$$\Delta w_{ai}^{(2)} = \left[-\frac{\partial E_a(ts)}{\partial w_{ai}^{(2)}} \right] \quad (20)$$

$$\begin{aligned} \frac{\partial E_a(ts)}{\partial w_{ai}^{(2)}} &= \frac{\partial E_a(ts)}{\partial J(ts)} * \frac{\partial J(ts)}{\partial p_i(ts)} * \frac{\partial p_i(ts)}{\partial q_i(ts)} * \frac{\partial q_i(ts)}{\partial X_k(ts)} * \frac{\partial X_k(ts)}{\partial w_{ai}^{(2)}(ts)} \\ &= e_a(ts) * \sum_{j=1}^{H_{na}} \left[w_{ci}^{(2)}(ts) * \left(\frac{1}{2} (1 - p_i^2(ts)) \right) * w_{cij}^{(1)}(ts) \right] \\ &\quad * \left[\frac{1}{2} * (1 - X_k(ts)) \right] * g_i(ts) \end{aligned} \quad (21)$$

- $\Delta w_{aij}^{(1)}$: Action network weight adjustment from input layer to hidden layer.

$$\Delta w_{aij}^{(1)} = \left[-\frac{\partial E_a(ts)}{\partial w_{aij}^{(1)}} \right] \quad (22)$$

$$\begin{aligned} \frac{\partial E_a(ts)}{\partial w_{aij}^{(1)}} &= \frac{\partial E_a(ts)}{\partial J(ts)} * \frac{\partial J(ts)}{\partial p_i(ts)} * \frac{\partial p_i(ts)}{\partial q_i(ts)} * \frac{\partial q_i(ts)}{\partial X_k(ts)} \\ &\quad * \frac{\partial X_k(ts)}{\partial g_i(ts)} * \frac{\partial g_i(ts)}{\partial h_i(ts)} * \frac{\partial h_i(ts)}{\partial w_{aij}^{(1)}} \\ &= e_a(ts) * \sum_{j=1}^{H_{na}} \left[w_{ci}^{(2)}(ts) * \left(\frac{1}{2} (1 - p_i^2(ts)) \right) * w_{cij}^{(1)}(ts) \right] \\ &\quad * \left[\frac{1}{2} * (1 - X_k(ts)) \right] * w_{aki}^{(2)}(ts) \\ &\quad * \left[\frac{1}{2} (1 - g_i^2(ts)) \right] * X + j(ts) \end{aligned} \quad (23)$$

5.3 Critic network

The non-linear architecture of critic neural network used in our ADP parameter tuner have $5 \times 6 \times 1$ number of neurons. Critic network is applied to approximate the cost function and its input consists of the following parameters:

- $x_1(ts), x_2(ts) = (\alpha, T)$ i.e. output of QoS prediction model whose values need to be optimised.
- $u(ts)$ i.e. output from the action network.
- $r(ts)$ i.e. reinforcement signals form the QoS prediction model.

5.3.1 Error function

The approximated error for critic neural network is defined as Bellman's error i.e. (Zhang et al., 2012):

$$e_c(ts) = e_{Bellman}(ts) = J(ts) - [J(ts-1) - r(ts)] \quad (24)$$

and the objective function of critic network is to minimise this error function $E_c(ts)$ that is used to update the parameter in network (Zhang et al., 2012):

$$E_c(ts) = \frac{1}{2} * e_c^2(ts) \quad (25)$$

5.3.2 Weight updation rule

Backpropagation chain rule applied to updating the weights in critic neural network is:

$$w_c(ts+1) = w_c(ts) + \Delta w_c(ts) \quad (26)$$

$$\Delta w_c(ts) = \left[\frac{-\partial E_c(ts)}{\partial w_c(ts)} \right] \quad (27)$$

$$\frac{\partial E_c(ts)}{\partial w_c(ts)} = \frac{\partial E_c(ts)}{\partial J(ts)} * \frac{\partial J(ts)}{\partial w_c(ts)} \quad (28)$$

where

- $w_c(ts)$ represent the weight parameters in the critic neural network.

