LEARNING MACHINES FOR RECOMMENDER SYSTEMS

Zeshan Aslam Khan

68-FET/PHDEE/S14

Submitted in partial fulfillment of the requirements for the PhD degree in Electronic Engineering at the Department of Electrical Engineering

Faculty of Engineering and Technology

International Islamic University,

Islamabad

Supervisor

Dr. Syed Zubair

January, 2020

[illegible]

PhD
519.92
KHL

Learning machines.

E. Commerce

[illegible]

Recommender system

DEDICATED TO

My Teachers,

Parents,

Brothers,

Sister,

Friends,

Wife,

and Kids

CERTIFICATE OF APPROVAL

Title of Thesis: Learning Machines for Recommender Systems

Name of Student: Zeshan Aslam Khan

Registration No: 68-FET/PHDEE/S14

Accepted by the Department of Electrical Engineering, Faculty of Engineering and Technology, International Islamic University, Islamabad, in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electronic Engineering.

Viva voce committee:

Dr. Suheel Abdullah Malik (Chairman)
Associate Professor
Department of Electrical Engineering
International Islamic University, Islamabad.

[illegible]

Dr. Ihsan Ul Haq (Internal Examiner)
Associate Professor
Department of Electrical Engineering
International Islamic University, Islamabad.

[illegible]

Dr. Muhammad Usman (External Examiner - I)
Director, AERO Wah Cantt.

[illegible]

Dr. M. M. Talha (External Examiner - II)
Principle Scientist
KRL, Islamabad.

[illegible]

Prof. Dr. Syed Zubair (Supervisor)
Director, Research and Development
University of Sialkot, Sialkot

[illegible]

Dr. Muhammad Amir (Dean)
Professor
Department of Electrical Engineering
International Islamic University, Islamabad.

[illegible]

January 28, 2020

ABSTRACT

The demand for recommender systems in E-commerce industry has increased tremendously. Efficient recommender systems are being proposed by different E-business companies with the intention to give users accurate and most relevant recommendation of products from huge amount of information. To improve the performance of recommender systems, various stochastic variants of gradient descent based algorithms have been reported in the literature. It has been observed that matrix factorization (MF) technique based on stochastic gradient descent (SGD) algorithm has been widely used by recommender systems for providing accurate and fast recommendations of products to users. The performance of MF-based SGD methods for recommender systems is improved further through computing paradigms which are designed by utilizing the concept of fractional order gradient in addition to the integer order gradient in standard SGD methods. It has been noticed that the fractional version of MF-based SGD methods outperformed the standard counterparts. Therefore, by exploiting the strong mathematical concepts of fractional calculus, we have developed fractional calculus based three SGD strategies for fast and efficient matrix factorization of algorithms for recommender systems such as fractional stochastic gradient descent (F-SGD), momentum fractional stochastic gradient (mF-SGD) and normalized fractional stochastic gradient descent (NF-SGD). The performance in terms of estimated accuracy and convergence of standard SGD is improved in F-SGD. The scalability requirement of recommender systems in terms of rapidly increasing size of users and items, needs algorithms with fast convergence for providing efficient and relevant recommendations. The convergence of F-SGD is further accelerated through scalable F-SGD, termed as mF-SGD. In mF-SGD a proportion of previous gradients information is utilized through the addition of a momentum term to F-SGD. Both F-SGD and mF-SGD outperform matrix factorization based standard SGD counterparts but suffer from time varying appropriate selection of learning rate parameter. Therefore, a nonlinear computing paradigm (NF-SGD) based on normalized version of F-SGD is developed to investigate the adaptive behavior of learning rate for smooth and fast convergence with novel application to recommender systems. Performance of the proposed fractional SGD methods is verified through root mean square error (RMSE) against SGD baselines for Movie-Lens datasets. It is seen that proposed fractional adaptive methods achieve substantial percentage increase in performance for higher values of fractional order as compared to baselines.

Apart from improving the performance of recommender systems with regard to fast recommendations of items to users using MF-based algorithms, another challenge for E-commerce industry is to promote online businesses and sales by accurately predicting the list of those items that best match customers' tastes known as Top-N recommendations. Deep learning based auto-encoder models have further improved the performance of recommender systems by predicting Top-N recommendations. Motivated by the performance of deep auto-encoders, we propose a novel users rating-trend based collaborative denoising auto-encoder (UT-CDAE) which helps to predict improved Top-N recommendations. The suggested UT-CDAE determines user-item correlations by evaluating the rating-trend (High or Low) of a user towards set of items. The correctness of the suggested method is verified through different ranking evaluation metrics i.e., (mean reciprocal rank, mean average precision and normalized discounted gain). Experiments on standard Movie-Lens datasets show that UT-CDAE has gained higher average improvements in performance for chosen ranking-based evaluation metrics, with small proportions of noise, over state-of-the-art denoising auto-encoder models.

LIST OF PUBLICATIONS AND SUBMISSIONS

[1]. **Z. A. Khan,** N. I. Chaudhary and S. Zubair, "Fractional stochastic gradient descent for recommender systems," *Electronic Markets.*, vol. 29, pp. 275-285, 2019.

(IF 3.553)

[2]. **Z. A. Khan**, S. Zubair, K. Imran, R. Ahmad, S. A. Butt and N. I. Chaudhary, "A New Users Rating-trend based Collaborative Denoising Auto-Encoder for Top-N Recommender Systems" *IEEE Access.*, vol. 7, pp. 141287-141310, 2019. **(IF 4.098)**

[3]. **Z. A. Khan**, N. I. Chaudhary, S Zubair, M. A. Z. Raja, F. A. Khan and N. Dedovic, " Design of Normalized Fractional SGD Computing Paradigm for Recommender Systems," *Neural Computing and Applications*, pp. 1-18, Oct. 2019. [Online]. Available: doi: 10.1007/s00521-019-04562-6. **(IF 4.664)**

[4]. **Z. A. Khan**, S. Zubair, H. Alquhayz, M. Azeem and A. Ditta, " Design of Momentum Fractional Stochastic Gradient Descent for Recommender Systems " *IEEE Access.*, vol. 7, pp. 179575-179590, 2019. **(IF 4.098)**

[5]. N. I. Chaudhary, **Z. A. Khan,** S Zubair, M. A. Z. Raja and N. Dedovic, "Normalized Fractional Adaptive methods for Nonlinear Control Autoregressive Systems," *Applied Mathematical Modelling*, vol. 66, pp. 457-471, 2019. **(IF 2.841)**

[6]. S. Zubair, N. I. Chaudhary, **Z. A. Khan,** and W. Wang, "Momentum fractional LMS for power signal parameter estimation," *Signal Processing*, vol. 142, pp. 441-449, 2018. **(IF 4.086)**

[7]. N. I. Chaudhary, M. Ahmed, **Z. A. Khan,** S Zubair, M. A. Z. Raja and N. Dedovic, "Design of normalized fractional adaptive algorithms for parameter estimation of control autoregressive autoregressive systems," *Applied Mathematical Modeling*, vol. 55, pp. 698-715, 2018. **(IF 2.841)**

[8]. S. Batool, **Z. A. Khan,** W. Kamal, G. Mushtaq, and M. A. Kamal, " In silico Screening for Identification of Novel Anti-malarial Inhibitors by Molecular Docking, Pharmacophore Modeling and Virtual Screening," *Medicinal Chemistry,* vol. 11(7), pp. 687-700, 2015 **(IF 2.53)**

[9]. **Z. A. Khan,** E. P. de Freitas, T. Larsson, and H. Abbas "A multi-agent model for fire detection in coal mines using wireless sensor networks," In 2013, *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications,* pp. 1754-1761. IEEE, 2013.

[10]. M. Muzammil, **Z. A. Khan**, M. O. Ullah and I. Ali, "Performance analysis of block matching motion estimation algorithms for HD videos with different search parameters," 2016 International Conference on Intelligent Systems Engineering (ICISE), Islamabad, 2016, pp. 306-311.

SUBMITTED PAPERS

[1]. S. Iqbal, O. Hasan, R. Hafiz and **Z. A. Khan**, "LPQ-SAM: A Low Power Quality Scalable Approximate Multiplier" *Journal of Circuits, Systems and Computers,* (Submitted) 2019.

The research work presented in this dissertation is based on the accepted publications 1 to 4.

ACKNOWLEDGEMENTS

In the name of Allah (Subhanahu Wa Ta'ala), Who is the most gracious and the merciful. I would like to thank Allah for giving me strength and patience to complete this research work. Peace and blessings of Allah be upon His last Prophet Muhammad (Sallulah-o-Alaihihe-Wassalam) and all his Sahaba (Razi-Allah-o-Anhu) who dedicated their lives for Dawah and spread of Knowledge.

I am truthfully thankful to my supervisor Dr. Syed Zubair whose encouragement, motivation, thoughts and determinations make it possible for me to complete my higher studies. He has been a role model for me and many others in teaching, research and other aspects of life. I would also like to acknowledge and recognize the efforts, support and guidance of Dr. Naveed Ishtiaq Chaudhary during my research work.

I offer heartfelt thanks to my colleagues Engr. Sharjeel Abid Butt, Engr. Rehan and Engr. Khizer Mehmood for their never-ending support during last few years. I would like to acknowledge the support of International Islamic University Islamabad, Pakistan for providing me full fee waiver during the PhD studies. I am thankful to administration at department, as well as university level for their kind support.

I am highly grateful to my father, mother, brothers and sister for their prayers, love, inspiration and support throughout my life. I am also very thankful to my wife for her tolerance, cooperation and reinforcement during every stage of my PhD degree. Finally, I am thankful to my kids, whose innocent gestures were source of stimulation for me.

(Zeshan Aslam Khan)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

RS	Recommender systems
CF	Collaborative filtering
CB	Content based filtering
MF	Matrix factorization
NN	Neural Networks
GD	Gradient Descent
SGD	Stochastic Gradient Descent
F-SGD	Fractional stochastic gradient descent
mF-SGD	Momentum fractional stochastic gradient descent
NF-SGD	Normalized fractional stochastic gradient descent
LMS	Least Mean Square
DAE	Denoising auto-encoder
CDAE	Collaborative denoising auto-encoder
UT-CDAE	Users rating-trend based collaborative denoising auto-encoder
RMSE	Root Mean Square Error
MAP	Mean Average Precision
NDCG	Normalized cumulative discounted gain
MRR	Mean Reciprocal Rank
ICR	Input Corruption Ratio
HTUN	High-Trend-User-Node
LTUN	Low-Trend-User-Node

LIST OF SYMBOLS

A list of commonly used symbols in this dissertation are given below.

$\mathcal{L}$	Loss function
G	Cost function
R	Rating matrix
f_r	Fractional order
α	Proportion of previous gradients
μ	Step size
μ	Step size related to first order gradient
μ_{f_r}	Step size related to fractional order gradient
k	Number of features
$\mathbf{w}$	Weight vector
$\boldsymbol{v}$	Weight vector holding previous gradients
$\boldsymbol{v}_{hut}$	Weight vector for high trend user node
$\boldsymbol{v}_{lut}$	Weight vector for low trend user node
ϵ	Masking noise
λ_1	Regularization Rate for low rating-trend
λ_2	Regularization Rate for high rating-trend

Chapter 1. Introduction

1.1 Introduction

In this chapter the need, importance and applications of recommender systems are briefly discussed. In addition to advantages and limitations of recommender systems, standard methods employed by recommender systems are also covered. The vital role of model-based collaborative filtering method for designing innovative, robust, adaptive, reliable and convergent algorithms with the aim of rating and ranking prediction for recommender systems is recognized as well. Moreover, the introduction of collaborative filtering based proposed matrix factorization methods for rating prediction and novel deep neural network method for Top-N recommendations is also presented.

1.2 Background and Motivation

Nowadays, the need of e-commerce industry has increased rapidly and captured the businesses' interests in a short time span. At present, people are becoming habitual of using e-commerce applications and e-systems as part of digital technology. Variety of available products pose challenges for businesses to fulfill users' diverse demands. E-systems provide ease in users' taste management and also allow users to explore a variety of options before taking a decision for a specific product. However, it is somehow difficult to get the useful data (information) of products for millions of users from enormous amount of available and emerging data. To solve this problem, automated recommender systems are used by e-businesses with the intension to give users a precise and relevant recommendation of products.

Recommender systems (RS) are programs and procedures giving useful suggestions to users according to their preference for different products [1][2]. An important feature of recommender systems is to predict user's interest by analyzing the transactional behavior of a particular user to give useful recommendations [3]. Recommender system plays a significant role for the customers as well as for service providers. For customers, it is used to find interesting items and products, locate appropriate news content [4], discover new products and to explore new options matched with their interests. On the other hand, for service providers, recommender systems are used to promote their products, develop customer trust, obtain more knowledge about customers and enhance sales. The applications of recommender systems can be categorized into many categories such as e-commerce/e-shopping, e-government, e-group activities, e-tourism, e-library, e-learning, e-resource services and e-business [5]. Recommender systems have also been commonly used in entertainment (e.g. music and movie recommendations), content (e.g. recommendations for documents, news and e-applications), e-commerce (e.g. recommendation for items to buy such as camera and books) and services (e.g. travel and houses for rent service recommendations) [3][5].

Recommender systems employ different methods to provide recommendations. Frequently used recommendation methods are collaborative filtering (CF), content based filtering (CB), demographic, knowledge-based, community-based and hybrid recommender systems. Widely applied techniques amongst those are CF [6]–[12] and CB [13][14]. In CB filtering approach, a system learns to recommend the same products to users that the user preferred in the past [15]. While, in case of CF, recommendation of items for the specific users are based on those items which are mutually liked by other users. CF is also referred to as "people-to-people correlation" [16]. CF has scalability, sparseness and cold start issues whereas CB may provide overspecialized recommendations [6]. We will deeply elaborate CF-based modelling of the learning machines for recommender systems in succeeding Sub-Section

1.2.1 Collaborative Filtering in Learning Machines for Recommender Systems

To predict preferences of users on set of items, CF considers users' past activities such as viewing/purchasing history or users' rating patterns for items. In a CF based system, an active user for example, U_5 being the main participant pursues either for a rating prediction or looks for ranking of items. A CF based recommender system produces recommendations to user U_5 by using preference history of like-minded users as a reference for finding associations among compatible users. Such CF system entirely depends upon the taste of like-minded users. Usually a CF system includes a set of m users $U = \{U_1, U_2, \ldots, U_m\}$ and n items $I = \{I_1, I_2, \ldots, I_n\}$. The CF system builds a user-item rating matrix with dimension $(m \times n)$. The rating matrix contains preferences of users for items, where the preference opted by user U_i for item I_j is represented by $r_{i,j}$. To provide a useful recommendation to user U_5 for target item I_7, a CF based method either determines the appropriate rating for item I_7 or offers most suitable Top-N items list for user U_5. The graphical interpretation of CF procedure for recommender systems is as shown in Figure 1.1.

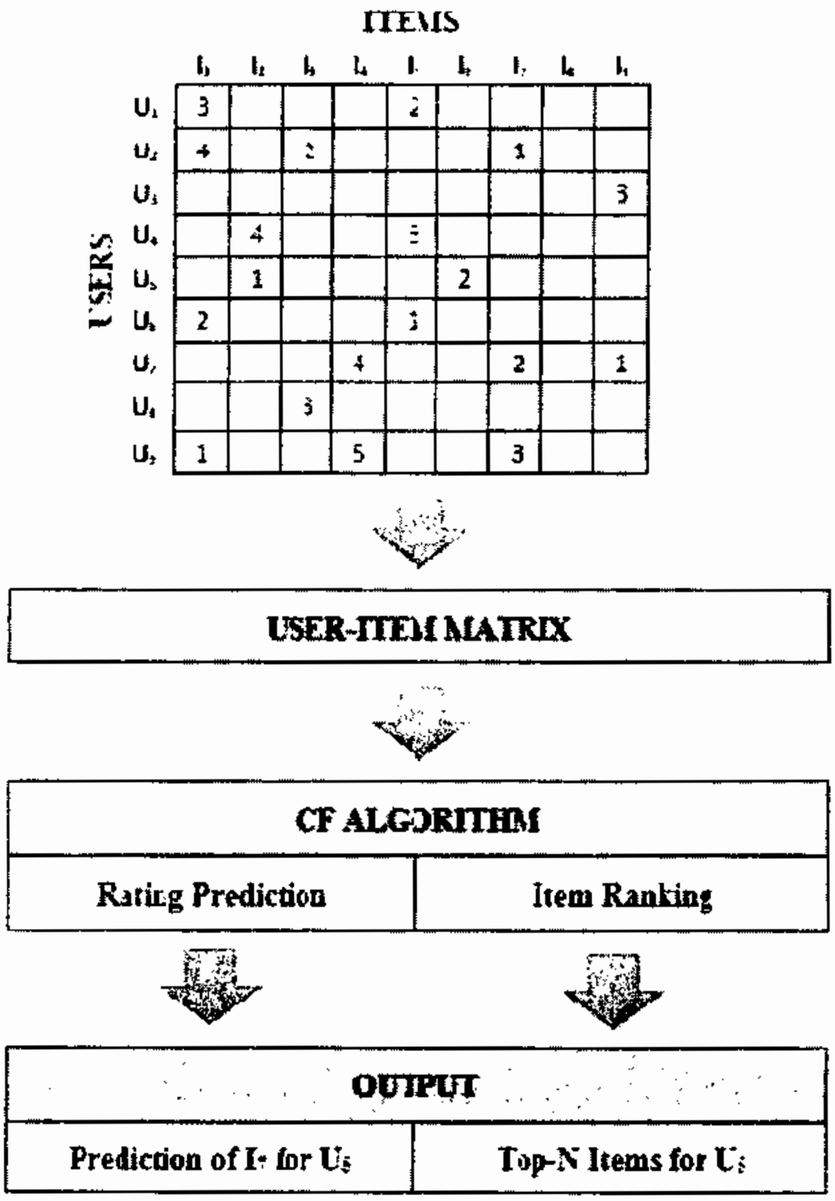


Fig. 1.1 Graphical view of Collaborative Filtering

Two types of methods are mainly used in CF for generating appropriate and precise recommendation, such as memory-based CF [15][17] and model-based CF [18][19]. Memory-based methods are also known as neighborhood-based and model-based methods which are termed as latent factor-based methods. Memory-based methods predict user-item interactions (ratings) using neighborhoods (item-item and user-user) information whereas, model-based methods give item recommendations by building a user ratings model. Model construction in latent factor based methods is performed by different data mining and machine learning methods such as genetic algorithms [20], memetic algorithm [21], deep neural networks (DNN) [22], Bayesian classifiers [23], latent factors [24] and matrix factorization (MF) [25]. The performance of model-based CF methods for rating prediction particularly matrix factorization [19][26] and ranking of Top-N items specially deep neural networks [27][28], motivated us to explore research opportunities and design efficient learning machines for enhancing the performance of MF and DNN-based adaptive algorithms for recommender systems. Therefore, for solving recommender systems problem through CF, we propose unique matrix factorization based learning machines for rating prediction and a novel deep neural network based learning machines for ranking prediction of items. Moreover, one of the important reasons for conducting the research using model-based CF methods, is the flexibility of model-based CF for designing innovative, robust, adaptive, reliable and convergent learning machines for recommender systems. The overall graphical illustration of CF approach for model-based proposed algorithms is presented in Fig. 1.2.

1.3 Research Problem/Problem Formulation

Several model-based CF methods (learning machines) are proposed to improve the performance of recommender systems in terms of speed, correctness, robustness, adaptability and reliability. The examples of such model-based learning machines include incremental and scalable CF methods using matrix factorization (MF) for recommender systems [25][26] and deep neural network (DNN) based CF models using auto-encoders [27][28]. Due to the increases in size of users, products and data, the e-commerce industry needs more accurate, adaptive and effective algorithms to fulfill users' demand by efficiently recommending related products to users. Therefore, in our research, the goal is to explore model-based CF machines by using the strong concepts of fractional calculus for effective MF and exploiting DNN-based new adaptive method for improving the performance of recommender systems. Following sub-sections includes our research problem in the context of MF and DNN-based CF.

1.3.1 Matrix Factorization based Collaborative Filtering

Matrix factorization based CF handles high volume of user/item data even when items are not rated frequently by the users [29]. To improve the convergence rate and

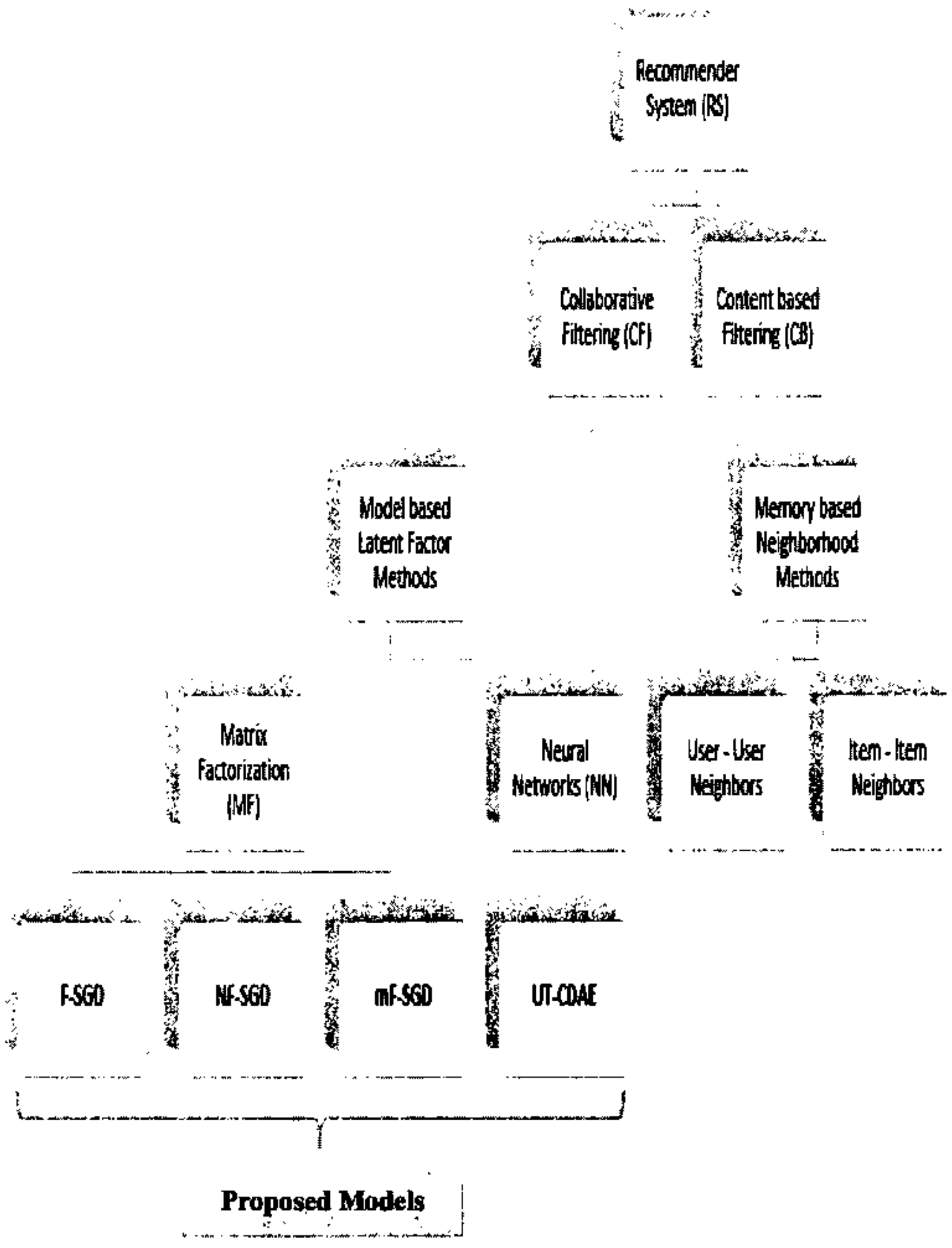

Recommender System (RS)
Collaborative Filtering (CF)
Content based Filtering (CB)
Model based Latent Factor Methods
Memory based Neighborhood Methods
Matrix Factorization (MF)
Neural Networks (NN)
User - User Neighbors
Item - Item Neighbors
F-SGD
NF-SGD
mF-SGD
UT-CDAE
Proposed Models

Fig. 1.2 Graphical representation of model-based Collaborative Filtering methods

correctness of model-based CF, several stochastic gradient descent based matrix factorization models [26][30] are suggested for recommender systems.

Recently, fractional calculus based SGD adaptive algorithms [31] have become popular because of the strong mathematical foundations of fractional calculus and their wide applications in diverse engineering domains such as [32]–[40]. Fractional adaptive methods outperformed in these fields than their standard counterparts. The designing of fractional gradient based SGD methods under model-based CF seems to be an encouraging field to discover and apply in the area of recommender systems for providing fast and accurate recommendations. Therefore, to enhance convergence rate and accuracy of recommender systems, our research work aims to investigate, design and use fractional calculus based SGD methods for efficient matrix factorization of learning machines for recommender systems.

1.3.2 Deep Learning based Collaborative Filtering

At present, matrix factorization based CF models are already in production to provide good recommendations, yet they are based on dot product of latent factors learned through matrix factorization which limit them in capturing subtle interactions of users and items. Deep learning based methods have further pushed the boundary of model-based CF for recommender systems research by either acting as a provider of latent features to conventional collaborative filtering methods [41][42] or fully substituting matrix factorization based methods [27], [28], [43]. Although deep learning neural models have been implemented for model-based CF by simply applying them on recommender systems' data, they still lack the modelling of users' preference behavior/trend while learning users-items interactions. For example, many items are given more preference by the users by assigning higher rating values while some items receive low ratings. This preference behavior establishes rating trend of different users for various items and this trend needs to be modelled while learning user-item interactions. In our research, we also investigate the behavior of a new DNN-based learning machine for modelling the users' preference trend by giving weights to different rating-trends of users for top-N recommendations.

1.4 Research Objectives

The objectives of our research are centered in two dimensions.

1. One of our objectives is to develop fractional calculus based nonlinear learning machines for model-based CF to exploit properties of stochastic gradient descent algorithm with novel applications to solve recommender system problem effectively and efficiently. We achieved our objective by developing the following fractional calculus based adaptive learning models for model-based CF.
   a) Fractional stochastic gradient descent (F-SGD).
   b) Momentum fractional stochastic gradient descent (mF-SGD).
   c) Normalized fractional stochastic gradient descent algorithm (NF-SGD).
2. Our another objective is to develop DNN-based a novel learning machine through model-based denoising auto-encoder for top-N recommender systems, that models users' rating-trend for giving trend based top-N recommendation of items to users. We achieved our goal by designing the following DNN-based auto-encoder.
   a) Fractional stochastic gradient descent (F-SGD).

1.5 Research Hypothesis

The research hypotheses formulated for the study are:

- The inclusion of fractional calculus notions in standard SGD method for matrix factorization procedure, may provide better estimated accuracy and fast convergence speed of related recommendations to users.
- To control the convergence of SGD-based MF methods, fractional adaptive algorithms may offer more control parameters than other standard algorithms
- The development of a ranking prediction based nonlinear computing model which is built on users' rating behavior for observed set of items, may provide flexibility in terms of regularization and incorporates the features of other well established techniques for top-N recommendations.
- Rating-trend based setting in proposed deep auto-encodes-based model may succeed in exploiting the latent representations required to predict the rating behavior of a user towards items, which authenticates the robustness and correctness of the model.

1.6 Thesis Outlines

The arrangement of the work presented in this thesis is as follows.

Chapter 1 presents a conceptual outline of the whole thesis, consisting of background and motivation for problem identification and defining Research Problem along with research gaps, statement and definition, clearly defining the Research Objectives and hypothesis. Further it presents a diagrammatic view of the methodology.

Chapter 2 gives a critical Literature Review explaining background, contemporary research related to our work supported by reports and articles in terms of techniques and procedures.

Chapter 3 elaborates the research methodology with matrix factorization technique by discussing three proposed algorithms using MF and their pseudocodes for showing results for a given set of parameters.

Chapter 4 includes the three algorithms of F-SGD, mF-SGD, NF-SGD with pseudocode given in Chapter 3, which are tested for performance on ML-100k and ML-1M datasets under three Case Studies for example Case-Study-I, Case-Study-II, and Case-Study-III

Chapter 5 explains the research methodology with deep learning scheme by discussing a suggested algorithm using deep learning and its pseudocode for presenting results for a given set of parameters. The proposed algorithm is also tested for performance with standard counterparts on ML-100k and ML-1M datasets.

Chapter 6 concludes the thesis by highlighting the outcomes of our research and how much of the research objectives are achieved by comparing the results with those of contemporary research works, justifying marked contribution. Further, it suggests directions on how the research may be continued in the related field. It also lists down the publications we have made out of work.

Chapter 2. **Critical Literature Review**

2.1 Introduction

This chapter is divided into two main halves. First half of the chapter includes the basic concept of matrix factorization especially for recommender systems, literature survey of matrix factorization techniques along with their applications to different fields including recommender systems. In second half of the chapter, we present deep learning based auto-encoders for recommender systems, literature survey of auto-encoders for Top-N recommender systems followed by summary of the chapter at the end of chapter.

2.2 Matrix Factorization for Recommender Systems

The e-commerce industry has widely captured the interest of businesses nowadays. The businesses are getting complicated and multidimensional due to the drastic increase in the variety of products and demands of the users. For the selection of the right product for the right user, recommender systems are incorporated in different industrial applications to provide most relevant recommendation of items to the users [2], [44]–[48]. One of the main methods for solving recommender system problem using CF is matrix factorization, which finds latent factors to relate users with the items of their interest. Matrix factorization being latent factor based method uses known ratings given by users for set of items to acquire an approximate model. This approach represents user – item connections by factors expressing the hidden (latent) features of users and items for a particular system that is priority-wise standing of users and category-wise standing of items. Weight of factors depends upon the measure of the liking rated by specific user for a particular item. For the realization of MF as latent factor based model, there are two important problems faced by MF techniques for the implementation of recommender systems: (i) rating matrix is partially filled by users for several items (sparse matrix), (ii) size of the data grows exponentially (scalability). These issues are well addressed by matrix factorization (MF) [25][29]. MF is the most useful technique for dealing millions of users with billions of ratings. The MF of a given rating matrix is presented graphically in Fig. 2.1.

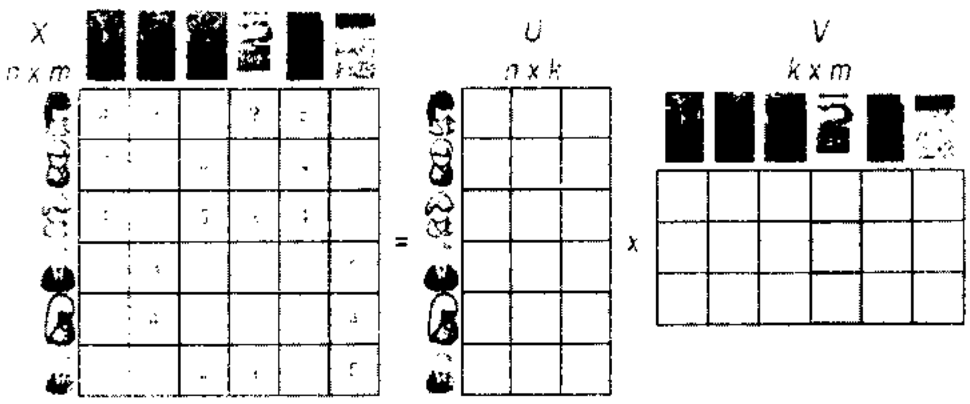


Fig .2.1 Block diagram of basic Matrix factorization for recommender systems

Using MF, numerous methods have been effectively applied for solving RS problems. Some of the techniques are maximum margin matrix factorization [49], probabilistic matrix factorization [18], alternating least squares [50], probabilistic latent semantic analysis [51], singular value decomposition [52] and expectation maximization for MF [53].

A regularized version of matrix factorization has been implemented by Simon Funk using gradient descent learning algorithm [54]. Other novel and efficient algorithms like biased matrix factorization and asymmetric factor models have been presented by Paterek in [30]. An incremental training technique has been suggested in [25], which includes updating trained parameters based on the new data. To deal with large scale datasets and to factorize large scale matrices, different algorithms have been proposed. Alternating least squares (ALS) and stochastic gradient descent (SGD) algorithms gained much attention during recommender system competitions and are extensively used for matrix factorization [55][56]. To speed up the process of matrix factorization, momentum SGD [26] and ALS based coordinate descent methods [57] were proposed.

In literature, different variants of gradient descent (GD) and stochastic gradient descent (SGD) have been suggested with the aim of increasing performance in terms of accuracy and convergence speed. In both methods, parameters are updated in an iterative manner to minimize objective function. In GD, a specific iteration involves running through all the samples in training set for a single update of a parameter, while in SGD, a single or a subset of samples from the training set is taken for parameter update. This makes GD highly computationally complex for large number of training samples. Thus the suitable choice for many recommender systems involving large number of training samples is SGD. It has been observed that SGD is simpler and efficient than ALS [58], yet SGD cannot be easily parallelized to deal large scale data sets [58]. For the parallelization of SGD, some computationally efficient techniques have been suggested such as fast parallel SGD (FSGD) [59] and distributed SGD (DSGD) [60].

Furthermore, to achieve improved performance in terms of estimated accuracy, adaptability and speed of standard SGD-based MF methods for example [25][26], we have proposed three MF based algorithms by (1) exploiting strong mathematical foundations of fractional calculus as already proved in [37][39][61][62] and (2) applying those fractional concepts in standard SGD-based MF methods . The standard SGD-based adaptive methods for recommender systems presented in this section use integer (first) order gradient during parameters update mechanism. While the suggested fractional order adaptive schemes use fractional order gradient along with integer order gradient in their optimization procedure. The proposed F-SGD method exploits power of fractional order in addition to first order gradients to provide fast and accurate recommendations but lacks automatic step-size tuning for achieving improved performance. Automatic adjustment of step-size is introduced in the proposed NF-SGD method, which considerably improves the performance in terms of convergence speed and accuracy. Finally, mF-SGD is designed to increase the convergence rate by using the proportion of previous gradients (integer order and fractional order). In addition to convergence speed the momentum term in mF-SGD also helps to avoid trapping in local minima.

2.3 Auto-Encoders for Recommender Systems

Auto-encoder is a feedforward unsupervised DNN architecture, which is trained to encode input into latent representations (encoding) so that the input is reconstructed back using those representations (decoding) [63]. Auto-encoder also provides relevant predictions to users [64]–[66]. Few variants of auto-encoder are proposed to provide ratings as well as ranking based predictions for top-N recommender systems [65][67]. The auto-encoders solving recommender systems include marginalized auto-encoder, contractive auto-encoder, sparse auto-encoder, denoising auto-encoder and variational auto-encoder [65], [67]–[69].

A simple auto-encoder [27] for recommender system consists of input layer, a hidden layer and an output layer for encoding a high dimensional partially observed input vector $\boldsymbol{t} \in \mathbb{R}^{d}$ (user based or item based) of a rating matrix $R \in \mathbb{R}^{p\times q}$ to a low dimensional space (K), which is reconstructed at the output layer (also called decoder part). where $\boldsymbol{b}_{\mathcal{H}} \in \mathbb{R}^{K}$, is bias of the hidden layer and matrix $\mathbf{W}_1 \in \mathbb{R}^{d\times K}$ represents the weights of input layer associated with the hidden layer. Similarly,$\boldsymbol{b}_{O} \in \mathbb{R}^{d}$, is bias of the output layer and matrix $\mathbf{W}_2 \in \mathbb{R}^{K\times d}$ denotes the weights tied to the hidden layer and the output layer. The network design of a basic auto-encoder is given in Fig. 2.2.

Parameters $\Phi = \{\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_{O}\}$ of an auto-encoder are learned (trained) through back-propagation to solve the reconstruction loss. The loss can be squared-loss (for regression) or cross-entropy-loss (for binary inputs). For user input vectors the squared objection function$(\mathcal{L})$ for auto-encoder is given as:

$$\min_{\Phi} \frac{1}{p} \sum_{u=1}^{p} \mathcal{L}(\boldsymbol{t}, \hat{\boldsymbol{t}}) + \Re\,(\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_{O}) \tag{2.1}$$

$$\mathcal{L}(\boldsymbol{t}, \hat{\boldsymbol{t}}) = \|\boldsymbol{t}_u - \hat{\boldsymbol{t}}_u\|_2^2 \tag{2.2}$$

Here, $\mathcal{L}$ is the squared loss function and $\Re$ represents the regularization term. Where $\Re$ contains the squared ℓ_2 norm of parameters given as:

$$\Re\,(\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{v}_u, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_{O}) = \frac{\lambda}{2}(\|\mathbf{W}_1\|_2^2 + \|\mathbf{W}_2\|_2^2 + \|\boldsymbol{b}_{\mathcal{H}}\|_2^2 + \|\boldsymbol{b}_{O}\|_2^2) \tag{2.3}$$

The projection of the input $\boldsymbol{t}$ to the hidden space $\mathcal{H}$ through activation function is represented as:

$$\mathcal{H}(\boldsymbol{t}) = f(\mathbf{W}_1^{\mathrm{T}}\boldsymbol{t} + \boldsymbol{b}_{\mathcal{H}}) \tag{2.4}$$

The activation function $f(\cdot)$ can be Sigmoid or Identity as given in [27]. If we use Sigmoid as mapping function $f(\cdot) = \mathit{Sigmoid}$, then hidden layer is represented as:

$$\mathcal{H}(\boldsymbol{t}) = Sigmoid(\mathbf{W}_1^{\mathrm{T}}\boldsymbol{t} + \boldsymbol{b}_{\mathcal{H}}) = \sigma(\mathbf{W}_1^{\mathrm{T}}\boldsymbol{t} + \boldsymbol{b}_{\mathcal{H}}) \tag{2.5}$$

Reconstruction of the input is done in the output layer (O) by again projecting the hidden representation to the output layer and applying the activation function $g(\cdot) = Sigmoid$ at the output layer. The reconstructed input vector is given as:

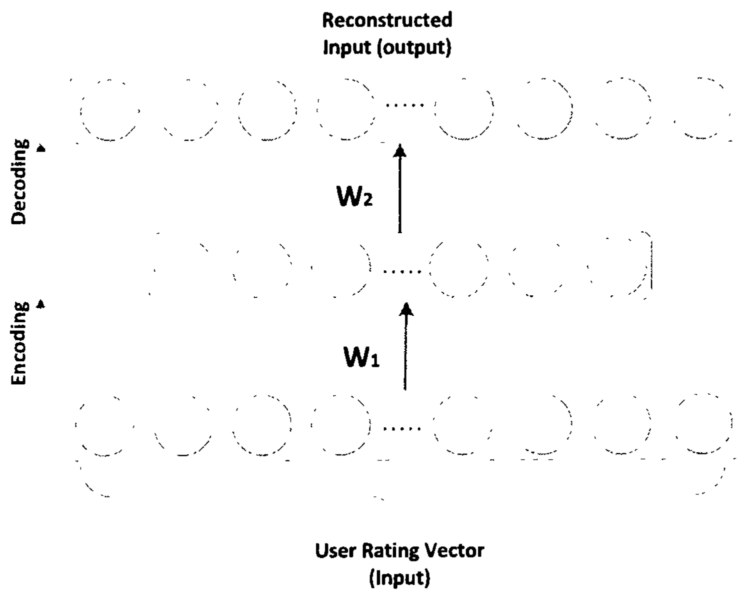


Fig. 2.2 Network diagram of a basic Auto-encoder recommender systems

$$\hat{\boldsymbol{t}} = g(\mathbf{W}_2^{\mathrm{T}}\mathcal{H}(\boldsymbol{t}) + \boldsymbol{b}_O) \tag{2.6}$$

$$\hat{\boldsymbol{t}} = Sigmoid(\mathbf{W}_2^{\mathrm{T}}\mathcal{H}(\boldsymbol{t}) + \boldsymbol{b}_O) = \sigma(\mathbf{W}_2^{\mathrm{T}}\mathcal{H}(\boldsymbol{t}) + \boldsymbol{b}_O) \tag{2.7}$$

The considerable contributions of auto-encoders for recommender systems include, generating predictions at the reconstruction layer and extraction of latent characteristics at the bottle-neck layer through dimensionality reduction procedure [65][66]. Recommender systems use auto-encoders to deal specifically with scalability and sparsity concerns.

The accuracy of recommendations provided by recommender systems through auto-encoders is assessed by two procedures i.e. rating prediction, frequently evaluated in terms of root mean square error (RMSE), and ranking, mostly computed with regard to normalized discounted gain (NDCG), precision (P) and recall (R), including few other metrics [70]. Rating prediction deals merely with observed ratings, whereas ranking deals with ratings for all items in the rating matrix whether specified or missing. Ranking is a helpful and useful method when the goal is to recommend each user a short list of N items often termed as Top-N recommendations.

Top-N Recommendations Task

Recently, the demand of items similar to the users' taste has increased radically. The challenge now is to accurately and efficiently recognize the list of products that best match customers' tastes. Top-N recommender systems have been extensively studied and widely used in E-commerce industry during the past several years. The job of the recommender system is said to be a Top-N recommendation task, when the aim of recommender system is to find a size-N ranked lists of particular items which are the most pleasing to the user [71][28]. The Top-N recommendation task has also been expressed as a ranking problem. Inspired by the work done in CDAE [28], we have proposed DNN based a novel method in this thesis, which is termed as users' rating-trend based collaborative denoising auto-encoder (UT-CDAE) for providing Top-N recommendations. UT-CDAE offers accurate and efficient Top-N recommendations to users as compared to few standard deep learning methods for Top-N recommender systems.

In the literature few methods for rating prediction and ranking have been proposed for recommender systems. Auto-rec is a single hidden layer ratings prediction based method proposed in [27], which has two variants i.e. item-based (I-Auto-rec) and user-based Auto-rec (U-Auto-rec). The work in [27] concludes that the performance of auto-encoders (I-Auto-rec) for recommender systems is effected by using different combination of mapping functions at the hidden and output layers. Moreover, it shows that deep layered I-Auto-rec reflects slight improvement in performance than shallow I-Auto-rec. The extension of Auto-rec is termed as Collaborative Filtering Neural Network (CFN) [72] which exploits de-noising approach and incorporate side information at the input layer. The de-noising method helps in learning more robust hidden representations while side information helps to alleviate the cold start and sparsity problems. Another extended variant of CFN [73] includes the side information not only at input layer but includes it to all layers. Apart from improving the training time, inclusion of side information also improves robustness and predictive correctness of the model.

One of the ranking based denoising versions of auto-encoder for top-N recommendations is Collaborative Denoising Auto-Encoders (CDAE) [28]. In CDAE, an additional user oriented node for each user is introduced at the input layer of denoising auto-encoder. That extra node at the input layer is connected to the hidden layer with unique weights (weight vector) for all user preference vectors. The performance of the model is significantly affected using these unique weights. In CDAE a negative sampling technique is introduced to reduce time complexity without sacrificing the ranking eminence [65][28].

The type of variational auto-encoder (Multi-VAE and Multi-DAE) with implicit feedback for recommender systems is specified in [73]. In this scheme, parameters are computed using Bayesian inference method. Such inference technique improves the performance of (Multi-VAE and Multi-DAE) over CDAE [28]. Another form of collaborative filtering using auto-encoders is Auto-encoder based Collaborative

Filtering (ACF) [74]. Instead of utilizing the sparse input vectors as given in preference matrix, ACF decomposes the original sparse input vectors based on the range of integer ratings e.g. 1-5. The decomposition corresponds to increase the sparseness of input vectors by decreasing the predictive accuracy. Moreover, ACF does not succeed to consider non-integer ratings as in [73][28].

Auto-encoders are also responsible for learning latent feature representations from user/item content characteristics for recommender systems [65]. In this context, a Bayesian based collaborative deep learning (CDL) [75] model is proposed to combine stacked denoising (SDAE) auto-encoder with probabilistic matrix factorization (PMF). Another model similar to that of CDL was proposed before CDL known as relational stacked denoising auto-encoders for tag recommendation (RSDAE) [76]. The main difference between CDL and RSDAE is that instead of integrating SDAE with PMF, RSDAE combines SDAE with relational information matrix. Collaborative variational auto-encoder (CVAE) [77] is another CDL extension which introduces variational auto-encoder to replace deep neural part of CDL. CVAE learns probabilistic hidden representations for data contents [65]. For top-N recommendations, collaborative deep ranking (CDR) [78] which is a new model-based on pair-wise objective function is proposed. Previous works in [65][78][79] have shown that pair-wise models are better fit for producing ranking lists.

2.3 Our Work

In this thesis, we have used fractional calculus based concepts in standard SGD-based MF methods and developed three fractional order based learning machines for recommender systems to achieve fast and effective matrix factorization for recommending appropriate recommendations of items to users. Moreover, to enhance and modify the idea of CDAE [28] for top-N recommendations, we proposed an innovative deep learning based learning machine for recommender systems to explore and apply users' rating-trend for collaborative filtering for the purpose of learning more robust and non-linear representations in the hidden layer of an auto-encoder for producing ranking based predictions.

2.4 Summary

The basic concepts and the need of matrix factorization based algorithms including various MF based learning machines proposed for solving recommender systems have been discussed in this chapter. Furthermore, the requirements and perceptions behind the learning machines comprising of different variants of deep auto-encoders for rating prediction of unobserved ratings as well as ranking prediction for providing Top-N recommendations of items to users are also presented.

In the next Chapter, proposed methodologies with matrix factorization procedure for recommender systems are comprehensively discussed.

Chapter 3.

Novel Learning Machines for Matrix Factorization of Recommender Systems

This chapter contains fractional calculus based novel learning machines with matrix factorization procedure for solving the recommender systems problem.

3.1 Introduction

The update rules of standard SGD based MF methods discussed in Chapter 2 are achieved by taking integer order gradient of the objective function only. The integer order gradient based SGD methods can be further improved by taking fractional order gradient using the concepts of fractional calculus as has been observed in different areas of research [80][38][81].