5.3.3 Learning in critic network

Hidden layer and output layer in critic network use the sigmoidal activation function. The output of critic network is calculated as follows:

$$J(ts) = \frac{1 - \exp^{-\sum_{i=1}^{H_{nc}} w_{ci}^{(2)} * p_i(ts)}}{1 + \exp^{-\sum_{i=1}^{H_{nc}} w_{ci}^{(2)} * p_i(ts)}} \quad (29)$$

$$p_i(ts) = \frac{1 - \exp^{-q_i(ts)}}{1 + \exp^{-q_i(ts)}} \quad i = 1, 2, \dots, h_{nc} \quad (30)$$

$$q_i(ts) = \sum_{j=1}^{I_{nc}} w_{cij}^{(1)} * X_j(ts) \quad (31)$$

where

$X_j(ts)$: Inputs to the input layer of critic network.

$q_i(ts)$: Input to i -th hidden node of critic network.

$p_i(ts)$: output of the i -th hidden node of critic network.

H_{nc} and I_{nc} : Number of neurons in hidden and input layer, respectively.

$w_{c_{ij}}^{(1)}$: weight parameter between j -th input neuron and i -th hidden layer of critic network.

$w_{c_{(i)}}^{(2)}$: weight parameter between i -th hidden node and output node.

By applying backpropagation chain rule, the weight parameters updation in critic network is performed as follow:

- $\Delta w_{c_i}^{(2)}$: Critic network weight adjustment between the hidden and output layer.

$$\Delta w_{c_i}^{(2)} = \left[\frac{-\partial E_c(ts)}{\partial w_{c_i}^{(2)}} \right] \quad (32)$$

$$\frac{-\partial E_c(ts)}{\partial w_{c_i}^{(2)}} = \frac{-\partial E_c(ts)}{\partial J(ts)} * \frac{\partial J(ts)}{\partial w_{c_i}^{(2)}} = e_c(ts) * p_i(ts) \quad (33)$$

- $\Delta w_{c_{ij}}^{(1)}$: Critic network weight adjustment between the input and hidden layer.

$$\Delta w_{c_{ij}}^{(1)} = \left[-\frac{\partial E_c(ts)}{\partial w_{c_{ij}}^{(1)}(ts)} \right] \quad (34)$$

$$\begin{aligned} \frac{\partial E_c(ts)}{\partial w_{c_{ij}}^{(1)}(ts)} &= \frac{\partial E_c(ts)}{\partial J(ts)} * \frac{\partial J(ts)}{\partial p_i(ts)} * \frac{\partial p_i(ts)}{\partial q_i(ts)} * \frac{\partial q_i(ts)}{\partial w_{c_{ij}}^{(1)}(ts)} \\ &= e_c(ts) * w_{c_i}^{(2)}(ts) * \frac{1}{2} * (1 - p_i^{(2)}(ts)) * x_j(ts) \end{aligned} \quad (35)$$

6 Experimental results

This section provides the details of data sets, performance metrics and parameters used in this paper.

6.1 Data set

All simulations are performed on a Dell desktop with an intel core i5 CPU and 4.00 GB RAM, running Microsoft Windows 7 Operating System. The simulations are implemented with MATLAB. For experimental purposes, we have considered the database from real world service database (Zheng et al., 2014) latest updated on 27/04/2016, which has data for both response time and throughput of range from 0.001 to 19.969 and 1.0 to 1000.0, respectively. It contains real-world QoS evaluation results from 339 users in format (User ID, IP Address, Country, Continent, Latitude, Longitude, Region, City) on 5825 web services in format (User ID, IP Address, Country, Continent, Latitude, Longitude, Region, City). So the total number of reviews is 19,74,675.

We have validated our models on the response time data set. It is a time duration between sending a request from the user and receiving a response. Selected data set is the widely used data set by researchers in quality recommender system domain.

To demonstrate the performance of proposed models, we divide the database in training and testing set in ratio of 90:10, 80:20 and 70:30, i.e. 10%, 20% and 30% of all data are selected as testing data, and the remaining data used for training of proposed models in each case, respectively.

6.2 Performance metrics

Here, Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Relative Error (MRE) and Mean Absolute Relative Error (MARE) are used to measure the accuracy of our proposed models with other existing approaches.

6.2.1 RMSE

It is a standard statistical metric to measure the difference between predicted measure and observed value. It is used to describe a complete picture of the error distribution and most suitable to describe normal distributed errors (Luo et al., 2015).