Fractional calculus deals with integrals and derivatives that are of non-integer order. Fractional order calculus and integer order calculus fields have been established long ago but the applicability of fractional order calculus has not been explored [82][83] as much as of the integer order calculus in different fields except Mathematics. Fractional calculus has been experimented by researchers in several fields of engineering and technology. For example, neural networks [84]–[86], controllers and oscillators [87]–[89], signal processing [90][91], system identification [92][93], circuit analysis [94][95], motion estimation [96], image denoising [97][98][33], chaotic systems [99], predictive maintenance [100], biomedical science [35], fractional control [32], economic systems [34], and differential equations [101][102].

3.2 Fractional order gradient-based methods

Different fractional order-based adaptive methods have been proposed in the field of signal processing and fractional order control. By applying concepts of fractional order derivative [103], modified versions of least mean square (LMS) algorithms are presented. The solution to different communication, control and signal processing problems such as speech enhancement [104], active noise control [40], parameter estimation [105][37], nonlinear system identification [106][107] and power signal modeling [38][61] have been devised by using modified fractional adaptive schemes. Fractional adaptive algorithms perform significantly better than standard adaptive strategies for demonstrated applications. These fractional calculus-based methods considerably improve the performance of adaptive algorithms. Therefore, the development of fractional order based adaptive algorithms (learning machines) for recommender systems appears to be a promising research space that needs to be explored for giving accurate recommendations to users.

In fractional adaptive algorithms, selection of fractional order is a bit tricky and important as well. The range of fractional order used in the proposed fractional order based algorithms in this thesis is $\in (0,1)$. To choose an appropriate fractional order, suggested methods are evaluated for different values of fractional orders between 0 and 1. It is observed in fractional adaptive algorithms that rate of convergence and steady state error increases for higher values of fractional order. The behavior of fractional order for convergence speed and accuracy has been explored further in [62] using the range of fractional order $\in (1,1.5)$. Similar behavior of fractional order in terms of accuracy and convergence speed is noticed by authors for standard LMS and its variants in [61][38].

In this thesis, we design different learning machines which are variants of basic SGD to further enhance the convergence speed and accuracy for providing accurate recommendations. The two proposed fractional order-based SGD methods given in this chapter are published and one method is submitted in different journals.

The learning machines proposed for solving recommender systems problem are:

1. Fractional stochastic gradient descent (F-SGD).
2. Momentum fractional stochastic gradient descent (mF-SGD).
3. Normalized fractional stochastic gradient descent algorithm (NF-SGD).

3.3 System Model: Recommender System

The objective function of recommender system for suggested fractional order based SGD learning machines using matrix factorization is as follows.

Let $\boldsymbol{Z} \in R^{p \times q}$ be a partially filled input rating matrix for recommender systems holding p users and q items. The objective function for resolving recommender system problem through matrix factorization is

$$G(\boldsymbol{a},\boldsymbol{b}) = \min_{\substack{A \in R^{k \times p} \\ B \in R^{k \times q}}} \sum_{(u,i) \in \Omega} (\boldsymbol{Z}_{ui} - \boldsymbol{a}_u^T \boldsymbol{b}_i)^2 = \min_{\substack{A \in R^{k \times p} \\ B \in R^{k \times q}}} \sum_{(u,i) \in \Omega} E_{ui}^2 \tag{3.1}$$

Where the error between observed and estimated rating is $E_{ui} = (\boldsymbol{Z}_{ui} - \boldsymbol{a}_u^T \boldsymbol{b}_i)$, $\boldsymbol{a}$ and $\boldsymbol{b}$ are the u^{th} user and i^{th} item column vectors of user features matrix A ($A \in R^{k \times p}$) and the item feature matrix $\boldsymbol{B}$ ($\boldsymbol{B} \in R^{k \times q}$) respectively, k denotes number of features for both users and items and Ω represents the specific indices for given ratings.

The goal of objective function (3.1) is to discover factorized matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ from sparse matrix $\boldsymbol{Z}$. The missing rating entries of $\boldsymbol{Z}$ are generated through the dot product of $\boldsymbol{A}$ and $\boldsymbol{B}$ that is $\boldsymbol{AB}^T$. Factors of $\boldsymbol{a}_u$ for a specific user u, define the amount of interest of a user for a variety of features of an item whereas, factors of $\boldsymbol{b}_i$ for a particular item i hold the features for that specific item. Liking of user u for item i is represented by $\boldsymbol{Z}_{ui}$.

The objective of the proposed methods is to approximate the interest of the user by reconstructing the rating matrix and recommending those items to the users which have not been rated by the users previously. Those missing entries by the users make the rating matrix sparse. The dataset with ratings forms input rating matrix $\boldsymbol{Z} \in R^{p \times q}$ which is very sparse in its composition and is decomposed by the proposed algorithms into user $\boldsymbol{A}$ ($\boldsymbol{A} \in R^{k \times p}$) and item $\boldsymbol{B}$ ($\boldsymbol{B} \in R^{k \times q}$) factor matrices. The resulting factor matrices provide latent factors against features for users and items learned by the proposed algorithms alternatively using their update equations. The suggested algorithms are evaluated by reconstructing the rating matrix $\widehat{\boldsymbol{Z}}$ with the help of the learned latent factors $\boldsymbol{AB}^T$ and finding the root mean square error (RMSE) of the original data matrix with the constructed ones. The reconstructed data matrix is dense and it also provides the recommendation values for the items that have not been rated in the original data matrix.

In Sections below, proposed matrix factorization methods of F-SGD, mF-SGD and NF-SGD are used to find the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

3.4 Fractional SGD (F-SGD)

Motivated by the recent outcomes of fractional calculus based methods, fractional order stochastic gradient descent is applied to speed up standard SGD, we call it as F-SGD. Various adaptive algorithms including different variants of stochastic gradient descent (SGD) have been suggested to improve estimated accuracy and convergence speed but those methods have not succeeded in enhancing convergence speed and estimated accuracy as compared to our proposed (F-SGD) technique. The fractional order variation in standard SGD substantially improves the speed and accuracy for providing better recommendations. As the implementation of fractional calculus has not been exploited yet in the field of recommender systems, we explore F-SGD in RS and investigate its effect for increasing convergence.

3.4.1 Mathematical Formulation

This section describes the development of proposed fractional stochastic gradient descent (F-SGD) adaptive algorithm along with derivations with respect to integer order as well as fractional order gradient of the objective function.

The alternative and recursive weight update expressions of standard SGD for both user and item feature vectors for the n-th iteration are written as:

$$\boldsymbol{a}_u(n+1) = \boldsymbol{a}_u(n) - \frac{\mu}{2}\frac{\partial G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{a}_u} \tag{3.2}$$

$$\boldsymbol{b}_i(n+1) = \boldsymbol{b}_i(n) - \frac{\mu}{2}\frac{\partial G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{b}_i} \tag{3.3}$$

Where μ represents the learning rate parameter.

By taking the gradient of objective function (3.1) with respect to user feature vector $\boldsymbol{a}_u$, we get

$$\frac{\partial G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{a}_u} = -2E_{ui}\boldsymbol{b}_i \tag{3.4}$$

Likewise, by calculating the gradient of (3.1) with respect to item feature vector $\boldsymbol{b}_i$, we achieve

$$\frac{\partial G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{b}_i} = -2E_{ui}\boldsymbol{a}_u \tag{3.5}$$

At iteration n, the user and item feature weight update equations are evaluated by putting equations (3.4) and (3.5) in (3.2) and (3.3) respectively:

$$\boldsymbol{a}_u(n+1) = \boldsymbol{a}_u(n) + \mu E_{ui}\boldsymbol{b}_i(n) \tag{3.6}$$

$$\boldsymbol{b}_i(n+1) = \boldsymbol{b}_i(n) + \mu E_{ui}\boldsymbol{a}_u(n) \tag{3.7}$$

The above update equations are derived on the basis of integer order gradient and are standard SGD updates. The SGD updates can be extended to fractional calculus based SGD by incorporating fractional order gradient, in addition to the integer order gradients. This achieves better convergence rate and estimation accuracy as compared to the standard SGD.

In the FSGD method, user feature and item feature vectors are updated as:

$$\boldsymbol{a}_u(n+1) = \boldsymbol{a}_u(n) + \mu E_{ui}\boldsymbol{b}_i(n) - \frac{\mu_{f_r}}{2}\frac{\partial^{f_r} G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{a}_u^{f_r}} \tag{3.8}$$

$$\boldsymbol{b}_i(n+1) = \boldsymbol{b}_i(n) + \mu E_{ui}\boldsymbol{a}_u(n) - \frac{\mu_{f_r}}{2}\frac{\partial^{f_r} G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{b}_i^{f_r}} \tag{3.9}$$

Where μ and μ_{f_r} denote the integer and fractional order learning rate parameters of the F-SGD algorithm respectively and f_r is the fractional order such that $0 < f_r < 1$.

For a function $y(t) = t^m$ the fractional derivative with order f_r is declared generally as in [108]:

$$\mathcal{D}^{f_r} y(t) = \frac{\Gamma(m+1)}{\Gamma(m - f_r + 1)} t^{m-f_r} \tag{3.10}$$

Where, the fractional (f_r) order gradient is represented by $\mathcal{D}^{f_r}$ operator and Γ denotes a gamma function, represented as:

$$\Gamma(t) = (t-1)! \tag{3.11}$$

Assuming the fractional order gradient of a constant value to be zero. Calculating the fractional order gradient of (3.1) with respect to the user feature vector and item feature vector respectively and using (3.10) and (3.11), we obtain:

$$\frac{\partial^{f_r} G(\boldsymbol{a}, \boldsymbol{b})}{\partial \boldsymbol{a}_u^{f_r}} \cong -2\, E_{ui}\, \boldsymbol{b}_i\, \frac{1}{\Gamma(2-f_r)} \boldsymbol{a}_u^{1-f_r} \tag{3.12}$$

$$\frac{\partial^{f_r} G(\boldsymbol{a}, \boldsymbol{b})}{\partial \boldsymbol{b}_i^{f_r}} \cong -2\, E_{ui}\, \boldsymbol{a}_u\, \frac{1}{\Gamma(2-f_r)} \boldsymbol{b}_i^{1-f_r} \tag{3.13}$$

After applying expressions (3.12) and (3.13) in (3.8) and (3.9) we get the F-SGD weight update rules for user features and items features vectors as:

$$\boldsymbol{a}_u(n+1) = \boldsymbol{a}_u(n) + \mu E_{ui} \boldsymbol{b}_i(n) + \frac{\mu_{f_r}}{\Gamma(2-f_r)} E_{ui}\, \boldsymbol{b}_i(n) \\ \circledcirc\, |\boldsymbol{a}_u(n)|^{1-f_r} \tag{3.14}$$

$$\boldsymbol{b}_i(n+1) = \boldsymbol{b}_i(n) + \mu E_{ui} \boldsymbol{a}_u(n) + \frac{\mu_{f_r}}{\Gamma(2-f_r)} E_{ui}\, \boldsymbol{a}_u(n) \\ \circledcirc\, |\boldsymbol{b}_i(n)|^{1-f_r} \tag{3.15}$$

Where the sign $\circledcirc$ denotes element-wise multiplication of two vectors, to ignore complex entries, absolute value of the vectors is taken. Equations (3.14) and (3.15) represent the F-SGD update relations for user and item feature vectors, respectively. The pseudo code of the proposed FSGD algorithm for recommender systems, named as Algorithm 1, is as shown in Table 3.1.

Table 3.1: Pseudo code of the proposed F-SGD algorithm

```
Algorithm 1: Pseudo code of proposed F-SGD method for recommender systems
    Input : $\boldsymbol{Z} \in R^{p \times q}$ : Training set = Rating matrix, $\mu$ : Learning rate, $\mu_{fr}$ :
Fractional learning rate, Epochs, $k$: Number of features and $fr$: Fractional order
    Output: $\hat{\boldsymbol{Z}} \in R^{p \times q}$ : updated rating matrix, $\boldsymbol{A} \in R^{k \times p}$, $\boldsymbol{B} \in R^{k \times q}$ : factor user
and item matrices respectively
1) Partition $\boldsymbol{Z}$ into two sets: $Train \in R^{p \times q}$ and $Test \in R^{p \times q}$
2) Initialize $A \in R^{k \times p}$ and $\boldsymbol{B} \in R^{k \times q}$ randomly (between 0 and 1).
3) Find indices for non-zero entries of $Train$
4) Loop until the terminal condition is reached or desired epochs:
5)     Iterate over users and items indices $(u, i)$ for non-zero entries of $Train$
6)         Compute $E_{ui} = (\boldsymbol{Z}_{ui} - \boldsymbol{a}_u^T \boldsymbol{b}_i)$
7)         Update $\boldsymbol{a}_u$, the $u^{th}$ column vector of $A$ according to Eq. (3.14);
8)         Update $\boldsymbol{b}_i$, the $i^{th}$ column of $\boldsymbol{B}$ according to Eq. (3.15);
9)     Reconstruct rating matrix using updated $A$ and $\boldsymbol{B}$, $\hat{\boldsymbol{Z}} = \boldsymbol{AB}^T$
10)    Calculate the RMSE on $Test$
11)    Check terminal condition
12) End
```

3.5 Momentum F-SGD (mF-SGD)

Based on this recent development [61] and the suggested learning machine in the preceding Section 3.4 , we propose a new efficient fractional SGD by adding a momentum term (mF-SGD) to the standard F-SGD update equation. This variation in the F-SGD offers improvement in the estimation accuracy and convergence behavior of the recommender systems. The weights update procedure for mF-SGD includes percentage of previously calculated gradients in the weight update relation, which improves the convergence speed of mF-SGD relative to standard F-SGD and mSGD for same learning rate parameters. We show that the proposed momentum F-SGD has higher convergence speed as compared to its counter-parts and achieves required estimation accuracy for less number of iterations as compared to the mSGD and F-SGD.

3.5.1 Mathematical Formulation

The F-SGD algorithm presented in Section 3.4 can be extended to a faster converging algorithm by introducing a momentum term in the update equation of the F-SGD. We call this proposed algorithm as momentum F-SGD. The gradient calculation is incorporated using notions of momentum term as it has already been applied for the standard LMS calculation [109]. The momentum term exhibits the percentage of prior gradients instead of merely the current gradients, which is added to the existing weights. Accumulated proportions of preceding gradients help in making convergence and optimal search process faster and avoid trapping in local minima.

The weight update expression for the proposed momentum F-SGD is given as:

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) - \boldsymbol{v}(n+1) \tag{3.16}$$

Where $\hat{\boldsymbol{w}}$ are the weights that need to be updated and the term $\boldsymbol{v}(n+1)$ is known as velocity term which holds the earlier gradients. The velocity vector is equal in dimension to the weight vector and can be calculated as:

$$\boldsymbol{v}(n+1) = \alpha \boldsymbol{v}(n) + \mu\, \boldsymbol{g}(n) \tag{3.17}$$

Where the range of α lying between 0 and 1, determines the percentage of previous gradients used for the current update of expression, μ is learning rate such that $\mu = \mu_{fr}$.

$\boldsymbol{g(n)}$ shows the gradient (integer order and fractional order gradient) part of the expression at current iteration and is given as:

$$\boldsymbol{g}(n) = \frac{\partial G(\widehat{\boldsymbol{w}})}{\partial \widehat{\boldsymbol{w}}} + \frac{\partial^{f_r} G(\widehat{\boldsymbol{w}})}{\partial \widehat{\boldsymbol{w}}^{f_r}} \tag{3.18}$$

Using Equations (3.16), (3.17) and (3.18), the updated weight update equations for proposed momentum FSGD (mF-SGD) for user and item vectors are:

$$\boldsymbol{a}_u(n+1) = \boldsymbol{a}_u(n) - \boldsymbol{v}_1(n+1) \tag{3.19}$$

$$\boldsymbol{b}_i(n+1) = \boldsymbol{b}_i(n) - \boldsymbol{v}_2(n+1) \tag{3.20}$$

Where $\boldsymbol{v}_1(n+1)$ and $\boldsymbol{v}_2(n+1)$ are velocity terms holding previous gradient proportions of $\boldsymbol{a}_u$ and $\boldsymbol{b}_i$ respectively:

$$\boldsymbol{v}_1(n+1) = \alpha\boldsymbol{v}_1(n) + \mu\boldsymbol{g}_1(n) \tag{3.21}$$

$$\boldsymbol{v}_2(n+1) = \alpha\boldsymbol{v}_2(n) + \mu\boldsymbol{g}_2(n) \tag{3.22}$$

While $\boldsymbol{v}_1(0) = \boldsymbol{v}_2(0) = 0$ and $\boldsymbol{g}_1(n)$ denote the gradient (integer order and fractional order) of G *w.r.t* $\boldsymbol{a}_u$ and $\boldsymbol{g}_2(n)$ represent gradient of G *w.r.t* $\boldsymbol{b}_i$ given in (3.4), (3.12) and (3.5), (3.13) respectively.

$$\boldsymbol{g}_1(n) = \frac{\partial G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{a}_u} + \frac{\partial^{f_r} G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{a}_u^{f_r}},$$

$$\boldsymbol{g}_1(n) = E_{ui}\boldsymbol{b}_i(n) + \frac{1}{\Gamma(2-f_r)} E_{ui}\,\boldsymbol{b}_i(n) \circledcirc |\boldsymbol{a}_u(n)|^{1-f_r} \tag{3.23}$$

Similarly, $\boldsymbol{g}_2(n)$ is computed as:

$$g_2(n) = \frac{\partial G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{b}_i} + \frac{\partial^{f_r} G(\boldsymbol{a},\boldsymbol{b})}{\partial \boldsymbol{b}_i^{f_r}},$$

$$g_2(n) = E_{ui}\boldsymbol{a}_u(n) + \frac{1}{\Gamma(2-f_r)} E_{ui}\, \boldsymbol{a}_u(n) \circledcirc |\boldsymbol{b}_i(n)|^{1-f_r} \tag{3.24}$$

The pseudo code of the proposed mF-SGD algorithm for recommender systems, termed as Algorithm 2, is as shown in Table 3.2. **In case of standard momentum SGD algorithm, fractional gradient is not used. Thus, only integer gradient terms in (3.23) and (3.24) are considered.**

Table 3.2: Pseudo code of the proposed mF-SGD algorithm

```
Algorithm 2: Pseudo code of proposed mF-SGD method for recommender systems

    Input : $\boldsymbol{Z} \in R^{p\times q}$ : Training set = Rating matrix, $\mu$ : Learning rate, $\mu_{fr}$ :
Fractional learning rate, Epochs, $k$: Number of features, $fr$: Fractional order and $\alpha$:
Proportion of previous gradients

    Output: $\hat{\boldsymbol{Z}} \in R^{p\times q}$ : updated rating matrix, $\boldsymbol{A} \in R^{k\times p}$, $\boldsymbol{B} \in R^{k\times q}$ : factor user
and item matrices respectively

  1) Partition $\boldsymbol{Z}$ into two sets: $Train \in R^{p\times q}$ and $Test \in R^{p\times q}$
  2) Initialize $\boldsymbol{A} \in R^{k\times p}$ and $\boldsymbol{B} \in R^{k\times q}$ randomly (between 0 and 1).
  3) Find indices for non-zero entries of $Train$
  4) Loop until the terminal condition is reached or desired epochs:
  5)     Iterate over users and items indices $(u, i)$ for non-zero entries of $Train$
  6)         Compute $E_{ui} = (\boldsymbol{Z}_{ui} - \boldsymbol{a}_u^T\, \boldsymbol{b}_i)$
  7)         Update $\boldsymbol{a}_u$, the $u^{th}$ column vector of $\boldsymbol{A}$ according to Eq. (3.19);
  8)         Update $\boldsymbol{b}_i$, the $i^{th}$ column of $\boldsymbol{B}$ according to Eq. (3.20);
  9)     Reconstruct rating matrix using updated $\boldsymbol{A}$ and $\boldsymbol{B}$, $\hat{\boldsymbol{Z}} = \boldsymbol{A}\boldsymbol{B}^T$
  10)    Calculate the RMSE on $Test$
  11)    Check terminal condition
  12) End
```

3.6 Normalized F-SGD (NF-SGD)

A normalized version of F-SGD with time varying learning rate is proposed in this section, termed as NF-SGD. The proposed method adaptively tunes the learning rate and provides fast recommendations according to the taste of the users.

3.6.1 Mathematical Formulation

In this Section, matrix factorization based proposed adaptive technique for solving recommender systems problem given in Section 3.3 is discussed in terms of its update rules. The derivations of update rules with respect to user factor vectors and item factor vectors are also given for suggested NF-SGD method.

To explore the adaptive nature of the learning rate in standard SGD and to improve convergence rate further, weight update relations of standard SGD presented in equations (3.6) and (3.7) are divided with the norm of input vectors (user factor vector and item factor vectors) and this method is called as normalized stochastic gradient descent algorithm (NSGD). The weight update relations for standard NSGD technique are given as follows:

$$\boldsymbol{a}_u(n+1) = \boldsymbol{a}_u(n) + \frac{\mu}{\|\boldsymbol{b}_i(n)\|^2}\left[E_{ui}\boldsymbol{b}_i(n)\right], \tag{3.25}$$

$$\boldsymbol{b}_i(n+1) = \boldsymbol{b}_i(n) + \frac{\mu}{\|\boldsymbol{a}_u(n)\|^2}\left[E_{ui}\boldsymbol{a}_u(n)\right]. \tag{3.26}$$

In equation (3.25), learning rate is divided with the magnitude of the item factor vector while in (3.26), learning rate is divided with the magnitude of the user factor vector.

Similarly, to increase the convergence rate of proposed F-SGD further by adjusting the learning rate adaptively, weight update relations of F-SGD given in equations (3.14) and (3.15) are divided with the norm of input vectors (user factor vector and item factor vectors) as shown in the update rules of NSGD using equations (3.25) and (3.26). This suggested method is known as normalized fractional stochastic gradient descent algorithm i.e., NF-SGD. The pseudo code for the suggested NF-SGD method, called Algorithm 3, is as shown in Table 3.3.

Table 3.3: Pseudo code of the proposed NF-SGD algorithm

```
Algorithm 3: Pseudo code of suggested NF-SGD algorithm for recommender systems

    Input : $\boldsymbol{Z} \in R^{p\times q}$ : Training set = Rating matrix, $\mu$ : Learning rate, $\mu_{fr}$ :
Fractional learning rate, Epochs, $k$: Number of features and $fr$: Fractional order

    Output: $\widehat{\boldsymbol{Z}} \in R^{p\times q}$ : updated rating matrix, $\boldsymbol{A} \in R^{k\times p}$, $\boldsymbol{B} \in R^{k\times q}$ : factor user
and item matrices respectively

    1)  Partition $\boldsymbol{Z}$ into two sets: $Train \in R^{p\times q}$ and $Test \in R^{p\times q}$
    2)  Initialize $\boldsymbol{A} \in R^{k\times p}$ and $\boldsymbol{B} \in R^{k\times q}$ randomly (between 0 and 1).
    3)  Find indices for non-zero entries of $Train$
    4)  Loop until the terminal condition is reached or desired epochs:
    5)      Iterate over users and items indices $(u, i)$ for non-zero entries of $Train$
    6)          Compute $E_{ui} = (\boldsymbol{Z}_{ui} - \boldsymbol{a}_u^T \boldsymbol{b}_i)$
    7)          Update $\boldsymbol{a}_u$, the $u^{th}$ column vector of $\boldsymbol{A}$ according to Eq. (3.27);
    8)          Update $\boldsymbol{b}_i$, the $i^{th}$ column of $\boldsymbol{B}$ according to Eq. (3.28);
    9)      Reconstruct rating matrix using updated $\boldsymbol{A}$ and $\boldsymbol{B}$, $\widehat{\boldsymbol{Z}} = \boldsymbol{A}\boldsymbol{B}^T$
    10)     Calculate the RMSE on $Test$
    11)     Check terminal condition
    12) End
```

The weight update relations for the proposed NF-SGD method are given as follows:

$$\boldsymbol{a}_u(n+1) = \boldsymbol{a}_u(n) + \frac{\mu}{\|\boldsymbol{b}_i(n)\|^2}\,[E_{ui}\boldsymbol{b}_i(n) + \frac{1}{\Gamma(2-f_r)}\,E_{ui}\,\boldsymbol{b}_i(n) \circledcirc |\boldsymbol{a}_u(n)|^{1-f_r}\,] \,, \tag{3.27}$$

$$\boldsymbol{b}_i(n+1) = \boldsymbol{b}_i(n) + \frac{\mu}{\|\boldsymbol{a}_u(n)\|^2}\,[\,E_{ui}\,\boldsymbol{a}_u(n) + \frac{1}{\Gamma(2-f_r)}\,E_{ui}\,\boldsymbol{a}_u(n) \circledcirc |\boldsymbol{b}_i(n)|^{1-f_r}\,] \,. \tag{3.28}$$

In equation (3.27) learning rate is divided by the magnitude of the item factor vector, while, learning rate in equation (3.28) is divided by the magnitude of the user factor vector.

3.7 Summary

Initially, the importance of fractional calculus is discussed in this chapter then related work done for the development of fractional order based strategies along with applications has been presented. Moreover, mathematical details of SGD-based different variations of fractional adaptive algorithms of Fractional SGD, momentum fractional SGD and normalized fractional SGD have been presented for solving recommender systems problem using matrix factorization.

The succeeding Chapter presents simulations and analytical results of three proposed algorithms as compared to baseline methods for different datasets under three Case Studies.

Chapter 4. **Simulation and Analytical Results**

4.1 Introduction

Each section of chapter 4 represents a case study consisting of two subsections that is, simulation parameters, results and discussion. Simulation parameters subsection contains parameters used for simulation of both standard and proposed algorithms. Results and discussion subsection represents simulation results graphically and exemplifies the performance of the algorithms using convergence table.

Moreover, this chapter holds simulations by applying the standard and the proposed algorithms to recommender systems. The performance in terms of convergence of algorithms is verified using the ML-100k dataset [110] for solving recommender systems problem. ML-100k dataset comprises of 943 users and 1682 items while the movie ratings rated by the users are equal to 100k. At minimum, 20 movies are rated by each user. The demographic information of the users is also given in the dataset. The column density and row density of the dataset is calculated using the input rating matrix which is composed of ratings given by the users against movies. The row density of the dataset is 106.04, which is an average rating given by a single user for different items while, the column density is 59.45, which shows the average rating for a single item by different users. The range of the ratings varies from 1-5. The total ratings in the dataset divided by the product of total number of users and items gives the overall density of the dataset, which is 6.30%.

$$Overall\ Density\ of\ Dataset = Total\ ratings/(\,Total\ Users \times Total\ Items)$$

Based on the dataset, partially filled rating matrix of users and items is made, which is further factorized into users factor matrix and items factor matrix. The hidden factors present in user and item factor vectors of factorized matrices are computed with the help of user and item update relations of the given algorithms. The liking of the user for items is estimated using the methods by rebuilding the rating matrix for the purpose of recommending unrated items to the users. The unrated entries in the matrix lead to sparse matrix. The reconstruction of the rating matrix is achieved through the dot product of the factorized matrices. The RMSE is then computed using the given rating matrix and the estimated rating matrix. The rebuilt (estimated) matrix is a completely filled matrix reflecting the recommendations for the items (movies) unrated in the observed rating matrix. RMSE is given by the following relation:

$$RMSE_{test} = \sqrt{mean\left(\sum_{(p,q)\in\Omega_{\mathrm{test}}} e_{pq}^2\right)} \quad . \tag{4.1}$$

To demonstrate the convergence of algorithms in a broader spectrum, algorithms in case study III are also tested on a bigger dataset i.e. ML-1M (6040 x 3952) other than ML-100k (943 x 1682).

Performance of the algorithms on both ML-100k and ML-1M datasets is tested for different values of latent features (k) with different variations in the learning rate parameters. In case of the proposed fractional adaptive methods, learning rate parameters of μ and μ_{fr} are used. Since standard adaptive algorithms have μ as learning parameter and the proposed fractional order based methods have μ_{fr} as additional learning rate parameter, hence we check standard algorithms for different values of μ

and proposed fractional order-based methods for different variations of μ and μ_{fr}. Another additional parameter in suggested fractional order based methods which is not found in standard ones is the fractional order fr. This fractional order plays a significant role for suggested methods to behave differently as compared to standard counterparts, hence we also test our proposed algorithms for various values of the fractional order such that $0 < fr < 1$. Each algorithm in case studies I and II is run for 200 iterations, whereas algorithms in case study III are run for 100 iterations to estimate the data matrix through which final RMSE value is calculated.

Experiments for each case study are performed on a laptop with (Core-i3-4005U @ 1.70 GHz) Processor and 4.00 GB DDR2 RAM. Simulations are carried out in Spyder 2.3.8, release 2015 using Python 2.7.13 (64 bit) on Windows 10 Pro Education (64 bit) operating system.

4.2 Case-Study-I: Standard SGD and proposed FSGD algorithms

This case study compares the performance of standard SGD and suggested F-SGD methods in terms of convergence speed and estimated accuracy for recommender systems. In this case study, update rules of standard SGD applied for recommender systems are given in equations (3.6) and (3.7), whereas the update relations of proposed F-SGD are specified in equations (3.14) and (3.15). General flow of proposed FSGD algorithm for Case-Study-I is presented graphically in flow diagram as shown in Fig. 4.1. Simulation parameters and simulation results are discussed in the following two subsections.

4.2.1 Simulation Parameters

Our proposed FSGD algorithm is experimented in terms of RMSE with five values of learning rates μ and μ_{fr} i.e. $\mu = \mu_{fr}$ = 0.1, 0.05, 0.01, 0.005, 0.001 for different k = 10, 20, 30. Optimal learning rates μ and μ_{fr} for the FSGD are selected by going through number of trials for three different fractional orders i.e., fr = 0.25, 0.50, and 0.75 to achieve best RMSE value after convergence. The FSGD shows inconsistency in RMSE with $\mu = \mu_{fr}$ = 0.1, 0.05, 0.01 and 0.005, but it outperforms at $\mu = \mu_{fr}$= 0.001 for three variations of k after convergence. Hence we selected $\mu = \mu_{fr}$ = 0.001 for our simulations. The selection of fractional order in fractional adaptive strategies is also important. In order to select a suitable value of fractional order, the proposed strategy is assessed for three different fractional orders of 0.25, 0.50 and 0.75. It is witnessed that the proposed method (FSGD) is accurate for all fractional orders but relatively improved results are achieved for higher value of fractional order i.e., f_r = 0.75, in all cases of k. Therefore, it is reasonable to take f_r = 0.75 for the suggested method (FSGD). Simulation parameters are summarized in Table 4.1.

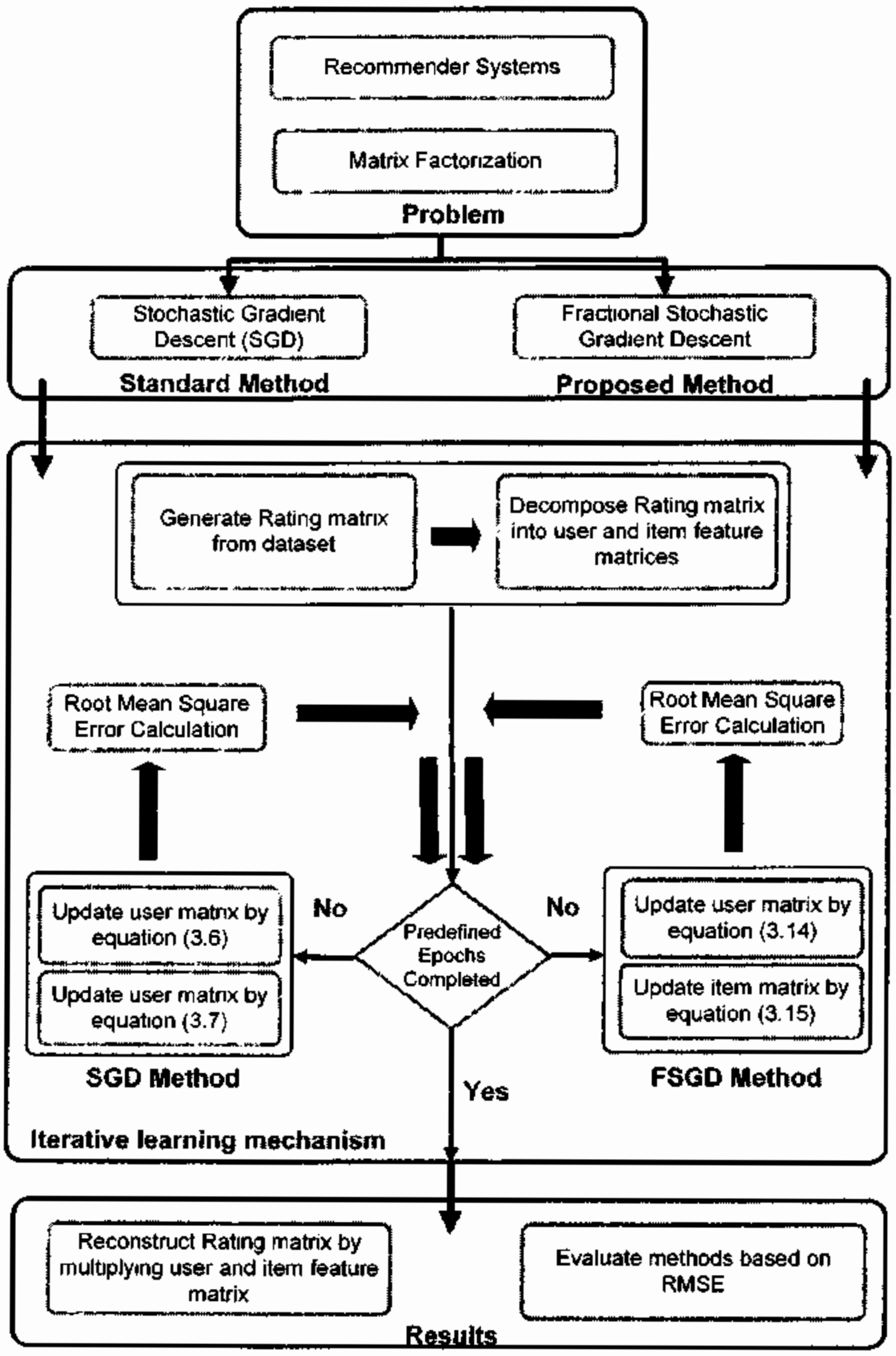

Recommender Systems
Matrix Factorization
Problem
Stochastic Gradient Descent (SGD)
Standard Method
Fractional Stochastic Gradient Descent
Proposed Method
Generate Rating matrix from dataset
Decompose Rating matrix into user and item feature matrices
Root Mean Square Error Calculation
Root Mean Square Error Calculation
Update user matrix by equation (3.6)
Update user matrix by equation (3.7)
No
Predefined Epochs Completed
No
Update user matrix by equation (3.14)
Update item matrix by equation (3.15)
SGD Method
Yes
FSGD Method
Iterative learning mechanism
Reconstruct Rating matrix by multiplying user and item feature matrix
Evaluate methods based on RMSE
Results

Fig. 4.1 General flow diagram of F-SGD for recommender systems

Table 4.1: Simulation parameters

Name	Symbol	Values	Usage
Learning rate	μ	0.001	Step-size for convergence
Fractional Learning rate	μ_{fr}	0.001	Fractional step-size for convergence
Fractional order	fr	0.25, 0.50, 0.75	A tuning parameter
Features	k	10, 20, 30	Represents user/item characteristics

4.2.2 Results and Discussion

For optimal value of learning rate $\mu = \mu_{fr} = 0.001$, RMSE of the FSGD is given in Table 4.2 for various fr and k values. Table 4.2 shows that the FSGD achieves best RMSE with $\mu = 0.001$ and $fr = 0.75$ for various k. It can be perceived from Table 4.2 that with the increase in values of fr against k variants, RMSE decreases gradually. For $fr = 1$, the FSGD becomes standard SGD. Table 4.2 also represents RMSE values for SGD with ($fr = 1$) against k latent features. Results in the Table 4.2 clearly show that FSGD outperforms SGD for given fr variations.

Graphical representation of convergence behavior of SGD and the FSGD in terms of RMSE and number of iterations is shown in Fig. 4.2, Fig. 4.3 and Fig. 4.4. Investigation of both algorithms is made using learning rate $\mu = \mu_{fr} = 0.001$ with three fractional orders (i.e., $fr = 0.25, 0.50, 0.75$) and different number of features (i.e., k = 10, 20 and 30).

Fig. 4.2 shows RMSE curves for SGD and FSGD for 200 iterations. It is observed in Fig. 4.2 that minimum RMSE = 0.770 achieved by SGD for $k = 10$ in 200 iterations is achieved by FSGD with ($\mu = 0.001$, $fr = 0.75$, $k = 10$) in quite less iterations. It also shows that the performance of the FSGD degrades with the decrease in the fractional order. However, Fig. 4.2 also depicts that even for a minimum fr value i.e., $(fr = 0.25)$ FSGD achieved minimum SGD RMSE in fairly fewer iterations.

Table 4.2: RMSE values for different parameters setting

Epochs	RMSE for $\mu = 0.001$											
	$k = 10$				$k = 20$				$k = 30$			
	SGD	FSGD	FSGD	FSGD	SGD	FSGD	FSGD	FSGD	SGD	FSGD	FSGD	FSGD
	—	fr 0.25	fr 0.50	fr 0.75	—	fr 0.25	fr 0.50	fr 0.75	—	fr 0.25	fr 0.50	fr 0.75
40	0.968	0.913	0.913	0.905	0.975	0.922	0.914	0.906	0.980	0.935	0.928	0.909
80	0.884	0.832	0.838	0.825	0.894	0.816	0.800	0.776	0.897	0.825	0.801	0.771
120	0.832	0.781	0.788	0.774	0.823	0.732	0.714	0.688	0.822	0.718	0.685	0.644
160	0.796	0.748	0.755	0.742	0.756	0.672	0.656	0.632	0.742	0.631	0.600	0.561
200	**0.770**	0.726	0.733	**0.720**	**0.703**	0.630	0.616	**0.596**	**0.672**	0.568	0.543	**0.507**

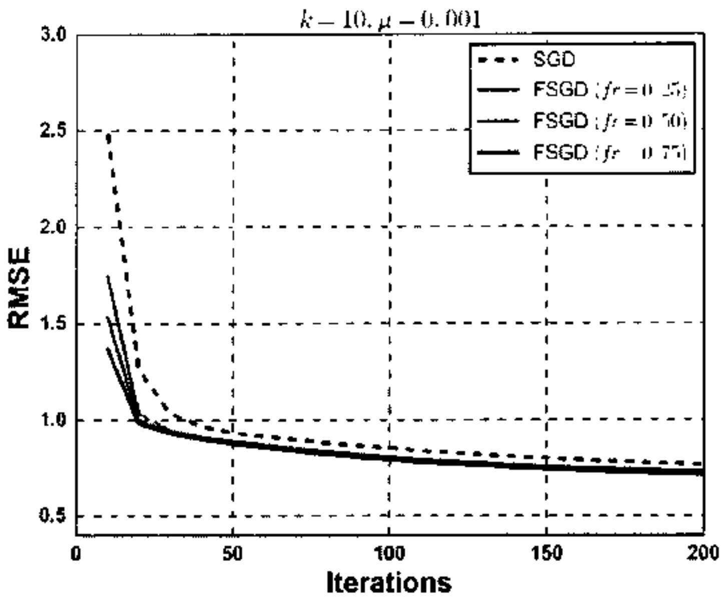


Fig. 4.2 RMSE of SGD vs FSGD for 200 Iterations against k features when $k = 10$

Similarly, Fig. 4.3 and Fig. 4.4 also depict the same behavior of FSGD as in Fig. 4.2. The only difference is the significant reduction in RMSE of FSGD with the increase in number of features for higher fr values.

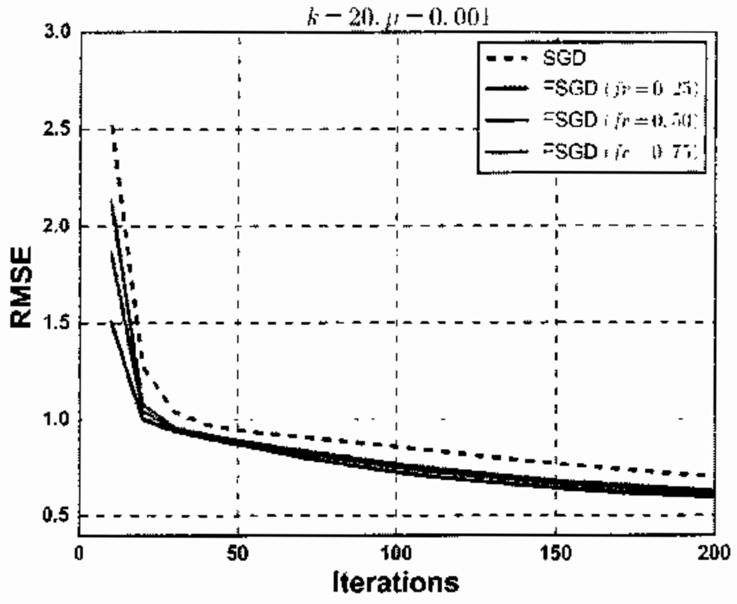


Fig. 4.3 RMSE of SGD vs FSGD for 200 Iterations against k features when $k = 20$

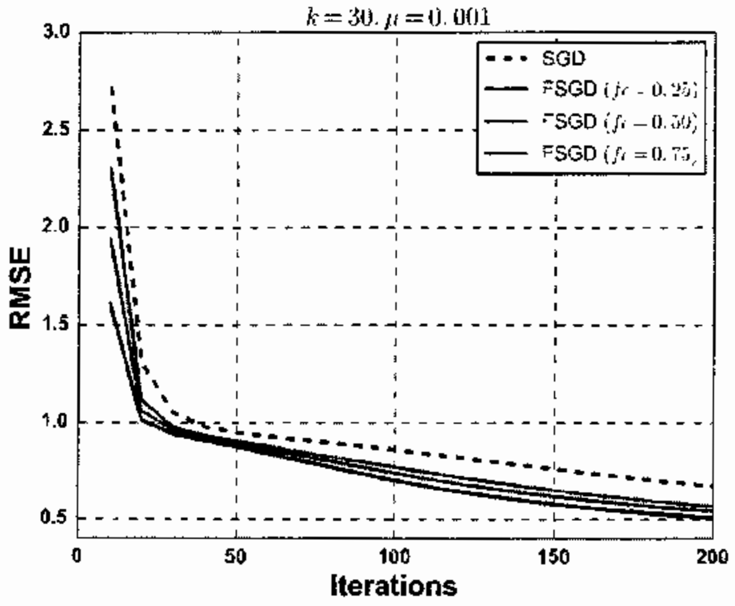
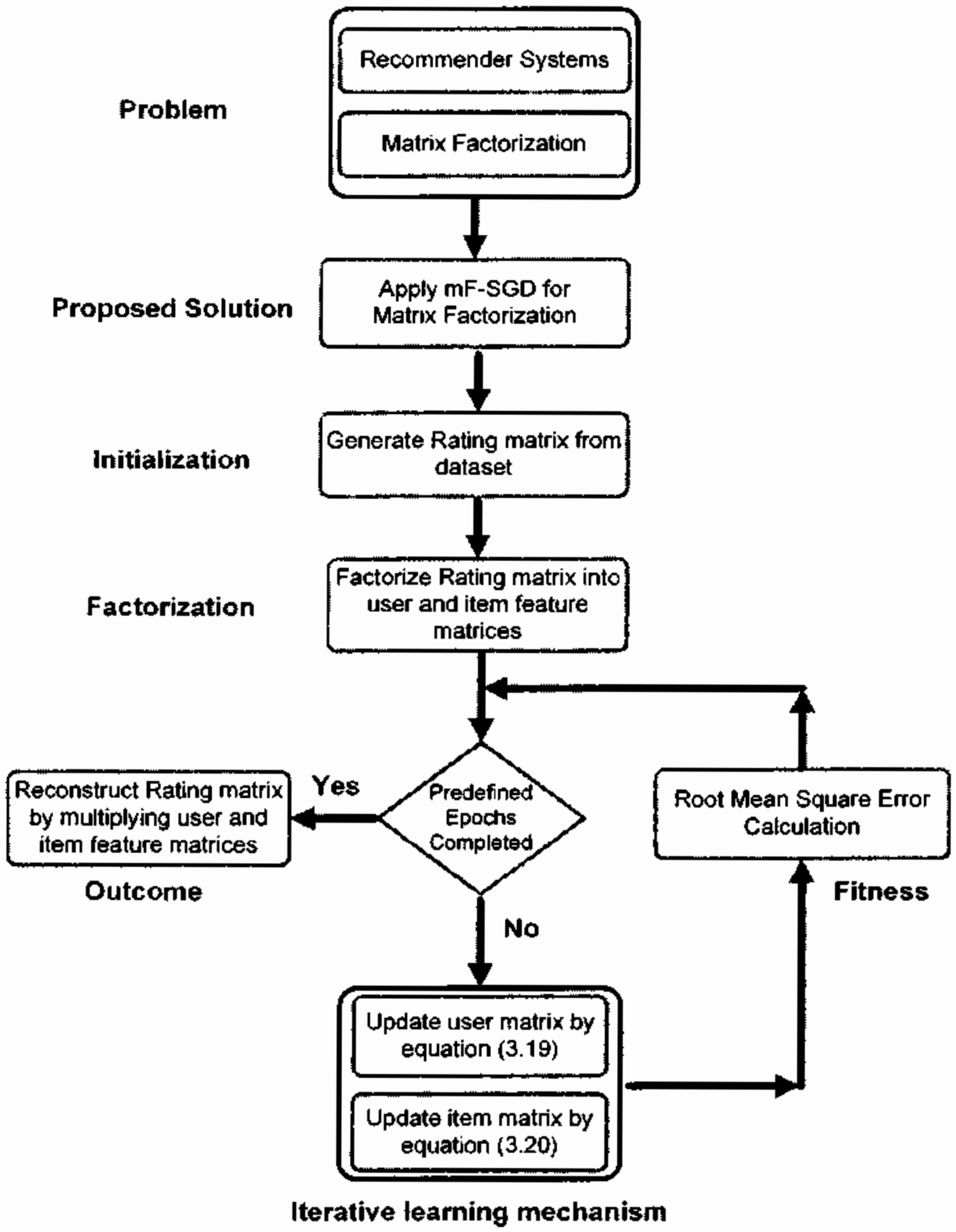


Fig. 4.4 RMSE of SGD vs FSGD for 200 Iterations against k features when $k = 30$

By analyzing the results, the superiority of the FSGD algorithm over SGD is seen clearly for all variants of fractional order. FSGD performs outstandingly better for k = 30 with all fr variations but the best performance of FSGD is achieved with fr = 0.75. It is also witnessed that FSGD achieves convergence quickly for k = 20 as compared to the SGD but the optimal convergence of FSGD is attained with fr = 0.75. The same behavior between FSGD and SGD is also observed for k = 10. Moreover, it is observed that by increasing number of features, RMSE is decreased considerably for both algorithms. It is also noted that for all variations of k, FSGD achieves the best RMSE values of SGD in relatively less number of iterations, which clearly demonstrates the improved behavior of the FSGD.