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\text{observed value} - \text{predicted value})^2} \quad (36)$$

6.2.2 MAE

It is another standard method to describe uniformly distributed errors (Luo et al., 2015).

$$MAE = \frac{1}{n} \sum_{i=1}^n |\text{observed value} - \text{predicted value}| \quad (37)$$

6.2.3 MRE

Mear relative error determines how large the absolute error value is compared to known value. It is the average of the relative errors (Finnie et al., 1997).

$$MRE = \frac{1}{n} \sum_{i=1}^n \frac{\text{observed value} - \text{predicted value}}{\text{observed value}} \quad (38)$$

6.2.4 MARE

The mean absolute related error is calculated as follow (Finnie et al., 1997):

$$MARE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\text{observed value} - \text{predicted value}}{\text{observed value}} \right| \quad (39)$$

6.3 Parameter setting

The following is the description of the parameters used in the experiment and the associated value of these parameters is shown in Table 2.

The notation of parameters are:

$N_c \rightarrow$ Internal Cycle of Critic Network

$N_a \rightarrow$ Internal Cycle of Action Network

$T_c \rightarrow$ Training error threshold value for Critic Network

$T_a \rightarrow$ Training error threshold value for Action Network

$X \rightarrow$ Input State i.e. (α, T)

Table 2 Values of parameter used in ADP tuner

Parameter	N_c	N_a	T_c	T_a	N_{hc}	N_{ha}
User	40	100	0.05	0.005	6	6

6.4 Performance comparison

This subsection presents the detailed comparative analysis of proposed model named optimised FM prediction model and optimised FCM prediction model with each other and some other basic prediction techniques i.e. User Mean, Item Mean and Item PCC QoS prediction techniques.

6.4.1 Comparison between Opt-FM and Opt-FCM

The main objective of our research is to improve the performance of matrix factorisation and fuzzy C-mean clustering using ADP parameter tuner. Therefore, it is also necessary to find out the detailed comparative analysis of these two techniques. Performance comparison of these two techniques performed on different kinds of sparse databases. To simulate the sparsity situation of matrix, we randomly remove some entries like 10%, 20%, 30% from the original response time matrix and then run both models on these sparse matrix. The detailed average result of simulation of our proposed models for different sparsity matrices in terms of performance indicators is shown in Table 3.

We have simulated Optimised Fuzzy C-Mean clustering (Opt-FCM) for a different number of clusters to obtain the optimal value of clusters that provide more accurate results. In the case of all data metrics i.e. with 10%, 20%, 30% sparsity, for the number of cluster = 16, 32, 64, all performance

measures have similar or near value to each other as shown in Table 3. It was found that the number of clusters 16 yields better results, and if we further increase the number of clusters, the cost of our objective function also increases and consumes more time. The Opt-FCM model with 16 numbers of clusters also outperformed the result of Opt-FCM with the number of clusters 4 and 8. In further discussion, we will consider the Opt-FCM model with 16 number of clusters for comparison with other techniques. The Opt-FCM model with 16 number of clusters provides better results than Opt-MF in terms of performance metrics i.e., as shown in Table 3, the RMSE and MAE values (0.76 and 0.13, respectively) of OPT-FCM is much less than RMSE and MAE (16.27 and 2.92, respectively) of Opt-MF model with 10%. Thus, Opt-FCM outperformed Opt-MF in every case of data sparsity.

6.4.2 Statistical analysis of Opt-FM and Opt-FCM

In this subsection, we have performed a statistical analysis of proposed models. Precision and ANOVA tests have been performed on Opt-MF and Opt-FCM with 16 clusters. The following precision formula is used to measure the performance of each model:

$$\text{Precision} = \frac{1}{n} \sum_{i=1}^n (\text{observed val} - \text{predicted val}) \leq 0.2 * 100\% \quad (40)$$

The results of precision and ANOVA tests are shown in Table 4. The difference in precision between Opt-FCM and Opt-MF is approximately 4 to 10 percent. Figure 3 shows the graphical representation of precision on different values of matrix sparsity. ANOVA test is used to confirm the difference between techniques in the form of p -value and in the case of 10% and 20% matrix sparsity, the results of our proposed models are not significant as the p -value is more than 0.05. Further the sparsity of the matrix increases, proposed models start showing significant differences in ANOVA test results.