4.3 Case-Study-II: F-SGD, mSGD and proposed mF-SGD algorithms

In this case study fast and convergent behavior of suggested approach of mF-SGD for recommender systems is compared with two adaptive strategies of F-SGD and mSGD against different hyper-parameter values. The update rules used in the iterative update mechanism of F-SGD and mF-SGD are given in (3.14), (3.15) and (3.19), (3.20) respectively. Overall graphical flow of the suggested mF-SGD for Case-Study-II is given in Fig. 4.5. Optimal parameter values for simulation and the outcome of algorithms based on these optimal parameter values are discussed in the two subsequent subsections.



Problem
Recommender Systems
Matrix Factorization
Proposed Solution
Apply mF-SGD for Matrix Factorization
Initialization
Generate Rating matrix from dataset
Factorization
Factorize Rating matrix into user and item feature matrices
Reconstruct Rating matrix by multiplying user and item feature matrices
Yes
Predefined Epochs Completed
Root Mean Square Error Calculation
Outcome
Fitness
No
Update user matrix by equation (3.19)
Update item matrix by equation (3.20)
Iterative learning mechanism

Fig. 4.5 Flow chart of the suggested mF-SGD for recommender systems

4.3.1 Simulation Parameters

For fair comparison of the proposed method (mF-SGD) with F-SGD and mSGD, the parameters for each method are selected empirically after plotting error curves for 200 iterations and choosing the best parameter values in each case. The error curves (Figs. 4.6 - 4.13) are obtained by randomly splitting the rating matrix in training and testing sub-matrices and calculating RMSE for each iterations. Different parameters required for tuning of each method are given in Table 4.3.

Table 4.3: Tuning parameters for different algorithms

Name	**Symbol**	**Usage**	**Algorithms Using parameters**
Learning Rate	μ	Step size for integer order gradient	F-SGD, mF-SGD, mSGD
Fractional Learning rate	μ_{f_r}	Step size for fractional order gradient	F-SGD, mF-SGD
Fractional Order	f_r	Fractional order of the gradient	F-SGD, mF-SGD
Features	k	Size of user/item feature vector	F-SGD, mF-SGD, mSGD

Tuning of optimal learning rate (μ, μ_{f_r})

The learning rate μ value of 10^{-4} for the given algorithms is empirically chosen after performing a number of experiments for different set of 0.0001, 0.0005, 0.001, 0.005 and 0.01 learning rate μ values, using four different values of α consisting of [0.3, 0.5, 0.7 and 0.9] and fractional orders (f_r) of 0.25, 0.5, 0.75 and 0.9 against different (k) features, to accomplish appropriate RMSE value. The values of two learning rate parameters of [fractional order and integer order] are same for fractional order methods, of $\mu_{f_r} = \mu = \mu$. It is observed that a large value of RMSE is obtained when higher values of μ [0.0005, 0.001, 0.005, and 0.01] are used or they did not show smooth convergence behavior for various (k) values.

Feature dimension (k) Selection

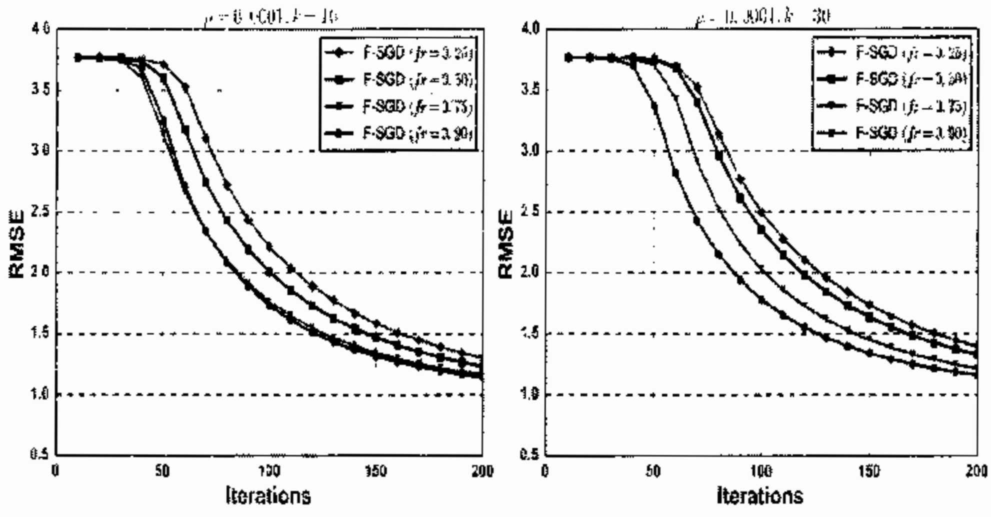
Computational efficiency of the proposed method (mF-SGD) is examined for variations in latent features (k) of 10, 20 and 30. It is observed that the computational complexity increases with the increase in number of features because time consumed by an algorithm primarily depends on the selection of the optimal number of runs and features (k). Algorithms are examined for 200 iterations to estimate the data matrix and to obtain the RMSE. It is also found that all the algorithms perform better when more features for example, 30 features are used. In Figs. 4.6 - 4.9, error curves for $k = 10$ and 30 are given to show the behavior of the algorithms due to variations in k.

Selecting the momentum term (α)

Two algorithms of mSGD and mF-SGD involve the momentum term α. Therefore, for both methods, four different values of α 0.3, 0.5, 0.7 and 0.9 are tested. The proposed mFSGD algorithm is evaluated by considering four different values of α 0.3, 0.5, 0.7 and 0.9 against four fractional orders (f_r) of 0.25, 0.5, 0.75, and 0.9 and learning rates (μ) values of 0.0001, 0.0005, 0.001, 0.005 and 0.01. It is seen that faster convergence of mF-SGD is accomplished with the increase in the value of α of 0.7 but at the cost of steady state performance which means better steady state is achieved for lower value of α and fast convergence is achieved for higher value of α.

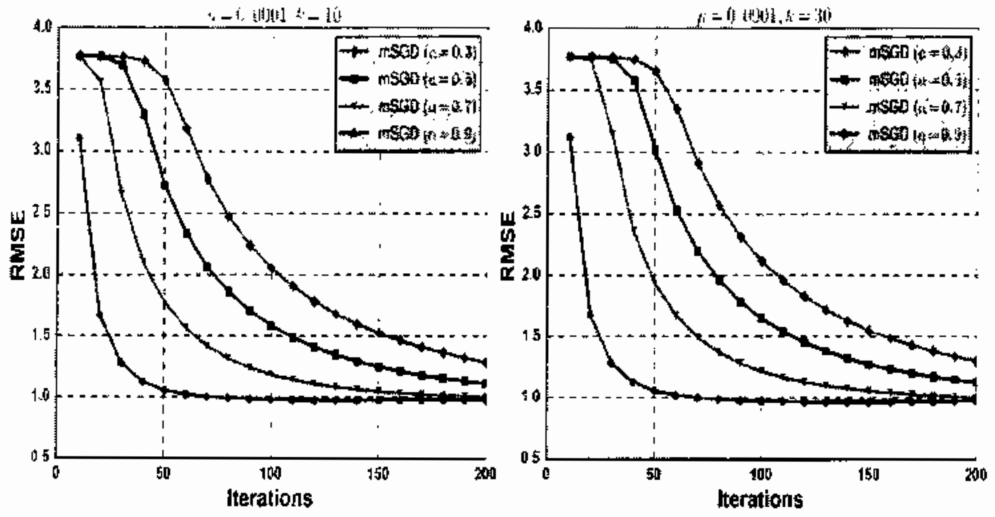
Selection of the fractional order (f_r)

The proposed method mF-SGD and FSGD are evaluated for four different values of fractional orders (f_r) 0.25, 0.5, 0.75, and 0.9, chosen from the range (0, 1). It is noticed that the rate of convergence and the steady state error increases for higher fractional orders of 0.9. The impact of fractional order on the convergence speed and accuracy has been explored in [62] using the fractional order in the range of (1, 1.5).



(a) (b)

Fig. 4.6 Convergence curves of F-SGD for different parametric values



(a) (b)

Fig. 4.7 Convergence curves of mSGD for various parametric values

4.3.2 Results and Discussion

The update rules in the standard SGD based algorithms are based on integer order gradient of the cost function. It is seen that by incorporating the fractional order gradient, the performance of mF-SGD is improved with respect to estimation accuracy and convergence at the cost of a small increase in computational complexity.

For the proposed method (mF-SGD), the convergence behaviour is estimated to demonstrate its performance. The fitness achieved in terms of RMSE for three methods (F-SGD, mSGD and mF-SGD) is presented in Tables 4.4 – 4.8 and graphically shown in Figs. 4.6 - 4.9 for 200 iterations with different number of features (k), fractional order (f_r) values and previous gradient values (α) against selected learning rate of $\mu = 0.0001$. It is observed from the Tables 4.4 – 4.8 and the curves in Figs. 4.6 - 4.9 that (mSGD) and (mF-SGD) exhibit faster convergence than F-SGD. Moreover, the convergence rate of momentum based methods increase with the increase in the percentage of previous gradients (α). It is also noticed that in terms of convergence, the proposed mF-SGD method outperforms other algorithms (mSGD and F-SGD) against different parameter values.

ML-100K Dataset

It is also demonstrated in Tables 4.6 - 4.8 and depicted in Figs. 4.8 - 4.9 that RMSE for mF-SGD decreases significantly for large values of f_r against different values of α and the number of latent features (k). The best RMSE of mF-SGD (0.962) is achieved with $\alpha = 0.7$, $f_r = 0.9$ and for $k = 30$, which clearly shows its advantage over other counterparts.

Table 4.4: Convergence comparison of F-SGD w.r.t RMSE attained for particular iterations

Epochs	$\mu = 0.0001$											
	$k = 10$				$k = 20$				$k = 30$			
	$f_r =$ 0.25	$f_r =$ 0.50	$f_r =$ 0.75	$f_r =$ 0.90	$f_r =$ 0.25	$f_r =$ 0.50	$f_r =$ 0.75	$f_r =$ 0.90	$f_r =$ 0.25	$f_r =$ 0.50	$f_r =$ 0.75	$f_r =$ 0.90
10	3.769	3.769	3.769	3.770	3.770	3.769	3.769	3.769	3.769	3.769	3.769	3.769
20	3.769	3.769	3.766	3.769	3.769	3.769	3.769	3.768	3.769	3.769	3.769	3.768
30	3.767	3.764	3.744	3.759	3.769	3.767	3.764	3.761	3.768	3.768	3.768	3.761
40	3.758	3.738	3.603	3.673	3.767	3.758	3.737	3.696	3.765	3.765	3.760	3.703
50	3.716	3.600	3.136	3.251	3.758	3.714	3.571	3.335	3.753	3.750	3.706	3.369
60	3.532	3.177	2.673	2.714	3.721	3.516	3.093	2.788	3.705	3.678	3.423	2.818
70	3.105	2.747	2.345	2.346	3.562	3.072	2.656	2.405	3.523	3.401	2.910	2.427
80	2.722	2.433	2.102	2.083	3.159	2.684	2.343	2.132	3.135	2.961	2.522	2.149
90	2.436	2.195	1.915	1.885	2.766	2.396	2.109	1.927	2.772	2.614	2.240	1.942
100	2.215	2.008	1.767	1.733	2.471	2.174	1.926	1.769	2.495	2.352	2.027	1.781
110	2.038	1.857	1.647	1.612	2.242	1.997	1.780	1.643	2.278	2.146	1.859	1.654
120	1.893	1.732	1.549	1.514	2.060	1.854	1.662	1.541	2.103	1.980	1.725	1.551
130	1.773	1.629	1.468	1.434	1.911	1.735	1.564	1.458	1.958	1.844	1.616	1.467
140	1.671	1.542	1.400	1.368	1.787	1.635	1.483	1.389	1.837	1.731	1.525	1.397
150	1.585	1.468	1.343	1.314	1.683	1.550	1.415	1.331	1.734	1.635	1.450	1.340
160	1.511	1.405	1.295	1.268	1.594	1.478	1.357	1.283	1.645	1.553	1.386	1.291
170	1.447	1.352	1.254	1.229	1.518	1.416	1.308	1.242	1.569	1.484	1.332	1.250
180	1.391	1.305	1.219	1.196	1.452	1.362	1.266	1.207	1.502	1.423	1.286	1.215
190	1.342	1.265	1.188	1.169	1.396	1.315	1.230	1.178	1.444	1.371	1.247	1.185
200	1.300	1.231	1.163	1.145	1.346	1.275	1.198	1.152	1.393	1.326	1.214	1.159

Table 4.5: Convergence comparison of mSGD w.r.t RMSE attained for particular iterations

Epochs	$\mu = 0.0001$											
	$k = 10$				$k = 20$				$k = 30$			
	$\alpha = 0.30$	$\alpha = 0.50$	$\alpha = 0.70$	$\alpha = 0.90$	$\alpha = 0.30$	$\alpha = 0.50$	$\alpha = 0.70$	$\alpha = 0.90$	$\alpha = 0.30$	$\alpha = 0.50$	$\alpha = 0.70$	$\alpha = 0.90$
10	3.769	3.769	3.764	3.104	3.769	3.769	3.768	3.118	3.769	3.769	3.769	3.116
20	3.768	3.763	3.555	1.677	3.769	3.765	3.699	1.684	3.769	3.768	3.741	1.682
30	3.762	3.696	2.649	1.282	3.767	3.727	2.913	1.284	3.765	3.749	3.139	1.284
40	3.728	3.298	2.103	1.127	3.755	3.438	2.249	1.125	3.747	3.577	2.360	1.128
50	3.573	2.725	1.779	1.057	3.693	2.841	1.872	1.053	3.653	3.015	1.938	1.056
60	3.187	2.337	1.568	1.023	3.454	2.412	1.631	1.016	3.346	2.527	1.674	1.020
70	2.776	2.065	1.423	1.004	3.020	2.118	1.468	0.996	2.910	2.200	1.497	1.000
80	2.471	1.864	1.319	0.993	2.647	1.904	1.352	0.985	2.568	1.964	1.373	0.987
90	2.239	1.709	1.243	0.986	2.372	1.741	1.267	0.977	2.314	1.787	1.283	0.979
100	2.056	1.588	1.186	0.981	2.160	1.614	1.203	0.972	2.116	1.649	1.216	0.972
110	1.908	1.492	1.143	0.978	1.992	1.512	1.155	0.969	1.957	1.541	1.165	0.968
120	1.787	1.413	1.110	0.976	1.855	1.430	1.119	0.968	1.827	1.453	1.127	0.965
130	1.686	1.349	1.084	0.975	1.742	1.363	1.090	0.967	1.719	1.382	1.097	0.964
140	1.600	1.295	1.064	0.975	1.647	1.308	1.068	0.968	1.629	1.324	1.074	0.963
150	1.527	1.251	1.048	0.975	1.567	1.262	1.050	0.969	1.552	1.275	1.055	0.963
160	1.465	1.214	1.035	0.976	1.499	1.223	1.036	0.972	1.486	1.234	1.040	0.965
170	1.411	1.182	1.025	0.977	1.440	1.190	1.024	0.974	1.430	1.200	1.028	0.967
180	1.364	1.156	1.016	0.978	1.389	1.163	1.015	0.977	1.381	1.170	1.018	0.969
190	1.323	1.133	1.009	0.979	1.345	1.139	1.007	0.980	1.338	1.146	1.010	0.973
200	1.287	1.113	1.004	0.980	1.306	1.119	1.001	0.984	1.300	1.124	1.003	0.977

Table 4.6: Convergence comparison of mF-SGD w.r.t RMSE attained for particular iterations with $k = 10$, $\mu = 0.0001$

Epochs	$\alpha = 0.3$				$\alpha = 0.5$				$\alpha = 0.7$				$\alpha = 0.9$			
	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**
10	3.769	3.769	3.768	3.770	3.769	3.769	3.768	3.767	3.761	3.764	3.747	**3.707**	2.427	2.068	1.928	1.797
20	3.767	3.765	3.750	3.762	3.766	3.755	3.685	3.576	3.116	3.083	2.556	**2.305**	1.364	1.252	1.186	1.152
30	3.752	3.718	3.503	3.600	3.700	3.419	2.801	2.576	2.163	2.090	1.808	**1.673**	1.101	1.064	1.035	1.027
40	3.612	3.320	2.790	2.850	3.042	2.589	2.163	2.015	1.718	1.642	1.463	**1.382**	1.024	1.012	0.993	0.991
50	3.044	2.697	2.307	2.304	2.396	2.107	1.803	1.692	1.463	1.393	1.276	**1.227**	0.998	0.993	0.979	0.980
60	2.551	2.290	1.990	1.964	2.006	1.803	1.573	1.488	1.303	1.244	1.167	**1.137**	0.987	0.986	0.974	0.976
70	2.216	2.008	1.767	1.733	1.746	1.594	1.417	1.350	1.200	1.153	1.100	**1.083**	0.982	0.982	0.972	0.975
80	1.973	1.801	1.603	1.568	1.562	1.445	1.307	1.255	1.132	1.097	1.059	**1.049**	0.980	0.981	0.972	0.976
90	1.789	1.643	1.479	1.445	1.428	1.336	1.228	1.186	1.087	1.061	1.033	**1.027**	0.980	0.981	0.973	0.978
100	1.646	1.520	1.383	1.352	1.327	1.255	1.170	1.137	1.056	1.038	1.016	**1.012**	0.981	0.983	0.976	0.981
110	1.531	1.423	1.308	1.280	1.250	1.194	1.128	1.100	1.036	1.023	1.004	**1.001**	0.981	0.985	0.978	0.983
120	1.438	1.345	1.248	1.224	1.191	1.149	1.096	1.073	1.021	1.012	0.996	**0.993**	0.982	0.987	0.980	0.986
130	1.363	1.282	1.201	1.180	1.147	1.114	1.072	1.053	1.010	1.005	0.990	**0.987**	0.984	0.990	0.983	0.989
140	1.300	1.231	1.163	1.145	1.112	1.088	1.053	1.037	1.003	0.999	0.986	**0.983**	0.986	0.992	0.986	0.992
150	1.249	1.189	1.132	1.117	1.085	1.068	1.039	1.025	0.997	0.995	0.983	**0.979**	0.988	0.995	0.989	0.996
160	1.206	1.155	1.107	1.094	1.064	1.052	1.027	1.015	0.993	0.992	0.980	**0.977**	0.990	0.998	0.992	0.999
170	1.171	1.127	1.086	1.075	1.048	1.040	1.019	1.007	0.990	0.990	0.979	**0.975**	0.992	1.000	0.995	1.002
180	1.142	1.103	1.070	1.060	1.035	1.030	1.012	1.001	0.987	0.988	0.978	**0.974**	0.994	1.003	0.997	1.006
190	1.117	1.084	1.056	1.048	1.025	1.022	1.006	0.996	0.985	0.987	0.977	**0.974**	0.996	1.006	1.000	1.009
200	1.097	1.068	1.044	1.038	1.017	1.016	1.001	0.992	0.984	0.985	0.977	**0.973**	0.998	1.009	1.003	1.012

It is observed that for all the algorithms (mSGD, F-SGD and mF-SGD), RMSE decreases with an increase in the number of features for different parameter values of f_r and α. It is seen from the Figs. 4.6 - 4.9 that mF-SGD and mSGD outperform F-SGD in terms of RMSE for various parameter setting.

For comparison purpose, the error curves for the three algorithms for different values of f_r, k and α with optimal learning rate $\mu = 0.0001$ are given in Figs. 4.10, 4.11 and 4.12 respectively. The improved performance of proposed method of mF-SGD is shown in Figs. 4.10 - 4.12 for different α and f_r values. An RMSE convergence of mF-SGD against F-SGD and mSGD with $\mu = 0.0001, k = 10$ and $\alpha = 0.3, 0.5, 0.7$ *and* 0.9 and $f_r = 0.9$ is presented in Fig. 4.10 (a), (b), (c) and (d) respectively, the finest convergence can be seen in Fig. 4.10 (c) in which mF-SGD for $\alpha = 0.7$ and $f_r = 0.9$ converge significantly fast and achieved minimum RMSE (0.973) after 200 iterations. Similarly, it is shown in Fig. 4.11 (a), (b), (c) and (d) that mF-SGD also accomplished fast convergence for $k = 20$ with ($\mu = 0.0001$ and $\alpha = 0.3, 0.5, 0.7$ *and* 0.9) with fractional order ($f_r = 0.9$) but mF-SGD optimal convergence in terms of RMSE (0.963) for $k = 20$ is achieved at $\alpha = 0.7$ and $f_r = 0.9$. Likewise, Fig. 4.12 (a), (b), (c) and (d) illustrate the better convergence offered by mF-SGD over F-SGD and mSGD for $k = 30$ with ($\mu = 0.0001$ and $\alpha = 0.3, 0.5, 0.7$ *and* 0.9) using fractional order ($f_r = 0.9$), while the minimum convergence in terms of RMSE (0.962) for 30 features is achieved for $\alpha = 0.7$ and $f_r = 0.9$. In Figs 4.10 (d), 4.11 (d) and 4.12 (d) RMSE curves for the proposed mF-SGD are early stopped at about 100 iterations to avoid over-fitting.

Furthermore, it is observed that for different features ($k = 10, 20, 30$), F-SGD and mSGD achieved the minimum RMSE after about 200 iterations, while mF-SGD takes a smaller number of iterations. It is also seen that the momentum based algorithms have shown improved performance in terms of RMSE for a large k. It is observed that after 200 iterations, bigger value of α and a large f_r leads to a smaller RMSE except for $\alpha = 0.9$. It is also observed that for the proposed method of mF-SGD, a large value of f_r with higher value of α speeds up the convergence. In addition, increasing the f_r value leads to a small increase in the accuracy.

ML-1M Dataset

A Comprehensive set of investigations of the algorithms (mF-SGD, mSGD and F-SGD) are carried out for bigger dataset i.e., ML-1M (1-million), for different values of α i.e. [0.3, 0.5, 0.7 and 0.9], fractional order (f_r) i.e., [0.3, 0.5, 0.75, and 0.9] and learning rates (μ) i.e., [0.0001, 0.0005, 0.001, 0.005 and 0.01] against 30 features. The results achieved by mF-SGD, mSGD and F-SGD based on optimal parameters for the ML-1M dataset are given in the Fig 4.13.

Behavior similar to that for the ML-100k dataset in terms of convergence is observed for the ML-1M dataset. The optimal learning rate (0.0001) chosen for the bigger dataset is the same as that for the smaller dataset. It is noticed that all the compared methods show divergence or increasing trend in terms of RMSE for bigger values of learning rate μ 0.0005, 0.001, 0.005, and 0.01 against different features. The proposed algorithm performs better for higher values of both the fractional order f_r and weight α. It is shown that fast convergence of mFSGD is achieved for a large value of α but with the compromise of steady state performance. The best RMSE value achieved by the proposed algorithm for the ML-1M dataset is 0.887 for α = 0.7, f_r = 0.9 after 150 iterations, whereas, the optimal RMSE values accomplished by

Table 4.7: Convergence comparison of mF-SGD w.r.t RMSE attained for particular iterations with $k = 20$, $\mu = 0.0001$

Epochs	$\alpha = 0.3$				$\alpha = 0.5$				$\alpha = 0.7$				$\alpha = 0.9$			
	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**
10	3.769	3.769	3.769	3.769	3.769	3.768	3.769	3.769	3.767	3.764	3.760	**3.737**	2.442	2.301	2.024	1.805
20	3.768	3.768	3.767	3.761	3.764	3.751	3.727	3.712	3.514	3.193	2.783	**2.427**	1.333	1.308	1.206	1.153
30	3.758	3.754	3.734	3.609	3.674	3.408	3.005	2.840	2.408	2.177	1.908	**1.731**	1.079	1.079	1.044	1.027
40	3.680	3.618	3.357	2.874	2.938	2.601	2.285	2.153	1.870	1.711	1.518	**1.414**	1.011	1.014	0.998	0.989
50	3.255	3.039	2.674	2.328	2.329	2.131	1.887	1.778	1.567	1.449	1.309	**1.245**	0.988	0.990	0.980	0.974
60	2.721	2.530	2.242	1.987	1.960	1.829	1.634	1.545	1.378	1.289	1.187	**1.147**	0.977	0.978	0.972	0.967
70	2.353	2.190	1.951	1.754	1.712	1.621	1.463	1.390	1.254	1.187	1.114	**1.088**	0.970	0.973	0.969	0.965
80	2.087	1.945	1.743	1.586	1.535	1.470	1.343	1.283	1.171	1.120	1.068	**1.051**	0.966	0.972	0.970	0.968
90	1.885	1.761	1.588	1.461	1.405	1.359	1.255	1.207	1.114	1.075	1.038	**1.026**	0.964	0.973	0.973	0.973
100	1.728	1.619	1.469	1.366	1.308	1.275	1.191	1.151	1.075	1.046	1.018	**1.009**	0.963	0.976	0.978	0.979
110	1.602	1.506	1.377	1.292	1.234	1.211	1.143	1.110	1.048	1.025	1.004	**0.997**	0.964	0.981	0.984	0.987
120	1.500	1.416	1.304	1.234	1.178	1.162	1.107	1.080	1.028	1.010	0.994	**0.988**	0.966	0.986	0.991	0.994
130	1.417	1.342	1.246	1.188	1.135	1.124	1.079	1.056	1.014	0.999	0.987	**0.981**	0.969	0.992	0.997	1.002
140	1.348	1.282	1.200	1.151	1.102	1.095	1.058	1.038	1.003	0.991	0.982	**0.976**	0.973	0.998	1.004	1.010
150	1.290	1.233	1.162	1.121	1.077	1.072	1.041	1.025	0.996	0.986	0.978	**0.972**	0.977	1.004	1.011	1.017
160	1.242	1.192	1.131	1.097	1.057	1.054	1.028	1.014	0.990	0.981	0.975	**0.969**	0.981	1.010	1.017	1.024
170	1.203	1.158	1.106	1.077	1.042	1.039	1.018	1.005	0.985	0.978	0.973	**0.967**	0.985	1.016	1.024	1.032
180	1.169	1.130	1.085	1.061	1.030	1.027	1.009	0.998	0.982	0.975	0.971	**0.965**	0.990	1.021	1.030	1.039
190	1.141	1.107	1.068	1.047	1.020	1.018	1.002	0.992	0.979	0.974	0.970	**0.964**	0.994	1.027	1.036	1.045
200	1.117	1.088	1.054	1.036	1.012	1.010	0.997	0.987	0.977	0.972	0.969	**0.963**	0.999	1.033	1.042	1.052

Table 4.8: Convergence comparison of mF-SGD w.r.t RMSE attained for particular iterations with $k = 30$, $\mu = 0.0001$

Epochs	$\alpha = 0.3$				$\alpha = 0.5$				$\alpha = 0.7$				$\alpha = 0.9$			
	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$\boldsymbol{f_r} =$ **0.90**	$f_r =$ **0.25**	$f_r =$ **0.50**	$f_r =$ **0.75**	$f_r =$ **0.90**
10	3.769	3.769	3.769	3.769	3.769	3.769	3.769	3.768	3.768	3.768	3.765	**3.752**	2.755	2.360	2.094	1.903
20	3.769	3.768	3.766	3.764	3.767	3.765	3.749	3.683	3.685	3.562	2.983	**2.533**	1.446	1.303	1.227	1.172
30	3.765	3.759	3.728	3.658	3.736	3.672	3.216	2.761	2.619	2.363	2.014	**1.776**	1.129	1.073	1.051	1.033
40	3.736	3.677	3.321	2.979	3.319	2.946	2.402	2.123	1.973	1.801	1.585	**1.439**	1.032	1.007	0.999	0.993
50	3.539	3.206	2.648	2.392	2.602	2.345	1.960	1.766	1.622	1.499	1.353	**1.260**	0.997	0.983	0.978	0.975
60	2.984	2.655	2.227	2.030	2.164	1.978	1.683	1.541	1.407	1.318	1.217	**1.157**	0.982	0.971	0.967	0.966
70	2.539	2.287	1.942	1.785	1.872	1.731	1.497	1.390	1.269	1.205	1.134	**1.094**	0.975	0.964	0.962	0.962
80	2.228	2.024	1.738	1.609	1.666	1.555	1.366	1.286	1.177	1.132	1.081	**1.055**	0.970	0.960	0.961	0.962
90	1.998	1.827	1.586	1.479	1.513	1.426	1.272	1.211	1.116	1.084	1.047	**1.030**	0.968	0.959	0.963	0.966
100	1.821	1.675	1.469	1.381	1.397	1.329	1.203	1.156	1.075	1.052	1.025	**1.012**	0.968	0.959	0.967	0.972
110	1.681	1.555	1.378	1.304	1.308	1.255	1.152	1.116	1.047	1.030	1.009	**0.999**	0.969	0.962	0.972	0.979
120	1.568	1.458	1.306	1.244	1.238	1.197	1.114	1.085	1.027	1.014	0.998	**0.990**	0.973	0.966	0.979	0.987
130	1.475	1.379	1.249	1.197	1.184	1.153	1.085	1.062	1.014	1.002	0.990	**0.983**	0.977	0.971	0.986	0.995
140	1.399	1.315	1.203	1.159	1.142	1.118	1.062	1.044	1.004	0.993	0.983	**0.977**	0.982	0.977	0.994	1.003
150	1.335	1.262	1.165	1.128	1.109	1.091	1.045	1.030	0.996	0.986	0.979	**0.973**	0.987	0.984	1.000	1.011
160	1.281	1.218	1.135	1.103	1.083	1.069	1.031	1.019	0.990	0.981	0.975	**0.969**	0.993	0.991	1.008	1.019
170	1.236	1.181	1.110	1.083	1.063	1.051	1.020	1.010	0.986	0.977	0.972	**0.966**	0.999	0.998	1.015	1.026
180	1.199	1.150	1.089	1.066	1.046	1.037	1.012	1.002	0.982	0.974	0.969	**0.965**	1.005	1.005	1.022	1.034
190	1.167	1.125	1.072	1.052	1.034	1.025	1.004	0.996	0.979	0.971	0.968	**0.963**	1.011	1.012	1.030	1.041
200	1.140	1.103	1.058	1.040	1.023	1.016	0.998	0.991	0.976	0.969	0.966	**0.962**	1.017	1.020	1.037	1.048

mSGD and F-SGD after 200 iterations are 0.897 for α = 0.7 and 0.930 for f_r = 0.9 respectively.

However, for $\alpha = 0.9$, mF-SGD has fastest initial convergence than mSGD and F-SGD. For about 55 early iterations mF-SGD remained convergent but it starts diverging for subsequent iterations. Whereas, mSDG achieved minimum convergence for $\alpha = 0.9$, which is attained 30 iterations later than mF-SGD but starts diverging after around 100 iterations. On the other hand mF-SGD exhibits slow initial convergent for $\alpha = 0.7$ and achieved minimum RMSE = 0.887 after 150 iterations and maintains stable steady state behavior for almost 180 iterations, which is not shown by mSGD for $\alpha = 0.9$.

Performance comparison with Deep Learning based Matrix Factorization models

Apart from competing methods of mSGD and F-SGD presented in this research, the effectiveness of the proposed mF-SGD is also proved by comparing it with recent deep learning based matrix factorization methods using **ML-100K** and **ML-1M datasets.** The performance in terms of RMSE of deep learning-based models is reported with optimum parameter values. Overview of the deep learning based MF models given in [111] is as follows.

- **ConvMF** [111] To improve the prediction accuracy of ratings, a context-aware recommendation model (ConvMF) is proposed. For achieving high prediction accuracy, ConvMF integrates CNN with PMF to capture contextual information of documents as stated in [112].

- **DBPMF** [111]: Deep Bias Probabilistic Matrix Factorization model (DBPMF) uses CNN to extract hidden user/item characteristics. Moreover, DBPMF also adds bias into PMF to tract user ratings behavior and item reputation.
- **DCBPMF** [111]: Deep Constrain Bias PMF method is used to further improve the performance of standard DBPMF by adding constrain to the user specific and item specific vectors.

The performance comparison between proposed mF-SGD and deep learning based MF methods for recommender systems [111] is demonstrated in Table 4.9 for both datasets i.e., ML-100K and ML-1M. It is noticed from the results given in Table 4.9 that mF-SGD achieved improved results than **ConvMF, DBPMF** and **DCBPMF** for both datasets. . The best performance of RMSE = 0.962 is accomplished by mF-SGD against counterparts for ML-100K is with $\alpha = 0.7, f_r = 0.7$ and $k = 30$. Whereas, for ML-1M dataset mF-SGD also has achieved finest RMSE = 0.887 against competing methods with similar parameters setting i.e., $\alpha = 0.7$, $f_r = 0.9$ and $k = 30$. The significant performance with regard to deep learning based MF methods confirms the usefulness of the proposed mF-SGD for proposing accurate and fast recommendations.

Table 4.9: Performance comparison of mF-SGD with deep learning based MF methods

DATASETS	METHODS	RMSE
ML-100K	ConvMF	1.000
	DBPMF	0.990
	DCBPMF	0.985
	mF-SGD	**0.962**
ML-1M	ConvMF	0.980
	DBPMF	0.945
	DCBPMF	0.943
	mF-SGD	**0.887**

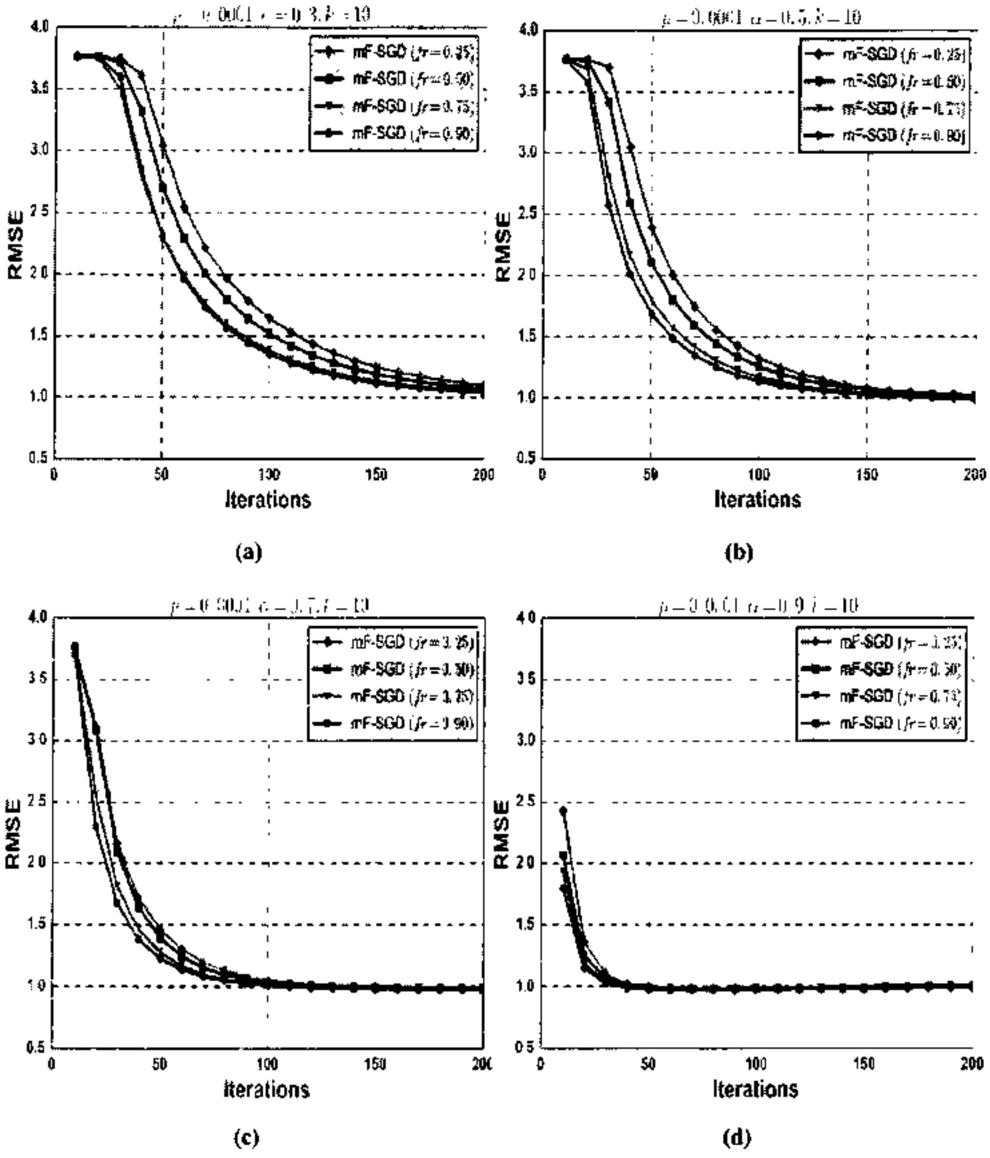


Fig. 4.8 Convergence curves of mF-SGD for different f_r and α with $k = 10$, $\mu = 0.0001$

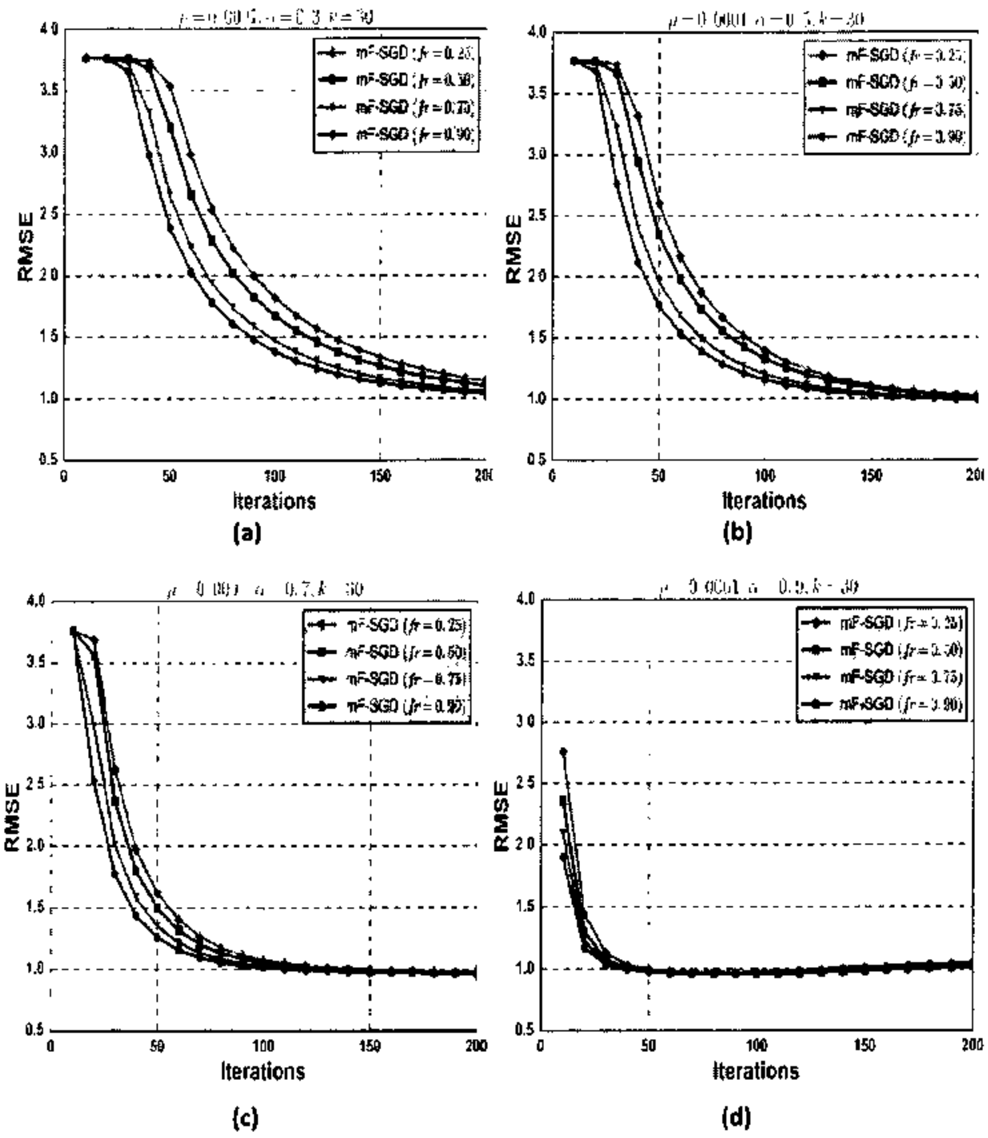


Fig. 4.9 Convergence curves of mF-SGD for different f_r and α with k = 30, μ = 0.0001

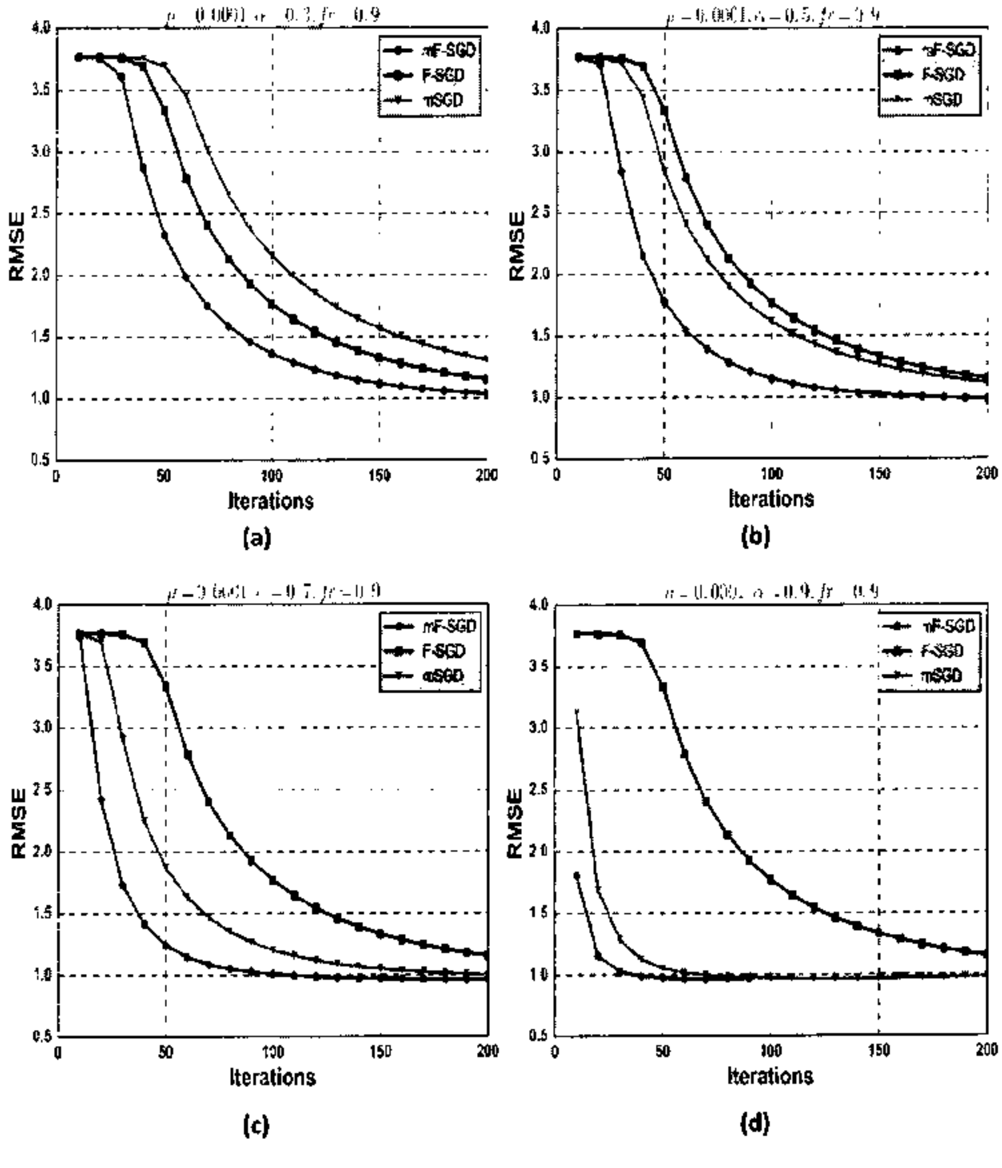


Fig. 4.11 Convergence curves of mF-SGD vs F-SGD and mSGD for various α and $k = 20$