Table 3 Average results

Average results of our proposed models							
Sparsity	Metrics	Opt-MF	Opt-FCM4	Opt-FCM8	Opt-FCM16	Opt-FCM32	Opt-FCM64
=10%	RMSE	16.26675	0.741428	0.758521	0.725599	0.72859	0.724049
	MAE	2.912187	0.129098	0.132787	0.127809	0.1278	0.125645
	MRE	-6.67174	0.298274	0.302367	0.289324	0.287	0.276317
	MARE	6.73281	0.301580	0.318654	0.299459	0.29643	0.287179
=20%	RMSE	16.01663	1.039993	1.05079	1.038088	1.082546	1.062038
	MAE	4.203416	.253023	.256002	.249273	.267184	.260783
	MRE	-8.82048	.529082	.536399	508177	585036	565326
	MARE	8.837919	.564715	.562505	.531134	.604053	.587448
=30%	RMSE	17.10742	1.426937	1.390444	1.382584	1.40424	1.382584
	MAE	5.690092	0.468204	0.449989	.444953	0.4557	0.444953
	MRE	-12.7477	1.142322	1.071416	1.050295	1.091122	1.050295
	MARE	12.76913	1.152764	1.087441	1.068897	1.07421	1.068897

Figure 3 Precision plot of Opt-MF and Opt-FCM

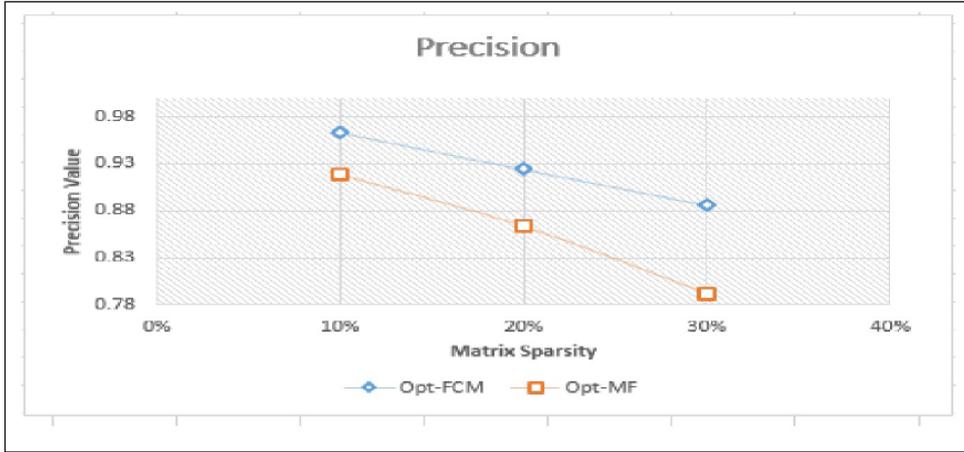


Figure 4 Distribution of predicted values by Opt-MF and Opt-FCM

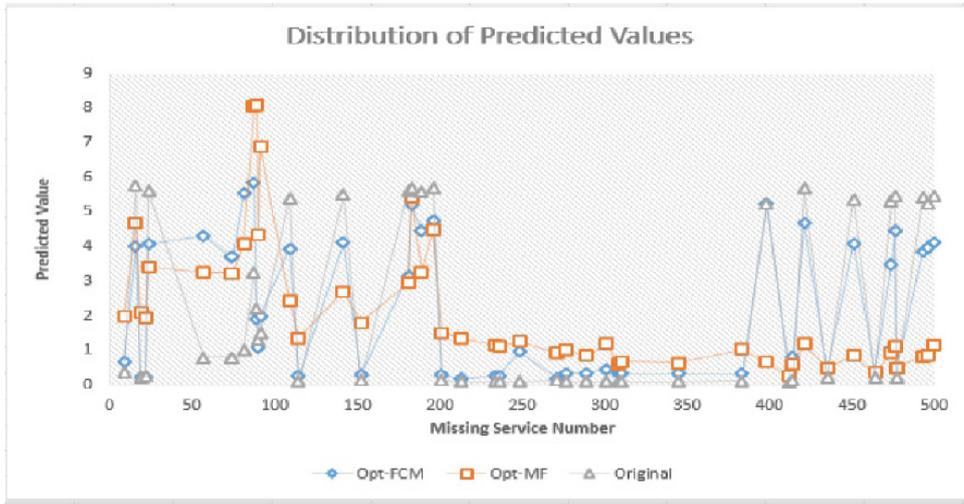


Table 4 Result of precision and ANOVA test

Sparsity	Opt-FCM (x)	Opt-MF (y)	Precision Diff (x-y)	P-Value
10%	96.29%	91.88%	4.41%	0.2245
20%	92.41%	86.41%	6%	0.1045
30%	88.59%	79.24%	9.35%	0.0166