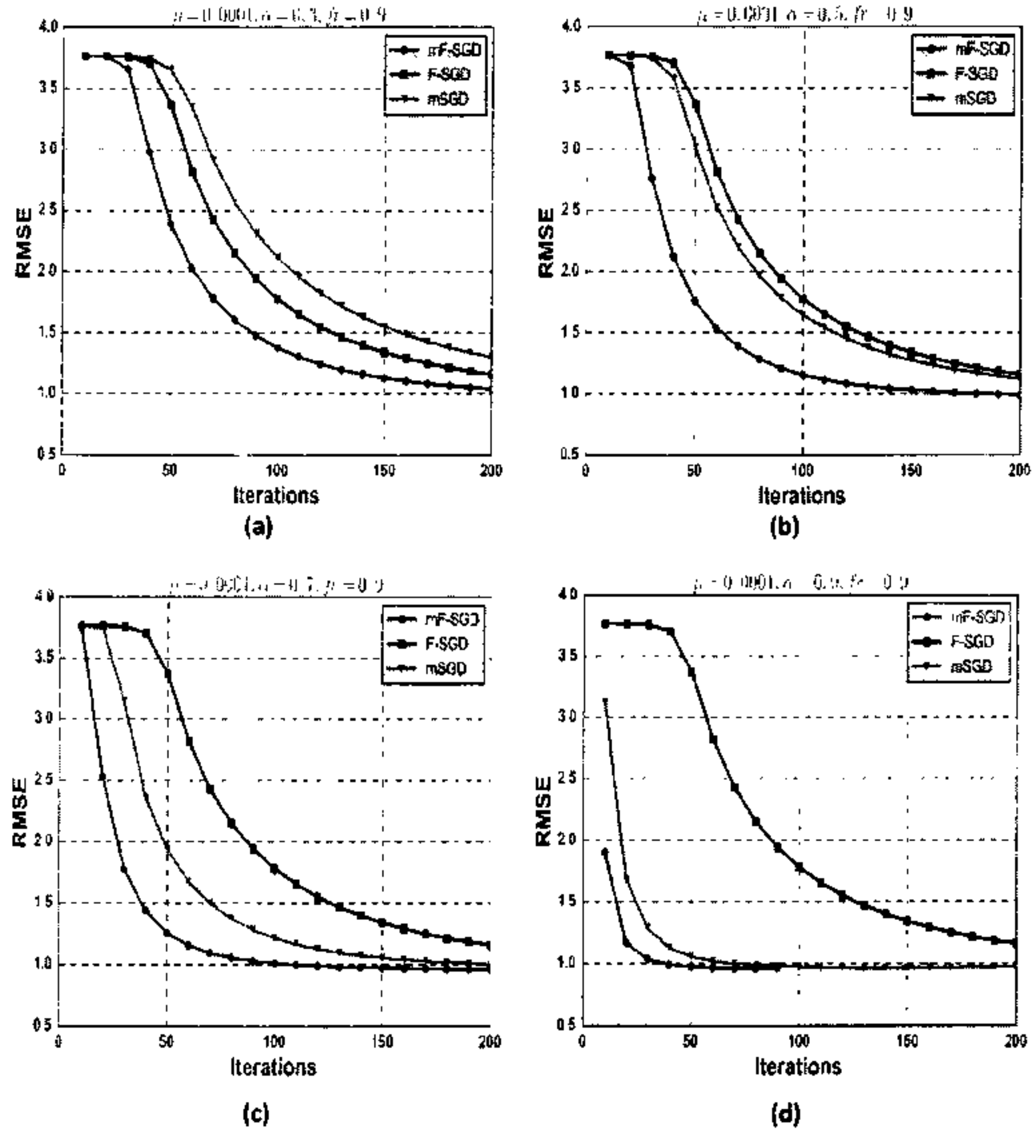


Fig. 4.12 Convergence curves of mF-SGD vs F-SGD and mSGD for various α and $k = 30$

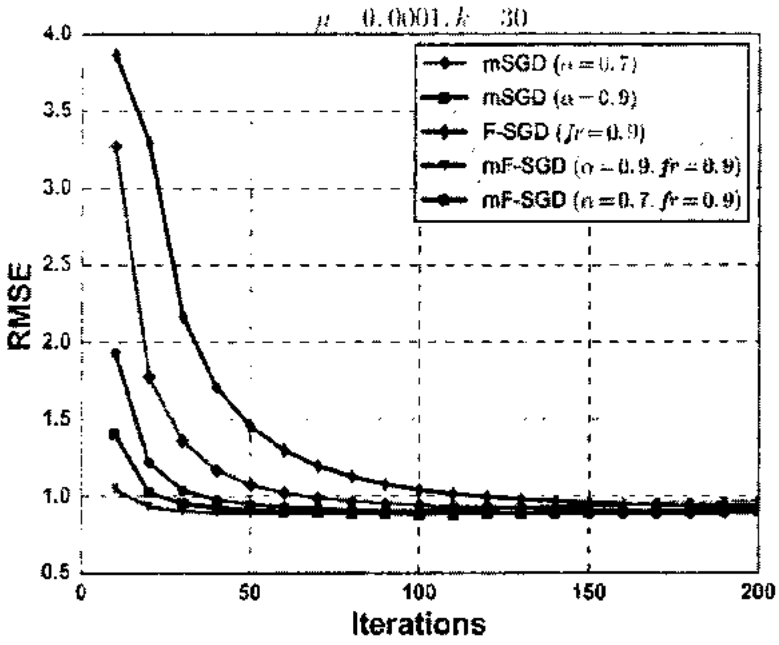


Fig. 4.13 RMSE Convergence of mF-SGD, F-SGD and mSGD for ML-1M Dataset

4.4 Case-Study-III: NSGD, F-SGD and proposed NF-SGD algorithms

In Case-Studies I and II, the proposed methods have demonstrated improved behavior than standard adaptive strategies for recommender systems but adaptive nature of learning rate important for the convergence and stability of adaptive methods has not been explored in those case studies. Therefore, in this case study a nonlinear computing paradigm based on normalized version of fractional SGD is presented and compared with standard adaptive methods to investigate the adaptive behavior of learning rate with novel application to recommender systems. The update relations of proposed NF-SGD for fast and precise recommendations are given in (3.27), (3.28) and mathematical expression representing update rules for standard methods are specified in (3.14), (3.15), (3.25), (3.26). Flow diagram of Case-Study-III (NF-SGD) using matrix factorization is graphically given in Fig. 4.14. Simulation parameters for this case study including simulation results are discussed in succeeding subsections.

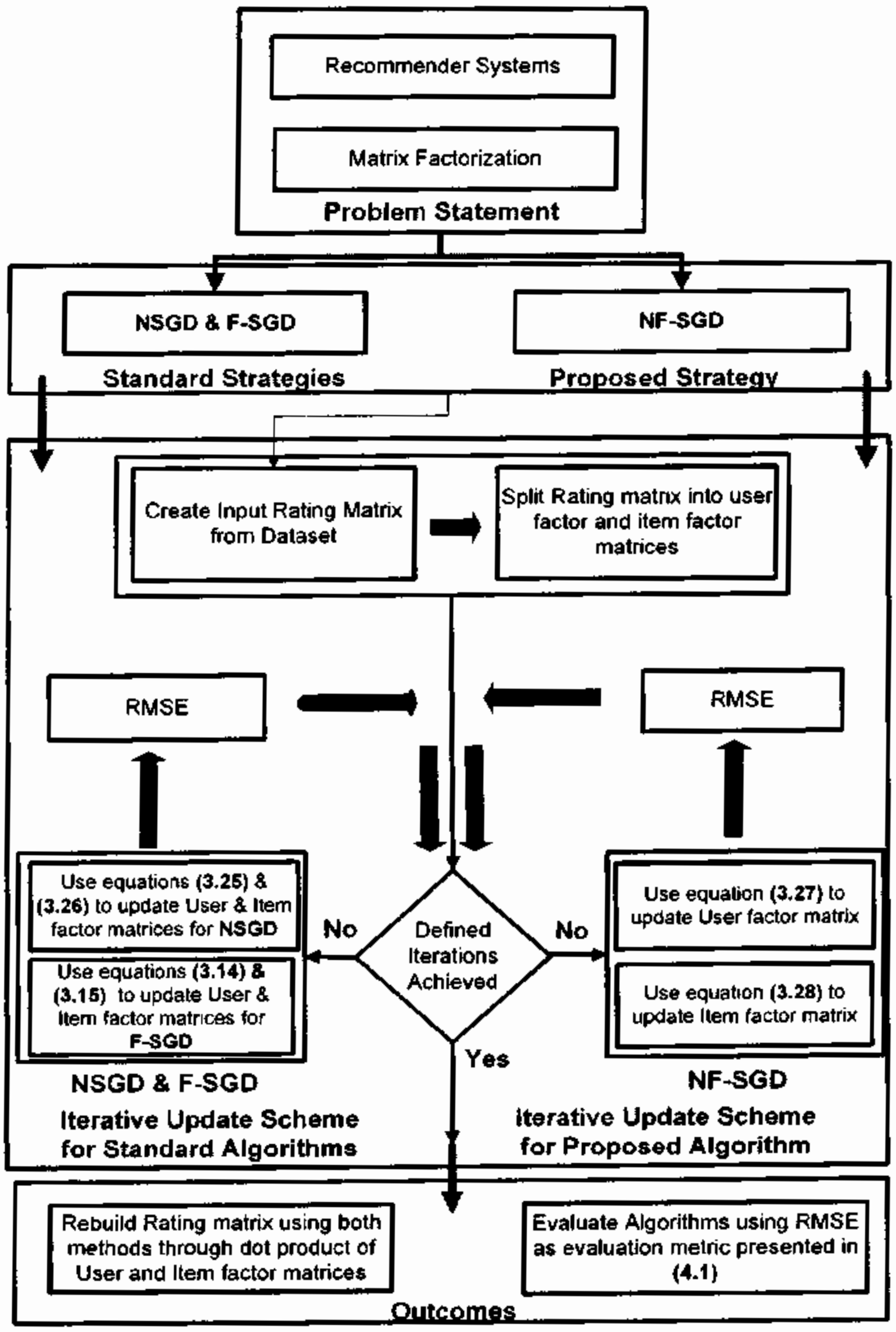
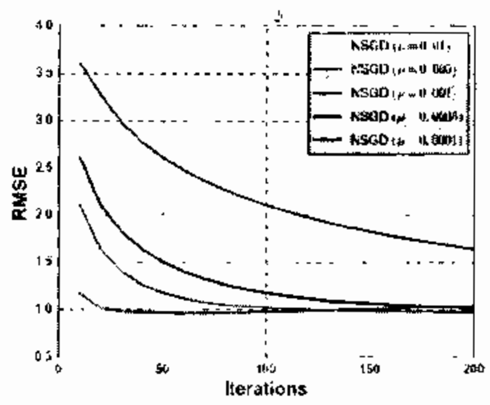

Recommender Systems
Matrix Factorization
Problem Statement
NSGD & F-SGD
Standard Strategies
NF-SGD
Proposed Strategy
Create Input Rating Matrix from Dataset
Split Rating matrix into user factor and item factor matrices
RMSE
RMSE
Use equations (3.25) & (3.26) to update User & Item factor matrices for NSGD
Use equations (3.14) & (3.15) to update User & Item factor matrices for F-SGD
No
Defined Iterations Achieved
No
Use equation (3.27) to update User factor matrix
Use equation (3.28) to update Item factor matrix
Yes
NSGD & F-SGD
Iterative Update Scheme for Standard Algorithms
NF-SGD
Iterative Update Scheme for Proposed Algorithm
Rebuild Rating matrix using both methods through dot product of User and Item factor matrices
Evaluate Algorithms using RMSE as evaluation metric presented in (4.1)
Outcomes

Fig. 4.14 Graphical flow of MF based NF-SGD for recommender systems

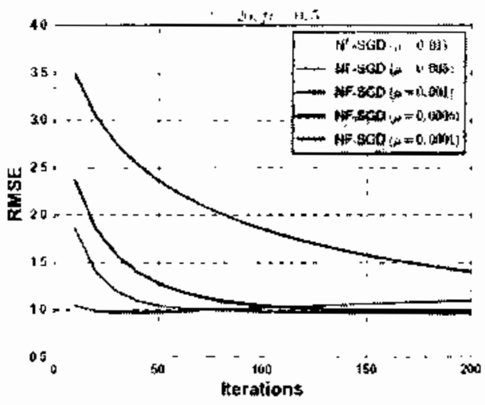
4.4.1 Simulation Parameters

For fractional gradient based algorithms, the learning rates corresponding to integer order and fractional order terms are taken at $\mu_i = \mu_{f_r} = \mu$. To achieve fast and stable convergence in terms of RMSE the optimal learning rate of $\mu = 10^{-3}$ for NSGD and NF-SGD and $\mu = 0.0005$ for F-SGD is selected after executing several trials for various learning rate parameter values of 0.01, 0.005, 0.001, 0.0005 and 0.0001, with various fractional order (f_r) values of 0.3, 0.5, 0.6 and 0.9 and for three values of features (k) of 10, 20 and 30. It is noticed that all the three algorithms show divergent behavior for learning rates 0.01 and 0.005 whereas they show comparatively fast and smooth convergence for $\mu = 0.0005$ and 0.001. For NSGD and NF-SGD algorithms, fastest and stable convergence is presented for $\mu = 0.001$, whereas, F-SGD shows fast and stable convergence for $\mu = 0.0005$. Hence, we select these values of μ for our experiments. Learning rate tuning of NSGD, NF-SGD and F-SGD is demonstrated in Fig. 4.15 (a), (b) and (c) respectively against μ = [0.01, 0.005, 0.001, 0.0005 and 0.0001], $k = 20$ and $f_r = 0.5$.

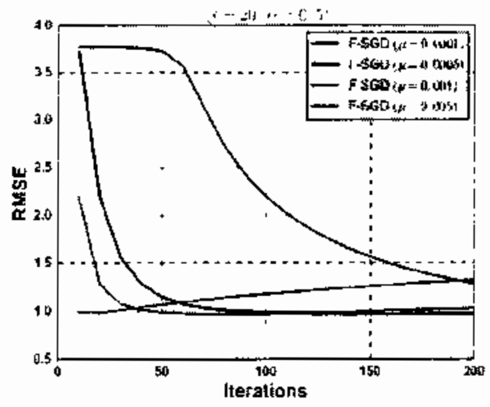
Choice of fractional order plays a significant role in the convergence of fractional adaptive schemes. To decide an optimal fractional order of our proposed method of NF-SGD, RMSE is calculated for different fractional order (f_r) values of 0.3, 0.6, and 0.9 for 200 iterations and learning curves are given Fig. 4.16 (a) and (b) for $k = 10$ and 30 respectively, with optimal learning rate μ = [0.001]. It is observed that for both variations of k, fast convergence rate and increased estimated accuracy are achieved for higher value of fractional order i.e., $f_r = 0.9$.



(a)



(b)



(c)

Fig. 4.15 Learning rate tuning for NSGD, F-SGD and NF-SGD with $k = 20$

Another important parameter that affects the convergence behavior of the recommender system is k, number of features of a latent vector. In this work, NSGD, F-SGD and NF-SGD are evaluated for three variants of hidden features of 10, 20 and 30. To approximate the rating matrix, given methods are observed for 200 epochs. It is noticed that these methods provide better RMSE for greater number of features of $k =$ 30. However, more features correspond to an increase in computational complexity.

The common perception is that the optimum performance achieved by adaptive algorithms is different for different datasets. No single model always produces the finest outcomes for all datasets. So, one must select the fitness function and values of hyper-parameters such as fractional order (f_r), learning rate (μ, μ_{f_r}) and number of features (k) according to the dataset. The simulation parameters used by the standard algorithms of F-SGD and NSGD and the proposed one (NF-SGD) for the achievement of best performance in terms of fast convergence are noted and summarized in Table 4.10. These summarized results may help the researchers to use these parameters settings for their matrix factorization-based models for recommender systems using ML-100K and ML-1M datasets.

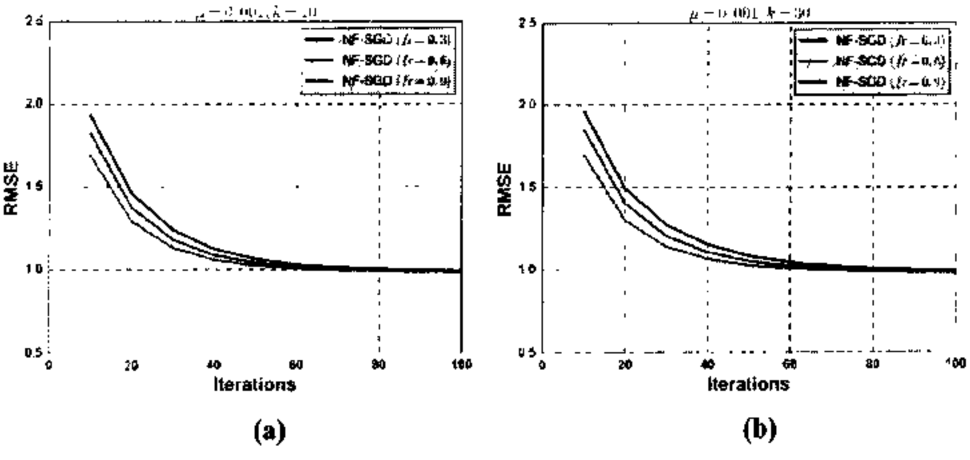


Fig. 4.16 Fractional order tuning for proposed NF-SGD with k = 10 and k = 30

4.4.2 Results and Discussion

The performance of NSGD, F-SGD and proposed NF-SGD is evaluated through RMSE values achieved at different iterations using optimal values of learning rate of μ = 0.001 for NF-SGD and NSGD, while μ = 0.0005 for F-SGD, and results are presented in Table 4.11 and 4.12 for ML-100k and ML-1M datasets respectively in case of all three variations of k. Fast convergence of NF-SGD than NSGD and F-SGD is evidently observed from Table 4.11 and 4.12 for different f_r variations.

Discussion (ML-100K)

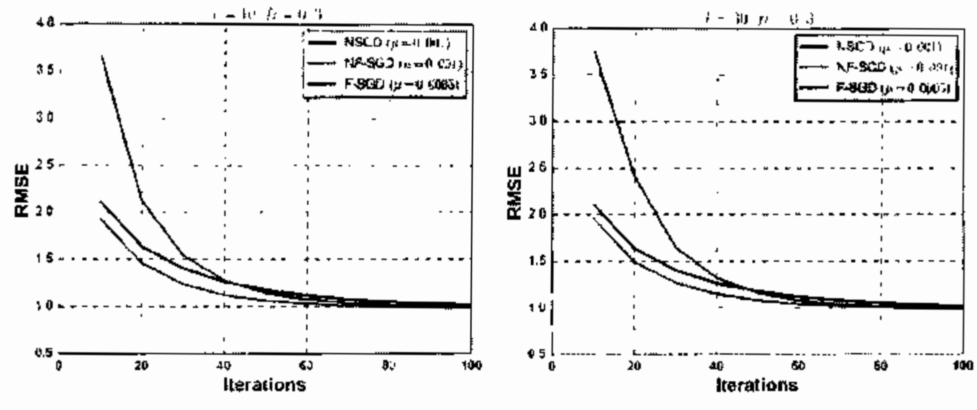
The comparison of initial convergence rates of NSGD, F-SGD and the proposed NF-SGD for ML-100K dataset for 100 iterations with various parameter values (f_r and k) is given in Table 4.11. It is perceived from Table 4.11 that the suggested NF-SGD achieves much faster initial convergence than other methods for higher values of fractional orders (f_r) and features (k). After 100 iterations, RMSE attained by NF-SGD and F-SGD for μ = 0.001, k = 30 and f_r = 0.9 are (0.956) and (0.974) respectively, while RMSE obtained by NSGD for μ = 0.0005, k = 30 and f_r = 0.9 is (0.976).

Learning curves of RMSE for 100 iterations using best initial value of the learning rate i.e., μ = 0.001 for normalized methods (NSGD and NF-SGD) and optimal value of learning rate i.e., μ = 0.0005 for F-SGD are represented in Figs. 4.17, 4.18 and 4.19. Learning curves in Figs. 4.17, 4.18 and 4.19 are given for three f_r i.e., [0.3, 0.6 and 0.9] and two k i.e., [10 and 30] variations. The substantial performance of the NF-SGD in terms of convergence is clearly seen in Figs 4.17, 4.18 and 4.19 for different parameter values. The convergence comparison for f_r = 0.3 between proposed NF-SGD and standard baselines i.e., (F-SGD and NSGD) for k =10 and k = 30 is given in Fig. 4.17 (a) and (b) respectively. It is seen that NF-SGD performs better in terms of convergence than F-SGD and NSGD for k values and achieves minimum RMSE (0.989) for f_r = 0.3 with k = 30. Convergence of NF-SGD is also compared with standard counterparts for f_r = 0.6 and f_r = 0.9 in Figs. 4.18 and 4.19 respectively. Moreover, considerably faster NF-SGD convergence for f_r = 0.6 than f_r = 0.3 over counterparts is also seen in Fig.

4.18 (a) and (b) for k variations. NF-SGD obtained minimum RMSE (0.981) for f_r = 0.6 with k = 30. The fastest convergence of NF-SGD is observed in Fig. 4.19 (a) and (b) for f_r = 0.9 with k variants than in Figs. 4.17 and 4.18 with f_r = 0.3 and f_r = 0.6 respectively. The proposed NF-SGD method accomplishes minimum RMSE (0.976) for k = 30 and f_r = 0.9.

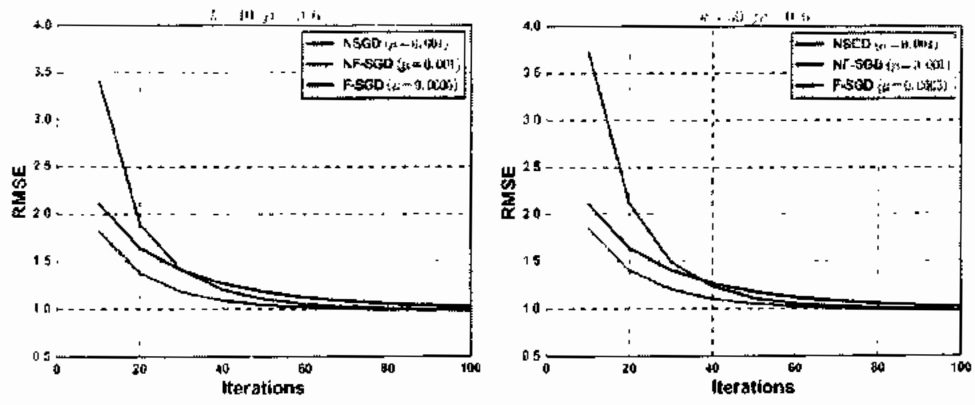
Analysis (ML-100K)

It is analyzed from the learning curves given in the Figs. 4.17, 4.18 and 4.19, that RMSE attained by the F-SGD and NSGD at different iterations is achieved by the NF-SGD in reasonably fewer iterations for different k variations. RMSE achieved by F-SGD for different f_r variations is obtained by proposed NF-SGD in slighty less iterations except for f_r = 0.9 for k = 20 and 30 where F-SGD achieves RMSE slightly better than NF-SGD. It is also noticed that with an increase in k, a slight increase in accuracy is achieved. It is seen that the proposed NF-SGD is accurate for all fractional order f_r values between 0 and 1. Moreover, faster convergence is achieved by NF-SGD with the increase in f_r value. The best convergence for NSGD is seen for μ = 0.001 and k = 30. Whereas, the fast convergence of F-SGD is observed for μ = 0.0005, k = 30 and f_r = 0.9. The paramount difference in convergence by the proposed NF-SGD with μ = 0.001, k = 30 and for all f_r variations, is noticed as compared to the best convergence of F-SGD and NSGD and is given in Fig. 4.20.