Figure 4 presents the predicted response time results by proposed models on first 500 cloud services. The triangles show the actual QoS values for missing entries, squares show the values predicted by Opt-MF and diamonds show the values predicted by Opt-FCM models, etc. We can see the variation between actual value and predicted value by Opt-MF as compared to value predicted by Opt-FCM and our main motive of this proposed models is to minimise this difference and to achieve satisfactory QoS prediction performance.

6.4.3 Comparison with other basic prediction techniques

Here, some of the basic QoS prediction i.e. User Mean (UMean), Item Mean (IMean), Item PCC (IPCC) are

considered as preferred techniques for comparative analysis with proposed techniques in terms of performance indicators. Therefore, based on optimised parameter values, detailed comparison of our proposed models with other basic techniques are shown in Table 5. It is concluded that Opt-FCM with 16 clusters gives better values for different performance indicators and has produced better results than Opt-MF model and other basic prediction techniques.

Table 6 provides improved results in the form of percentage change in terms of performance metrics used such as RMSE, MAE, MRE, MARE. It also concludes that the Opt-FCM model outperforms other basic approaches and shows the percentage improvement over Opt-MF. Most of the researchers have used only MAE as their performance indicators however, we have taken four performance indicators to compare the result of our models with other techniques in our research work.

There is no randomisation in the result of UMean and IPCC techniques yet the result of Opt-FCM and Opt-MF get affected by the outliers that obtained by QoS prediction model on worst value of α and T during parameter optimisation. If we remove these outlier values in results, the

performance of proposed models show much improvement in comparison to basic techniques. Optimal range that produce minimum error in proposed models lies between [7 and 25] for parameter T and [0.002–0.04] for parameter α and as the value of α and T increases the value of performance indicators also starts getting worst. The value of performance measure is at maximum or worst when the value of α and T approaches towards its max value or near to its upper limit.

Table 5 Min results at optimal value of parameters

Optimal results						
Sparsity	Metrics	Opt-MF	Opt-FCM (16)	UMean	IMean	IPCC
=10%	RMSE	0.570317	0.39953	0.7854	0.4950	0.6010
	MAE	0.082753	0.048258	0.1332	0.0685	0.1337
	MRE	-0.1765	-0.01106	-0.3958	-0.0827	0.4123
	MARE	0.2011	0.05267	0.4151	0.1000	0.4199
=20%	RMSE	0.756947	0.600655	1.1282	0.6950	0.8278
	MAE	0.172622	0.106276	0.2682	0.1366	0.2534
	MRE	-0.352806	-0.01382	-0.7823	-0.1647	0.7517
	MARE	0.394670	0.110238	0.8213	0.2002	0.7678
=30%	RMSE	0.908113	0.79499	1.3693	0.8410	0.9855
	MAE	0.264463	0.0178244	0.4007	0.2051	0.3612
	MRE	-0.59297	0.021803	-1.1809	-0.2484	1.0497
	MARE	0.647775	0.152465	1.2391	0.3012	1.0750

Table 6 Percentage improvement of Opt-FCM (16) on other techniques

Improvements result in percentage						
Sparsity	Metrics	Opt-MF	UMean	IMean	IPCC	
=10%	RMSE	29.94%	49.12%	19.27%	33.51%	
	MAE	41.59%	63.73%	29.48%	63.87%	
	MRE	93.73%	97.19%	86.57%	97.31%	
	MARE	73.80%	87.30%	47.30%	87.44%	
=20%	RMSE	20.63%	46.75%	13.56%	27.43%	
	MAE	38.47%	60.36%	22.18%	58.05%	
	MRE	96.08%	98.23%	91.62%	98.16%	
	MARE	72.08%	86.58%	44.95%	85.64%	
=30%	RMSE	12.45%	41.94%	5.46%	19.33%	
	MAE	32.62%	55.52%	13.11%	50.66%	
	MRE	96.32%	98.15%	91.22%	97.92%	
	MARE	76.44%	87.69%	49.36%	85.81%	

When the matrix sparsity is 10% the values of MAR and MARE are better than the values when the matrix sparsity is 20% thus, it concludes that the matrix with more filled entries provide more information for prediction of missing values as compared to sparse matrix.

6.5 Discussion

In the above subsections, we described the data set and experiments used in our research work to measure the validity of the proposed models and it is concluded that the results obtained using Fuzzy C-Mean clustering (with 16 clusters) method are much better than the results of Matrix Factorisation method. In every data sparsity case, results of our proposed work are also better than the results obtained by basic prediction techniques i.e. an improvement of 49.12% and 63.73% on UMean, 19.27% and 29.48% on IMean, 33.51% and 63.87% on IPCC in terms of RMSE and MAE respectively. Statistical analysis (i.e. precision and ANOVA test) is also performed on proposed approaches to prove the accuracy of results.

7 Conclusion and future work

Owing to the rapid growth of cloud services, comparisons and recommendations of functionally equivalent cloud services have become a time-consuming task. To handle the recommendation issues, two models namely Optimised Matrix Factorisation prediction model and optimised FCM clustering prediction model have been proposed in our research work. These models predict unknown QoS values of various cloud services. We have used the backpropagation neural network in the action and critic network of ADP tuner architecture and developed a weight updating method to handle the learning process of a backpropagation network that may help in the optimisation of the parameters of QoS prediction model. We have measured the performance of proposed models in terms of RMSE, MAE, MRE and MARE and compared with some other basic prediction techniques i.e. User Mean (UMean), Item Mean (IMean) and Item PCC (IPCC), etc. on a real-world public database to verify the effectiveness and accuracy of our research work. Experimental results show the improvement of 49.12% and, 63.73% on UMean, 19.27% and 29.48% on IMean, 33.51% and 63.87% on IPCC in terms of RMSE and MAE respectively. In the future, to improve the prediction of our proposed models' various other factors like online learning with time, scaling of user and service, etc. may be considered and we are also planning to use some hybrid approach with back propagation in our next research work.

References

- Chaudhuri, A. (2017) 'Hierarchical support vector regression for QoS prediction of network traffic data', *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, ACM, pp.1–15.
- Ding, F., Wen, T., Ren, S. and Bao, J. (2020) 'Performance analysis of a clustering model for QoS-aware service recommendation', *Electronics*, Vol. 9, pp.1–17.

- Feng, Y. and Huang, B. (2018) 'Cloud manufacturing service QoS prediction based on neighbourhood enhanced matrix factorization', *Journal of Intelligent Manufacturing*, Vol. 29, pp.1–12.
- Finnie, G.R., Wittig, G.E. and Desharnais, J-M. (1997) 'A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models', *Journal of Systems and Software*, Vol. 39, pp.281–289.
- Jin, Y., Guo, W. and Zhang, Y. (2019a) 'A time-aware dynamic service quality prediction approach for services', *Tsinghua Science and Technology*, Vol. 25, pp.227–238.
- Jin, Y., Wang, K., Zhang, Y. and Yan, Y. (2019b) 'Neighborhood-aware web service quality prediction using deep learning', *EURASIP Journal on Wireless Communications and Networking*, Springer, pp.1–10.
- Katarya, R. and Verma, O.P. (2017) 'An effective collaborative movie recommender system with Cuckoo Search', *Egyptian Informatics Journal*, Vol. 18, pp.105–112.
- Keshavarzi, A., Haghghat, A.T. and Bohlouli, M. (2019) 'Enhanced time-aware QoS prediction in multi-cloud: a hybrid k-medoids and lazy learning approach (QoPC)', *Computing*, Vol. 101, pp.1–27.
- Li, J. and Lin, J. (2020) 'A probability distribution detection based hybrid ensemble QoS prediction approach', *Information Sciences*, Vol. 519, pp.289–305.
- Li, S., Wen, J., Luo, F. and Ranzi, G. (2018) 'Time-aware QoS prediction for cloud service recommendation based on matrix factorization', *IEEE Access*, Vol. 16, pp.77716–77724.
- Liu, J. and Chen, Y. (2019) 'A personalized clustering-based and reliable trust-aware QoS prediction approach for cloud service recommendation in cloud manufacturing', *Knowledge-Based Systems*, Vol. 174, pp.43–56.
- Liu, J. and Chen, Y. (2019) 'HAP: a hybrid QoS prediction approach in cloud manufacturing combining local collaborative filtering and global case-based reasoning', *IEEE Transactions on Services Computing*, pp.1–14.
- Lu, W., Hu, X., Li, X. and Wei, Y. (2016) 'A new method of QoS prediction based on probabilistic latent feature analysis and cloud similarity', *International Journal of High Performance Computing and Networking*, Vol.9, pp.52–60.
- Lu, Y., Yan, D., Zhang, J. and Levy, D. (2014) 'Direct back propagation neural dynamic programming-based particle swarm optimisation', *Connection Science*, Vol. 26, pp.367–388.
- Luo, X., Luo, H. and Chang, X. (2015b) 'Online optimization of collaborative web service QoS prediction based on approximate dynamic programming', *International Journal of Distributed Sensor Networks*, Vol.11, pp.452–492.
- Luo, X., Lv, Y., Li, R. and Chen, Y. (2015a) 'Web service QoS prediction based on adaptive dynamic programming using fuzzy neural networks for cloud services', *IEEE Access*, Vol. 3, pp.2260–2269.
- Mansur, F., Patel, V. and Patel, M. (2017) 'A review on recommender systems', *Proceedings of the International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, IEEE, pp.1–6.
- Monika and Sangwan, O.P. (2019) 'QoS based scheduling techniques in cloud computing: systematic review', *International Journal of Computer Science and Information Security (IJCSIS)*, Vol. 17, pp.32–39.
- Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. (1994) 'GroupLens: an open architecture for collaborative filtering of netnews', *Proceedings of the ACM conference on Computer Supported Cooperative Work*, ACM, pp.175–186.
- Sangwan, O.P. (2018) 'Quality of service in dynamic resource provisioning: literature review', *Proceedings of the International Conference on Information, Communication and Computing Technology*, Springer, pp.44–55.
- Sedgwick, P. (2012) 'Pearson's correlation coefficient', *British Medical Journal*, Vol. 345, pp.44–83.
- Somu, N., Raman, G.M.R., Kaveri, A., Krithivasan, K. and Sriram, V.S.S. (2020) 'IBGSS: an improved binary gravitational search algorithm based search strategy for QoS and ranking prediction in cloud environments', *Applied Soft Computing*, Vol. 88, pp.1–48.
- Suganya, R. and Shanthi, R. (2012) 'Fuzzy c-means algorithm-a review', *International Journal of Scientific and Research Publications*, Vol. 2, pp.1–3.
- Tan, Z. and He, L. (2017) 'An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle', *IEEE Access*, Vol. 5, pp.27211–27228.
- Wu, C., Qiu, W., Zheng, Z., Wang, X. and Yang, X. (2015) 'Qos prediction of web services based on two-phase K-means Clustering', *Proceedings of the International Conference on Web Services (ICWS)*, IEEE, pp.161–168.
- Xu, J., Zheng, Z., Fan, Z. and Liu, W. (2016) 'Online personalized QoS prediction approach for cloud services', *Proceedings of the International Conference on Cloud Computing and Intelligence Systems*, IEEE, pp.32–37.
- Zhang, L., Zhang, B., Na, J., Huang, L. and Zhang, M. (2010) 'An approach for web service QoS prediction based on service using information', *Proceedings of the International Conference on Service Sciences (ICSS)*, IEEE, pp.324–328.
- Zhang, M., Liu, X., Zhang, R. and Sun, H. (2012) 'A web service recommendation approach based on QoS prediction using fuzzy clustering', *Proceedings of the International Conference on Services Computing (SCC)*, IEEE, pp.138–145.
- Zheng, Z., Zhang, Y. and Lyu, M.R. (2014) 'Investigating QoS of real-world web services', *IEEE Transactions on Services Computing*, Vol. 7, pp.32–39.
- Zhou, Q., Wu, H., Yue, K. and Hsu, C-H. (2019) 'Spatio-temporal context-aware collaborative QoS prediction', *Future Generation Computer Systems*, Elsevier, Vol. 100, pp.46–57.