(a) (b)

Fig. 4.17 Learning curves for NSGD, F-SGD and NF-SGD with f_r = 0.3 for ML-100K Dataset



(a) (b)

Fig. 4.18 Learning curves for NSGD, F-SGD and NF-SGD with f_r = 0.6 for ML-100K Dataset

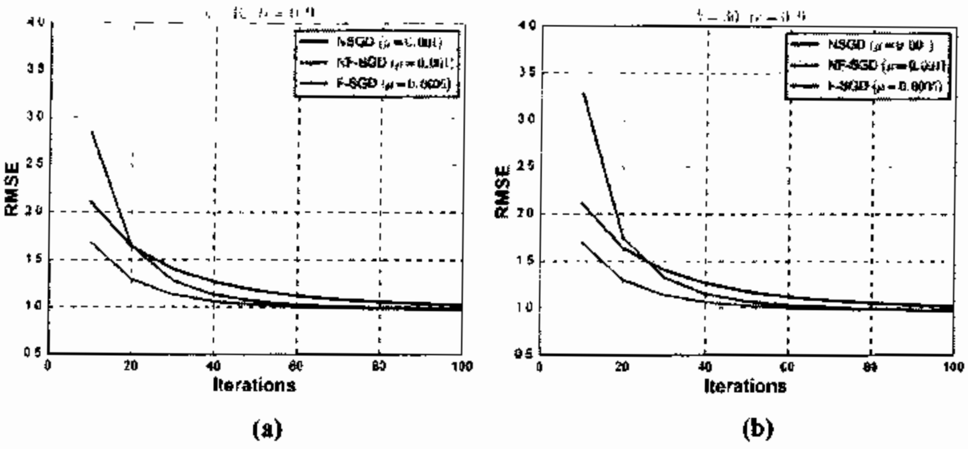


Fig. 4.19 Learning curves for NSGD, F-SGD and NF-SGD with f_r = 0.9 for ML-100K Dataset

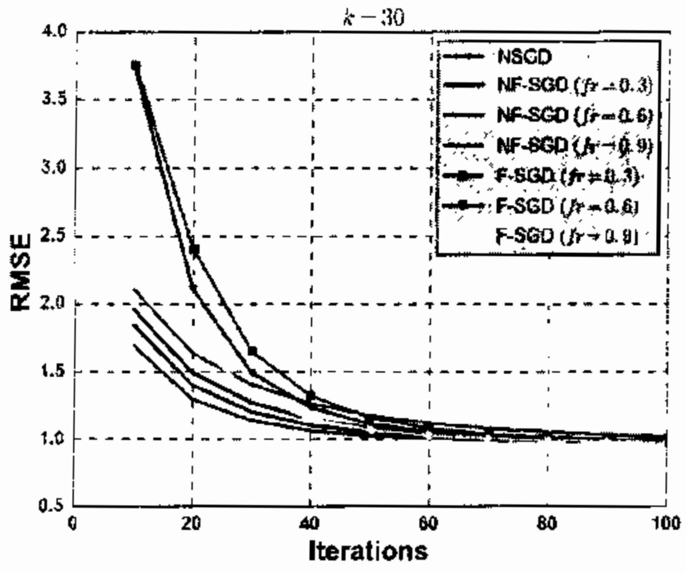


Fig. 4.20 Convergence comparison of NSGD, F-SGD and NF-SGD against f_r variations for ML-100K Dataset

Table 4.10: Simulation parameters

Parameters	**Symbol**	**Values used for tuning**	**Optimal values used by methods to achieve Min RMSE**		
			F-SGD	**NSGD**	**NF-SGD**
Learning rate	μ	0.0001, 0.0005, 0.001, 0.005, 0.01	0.001	0.0005	0.001
Fractional Learning rate	μ_{f_r}	0.0001, 0.0005, 0.001, 0.005, 0.01	0.001	0.0005	0.001
Fractional order	f_r	0.3, 0.6, 0.9	0.9	----	0.9
Features	k	10, 20, 30	30	30	30

Discussion (ML-1M) To validate the performance (RMSE) of the proposed scheme, NF-SGD is further analyzed for bigger dataset i.e., ML-1M, and the comparison of proposed NF-SGD with standard F-SGD and NSGD for ML-1M dataset is given in Figs. 4.21, 4.22, 4.23 and 4.24. The performance evaluation of proposed algorithm with standard counterparts for f_r = 0.3 and k (10 and 30) variations is presented in Fig. 4.21 (a) and (b). Similarly, an assessment in terms of RMSE of the NF-SGD for f_r = 0.6 is shown in Fig. 4.22 (a) and (b), while the respective performance curves in case of f_r = 0.9 are specified in Figures 4.23a and 4.23b. The identical convergence behavior of the proposed NF-SGD is noticed for ML-1M as compared to F-SGD and NSGD in case of ML-100k dataset. It is observed that convergence of NF-SGD is faster than counterparts i.e., F-SGD and NSGD, and by increasing the f_r value NF-SGD provides faster convergence. It is also seen that NF-SGD outperform other methods in terms of convergence for all k variations, while relatively better RMSE is achieved for bigger k values for 100 runs.

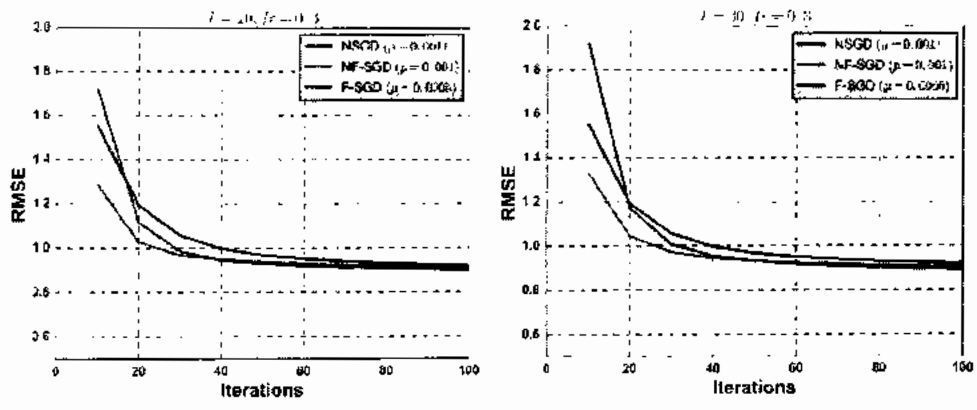
Table 4.11: Convergence comparison of NF-SGD with NSGD and F-SGD w.r.t RMSE achieved against particular iterations for ML-100k dataset with optimal learning rates

Method	k	f_r	Epochs									
			20	40	60	80	100	120	140	160	180	200
NSGD	10	--	1.639	1.267	1.12	1.053	1.02	1.003	0.993	0.986	0.982	0.978
F-SGD	10	0.3	2.118	1.283	1.078	1.017	0.995	0.985	0.98	0.979	0.979	0.98
F-SGD	10	0.6	1.89	1.201	1.049	1.005	0.988	0.981	0.979	0.979	0.98	0.981
F-SGD	10	0.9	1.643	1.13	1.025	0.994	0.982	0.978	0.977	0.978	0.98	0.982
NF-SGD	10	0.3	1.460	1.125	1.031	1.002	0.988	0.980	0.975	0.971	0.969	0.968
NF-SGD	10	0.6	1.374	1.087	1.016	0.994	0.983	0.976	0.971	0.968	0.966	0.966
NF-SGD	10	0.9	1.292	1.060	1.005	0.988	0.978	0.972	0.967	0.964	0.964	0.964
NSGD	20	--	1.638	1.265	1.118	1.051	1.019	1.002	0.992	0.985	0.981	0.977
F-SGD	20	0.3	2.389	1.354	1.101	1.025	0.997	0.983	0.976	0.973	0.972	0.974
F-SGD	20	0.6	2.087	1.247	1.061	1.007	0.986	0.976	0.972	0.971	0.973	0.976
F-SGD	20	0.9	1.733	1.148	1.028	0.991	0.976	0.969	0.967	0.969	0.973	0.979
NF-SGD	20	0.3	1.457	1.125	1.029	0.999	0.987	0.979	0.974	0.970	0.967	0.965
NF-SGD	20	0.6	1.375	1.087	1.016	0.994	0.983	0.976	0.971	0.967	0.965	0.964
NF-SGD	20	0.9	1.291	1.058	1.005	0.987	0.978	0.971	0.966	0.962	0.961	0.960
NSGD	30	--	1.638	1.266	1.119	1.052	1.019	1.002	0.992	0.985	0.98	0.976
F-SGD	30	0.3	2.405	1.323	1.08	1.014	0.991	0.98	0.973	0.969	0.966	0.966
F-SGD	30	0.6	2.107	1.235	1.053	1.003	0.983	0.974	0.968	0.966	0.966	0.968
F-SGD	30	0.9	1.748	1.15	1.028	0.991	0.974	0.966	0.962	0.964	0.968	0.974
NF-SGD	30	0.3	1.493	1.151	1.042	1.005	0.989	0.980	0.973	0.969	0.965	0.962
NF-SGD	30	0.6	1.401	1.103	1.021	0.994	0.981	0.973	0.968	0.964	0.960	0.958
NF-SGD	30	0.9	1.298	1.063	1.005	0.986	0.976	0.969	0.964	0.960	0.957	0.956

The performance of suggested NF-SGD and other counterparts is also assessed for initial convergence rates for ML-1M dataset and results for changes in fitness for first 100 iterations are shown in Table 4.12 for various f_r and k values. It is observed from the results that initial convergence of proposed NF-SGD is much rapid than standard adaptive methods and convergence rate increases with the increase in fractional order f_r value. The RMSE attained by NF-SGD with μ = 0.001, k = 30 and f_r = 0.9 after 100 runs is (0.886). Whereas, F-SGD accomplished RMSE of (0.887) with μ = 0.0005, k = 30 and f_r = 0.9 after 100 runs. However, the best RMSE value after convergence achieved by NSGD after 100 iterations with μ = 0.001, k = 30 is (0.920).

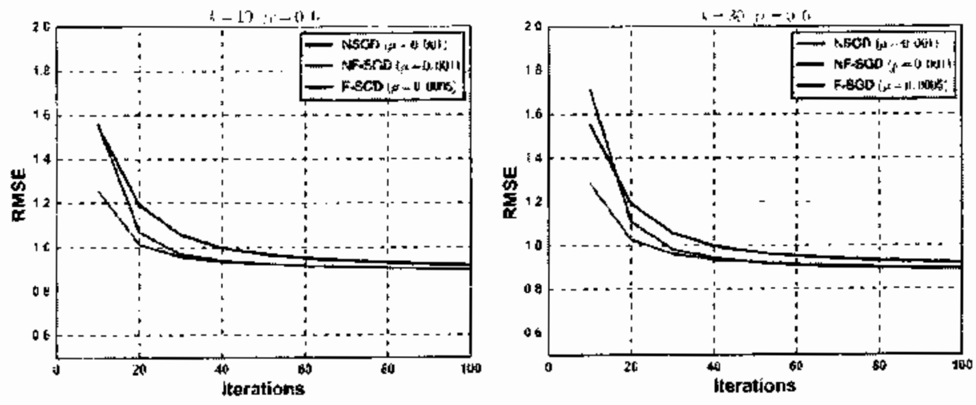
Analysis (ML-1M)

It is explored that all methods discussed in Case-Study-III are accurate and convergent using ML-1M dataset, but comparatively superior convergence is achieved by proposed NF-SGD for higher values of fractional order (f_r) and more number of features (k). Furthermore, it is perceived that NF-SGD achieves better steady state performance for lower fractional order (f_r) values. It is also seen that proposed NF-SGD remains stable for all fractional order (f_r) values and a slight difference in accuracy is noticed amongst different fractional order variations. To recognise convergence behaviour of NF-SGD and to differentiate the performance in terms of fast convergence among three methods i.e., (NSGD, F-SGD and NF-SGD) with optimal learning rates, all fractional orders and maximum number of features i.e., k = 30 for ML-1M dataset, the accomplished results by the methods are presented in the Fig. 4.24.



(a) (b)

Fig. 4.21 Learning curves for NSGD , F-SGD and NF-SGD with f_r = 0.3 for ML-1M Dataset



(a) (b)

Fig. 4.22 Learning curves for NSGD, F-SGD and NF-SGD with f_r = 0.6 for ML-1M Dataset

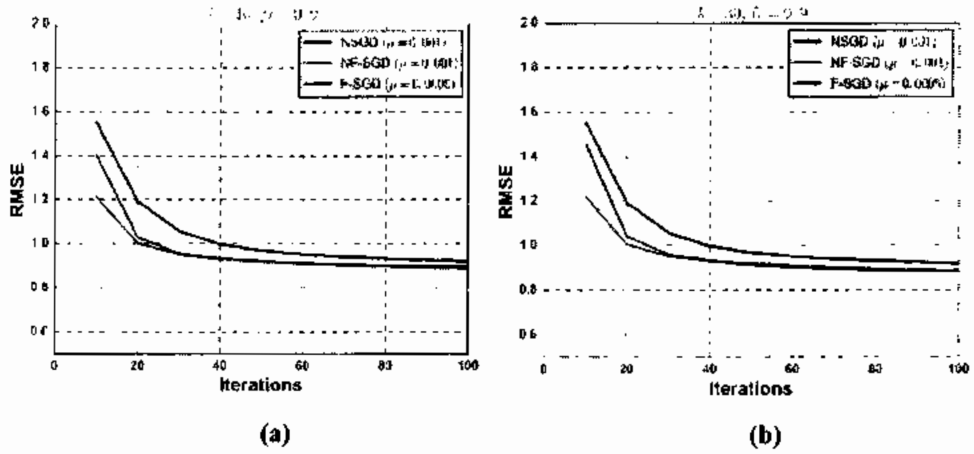


Fig. 4.23 Learning curves for NSGD , F-SGD and NF-SGD with f_r = 0.9 for ML-1M Dataset

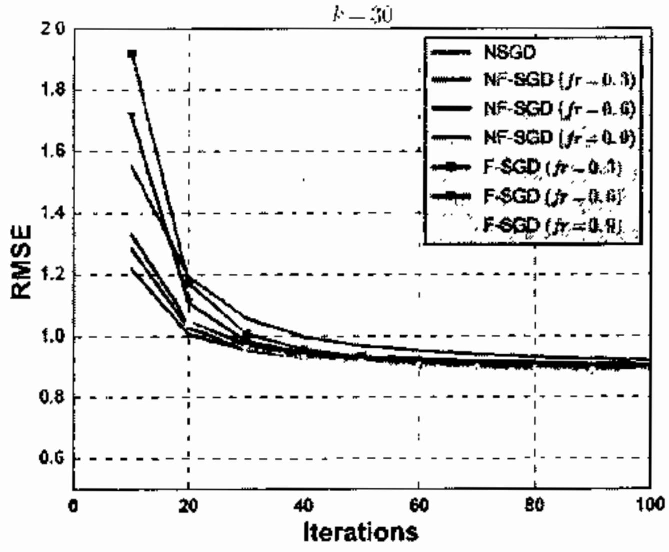


Fig. 4.24 Convergence comparison of NSGD , F-SGD and NF-SGD against f_r variations for ML-1M Dataset

Table 4.12: Convergence comparison of NF-SGD with NSGD and F-SGD w.r.t RMSE achieved against particular iterations for ML-1M dataset with optimal learning rates

Method	k	f_r	Epochs									
			10	20	30	40	50	60	70	80	90	100
NSGD	10	--	1.553	1.193	1.058	0.998	0.969	0.952	0.94	0.932	0.925	0.92
F-SGD	10	0.3	1.719	1.116	0.985	0.946	0.93	0.921	0.915	0.911	0.906	0.903
F-SGD	10	0.6	1.563	1.069	0.969	0.938	0.925	0.917	0.911	0.906	0.902	0.899
F-SGD	10	0.9	1.403	1.03	0.956	0.932	0.92	0.912	0.906	0.901	0.897	0.895
NF-SGD	10	0.3	1.287	1.027	0.968	0.948	0.938	0.93	0.924	0.919	0.915	0.911
NF-SGD	10	0.6	1.258	1.014	0.957	0.937	0.925	0.918	0.912	0.907	0.902	0.898
NF-SGD	10	0.9	1.212	1.002	0.952	0.93	0.919	0.909	0.903	0.897	0.892	0.889
NSGD	20	--	1.553	1.193	1.058	0.998	0.969	0.952	0.941	0.933	0.926	0.921
F-SGD	20	0.3	1.849	1.160	1.002	0.950	0.928	0.917	0.909	0.903	0.899	0.896
F-SGD	20	0.6	1.656	1.097	0.977	0.939	0.921	0.911	0.904	0.899	0.895	0.893
F-SGD	20	0.9	1.431	1.037	0.956	0.929	0.914	0.904	0.898	0.894	0.891	0.890
NF-SGD	20	0.3	1.331	1.046	0.975	0.952	0.939	0.931	0.924	0.919	0.914	0.909
NF-SGD	20	0.6	1.29	1.029	0.963	0.938	0.925	0.916	0.909	0.903	0.898	0.894
NF-SGD	20	0.9	1.219	1.006	0.954	0.932	0.92	0.911	0.903	0.897	0.892	0.888
NSGD	30	--	1.553	1.193	1.058	0.998	0.968	0.951	0.94	0.932	0.925	0.92
F-SGD	30	0.3	1.921	1.172	1.008	0.956	0.932	0.919	0.909	0.902	0.897	0.893
F-SGD	30	0.6	1.713	1.108	0.983	0.943	0.923	0.911	0.903	0.897	0.892	0.889
F-SGD	30	0.9	1.457	1.043	0.96	0.931	0.915	0.904	0.896	0.89	0.888	0.887
NF-SGD	30	0.3	1.33	1.046	0.973	0.945	0.932	0.924	0.918	0.913	0.909	0.905
NF-SGD	30	0.6	1.284	1.026	0.961	0.935	0.922	0.915	0.908	0.903	0.898	0.894
NF-SGD	30	0.9	1.22	1.006	0.953	0.931	0.919	0.909	0.902	0.896	0.89	0.886

Performance comparison of recent Matrix Factorization based methods:

To prove the worth of the proposed technique, we have also compared the performance of NF-SGD with few recent matrix factorization-based approaches apart from other contending approaches of NSGD and FSGD for **ML-100K** and **ML-1M datasets**. Results of each matrix factorization-based method are stated with best hyper-parameter settings. Brief overview of the baselines presented in [111] and [113] is as follows:

- **OLR** [111]: OLR is an Online Low-rank approximation method which optimizes the loss function directly [114] by learning a rank-k matrix factorization through online gradient descent.
- **ConvMF** [111] ConvMF is a context-aware recommendation model which combines CNN into PMF to capture contextual information of documents to improve ratings prediction accuracy [112].
- **SOCF_II** [111]: SOCF_II is a second-order Sparse OCF method that adds an absolute term to objective function to deal with user-item ratings for online collaborative filtering [115].
- **DBPMF** [111]: Deep Bias Probabilistic Matrix Factorization model is suggested by applying the CNN to extract latent user/item features and adding the bias into probabilistic MF to track user rating behavior and item popularity.
- **DCBPMF** [111]: Deep Constrain Bias PMF model is proposed to improve the performance of DBPMF model further by constraining the user-specific and item-specific vectors.

- **RDMC** [113]: RDMC is a Robust Discrete Matrix Completion method that predicts from the collection of user specified discrete values by introducing a new discrete constraint to the matrix completion model [116].
- **ODMC** [113]: ODMC is an Optimal Discrete Matrix Completion method which automatically learns optimal thresholds and also ensures an exact low-rank structure of the target matrix. [117].
- **BMF-D** [113]: Discrete Basic Matrix Factorization model is achieved from the basic matrix factorization model (BMF) that permits gradient based techniques to jointly learn both the matrix factorization model and a discretization operator.
- **DMF-D** [113]: Discrete Deep Matrix Factorization model is attained from the deep matrix factorization model (DMF) [118][119] that provides deep learning based approaches an opportunity to learn mutually both the matrix factorization model and a discretization operator.

Table 4.13 demonstrates a comparison between performance (RMSE) of the proposed NF-SGD model with MF based models for recommender systems presented in [111] for ML-100K and ML-1M datasets. It is perceived from Table 4.13 that the proposed NF-SGD achieves significant difference in RMSE when compared to other MF based counterparts. It is also noticed from Tables 4.11 and 4.12 that NF-SGD outperforms MF techniques given in case study III for all k and f_r variations. NF-SGD achieves best performance of RMSE = 0.956 and 0.886 with k = 30 and f_r = 0.9 for ML-100K and ML-1M respectively.

To validate the efficacy of the proposed model, performance (RMSE) of NF-SGD is further evaluated with other discrete matrix factorization-based models of **RDMC,**

ODMC, **BMF-D** and **DMF-D** [113] for recommender systems. The performance comparison in terms of RMSE values is presented in Table 4.14. It is observed from the results that NF-SGD accomplished more improvement than **RDMC, ODMC** and **BMF-D** models for all k and f_r values and achieves finest performance of RMSE = 0.956 with k = 30 and f_r = 0.9 for ML-100K dataset. Whereas, the performance of NF-SGD is comparable with **DMF-D** for k = 30 and f_r = 0.9 on ML-100K.

Table 4.13: Performance comparison of NF-SGD with MF methods for ML-100K and ML-1M datasets

DATASETS	METHODS	RMSE
ML-100K	OLR	1.048
	ConvMF	1.000
	SOCF_II	0.995
	DBPMF	0.990
	DCBPMF	0.985
	NF-SGD	**0.956**
ML-1M	OLR	1.038
	ConvMF	0.980
	SOCF_II	0.969
	DBPMF	0.945
	DCBPMF	0.943
	NF-SGD	**0.886**

As presented in Table 4.14, NF-SGD achieves substantial increase in performance as compared to the existing methods for ML-1M dataset as well. Our proposed NF-SGD outperforms **RDMC, ODMC** and **BMF-D** for different f_r and k variations. The supreme performance of RMSE = 0.886 of NF-SGD is accomplished with k = 30 and f_r = 0.9 for ML-1M dataset. For ML-1M dataset, a slight increase in performance of NF-SGD compared to **DMF-D** is observed for all k variations with f_r = 0.6 but NF-SGD outperforms **DMF-D** by large margin with f_r = 0.9. Such considerable behavior confirms the effectiveness of NF-SGD for providing accurate recommendations.

Table 4.14: Performance comparison of NF-SGD with discrete MF methods for ML-100K and ML-1M datasets

DATASETS	METHODS	RMSE
ML-100K	RDMC	0.971
	ODMC	0.968
	BMF-D	0.968
	DMF-D	0.941
	NF-SGD	**0.956**
ML-1M	RDMC	0.987
	ODMC	0.937
	BMF-D	0.921
	DMF-D	0.898
	NF-SGD	**0.886**

4.5 Summary

In this chapter, three Case-Studies have been presented to compare the performance in terms of RMSE of proposed fractional order based learning machines with standard SGD based strategies for recommender systems. It is shown that proposed FSGD improves accuracy and convergence of recommender systems. However, proposed mF-SGD further improves the convergence speed of FSGD through utilizing gradients previous information. As adaptive nature of learning rate is not explored in FSGD and mF-SGD. Therefore, NF-SGD is proposed to examine the adaptive nature of learning rate for smooth convergence in FSGD.

The subsequent Chapter provides the research methodology with deep learning based strategy for providing Top-N recommendations. The performance of suggested learning machine is also verified through ML-100k and ML-1M datasets.

Chapter 5. **Deep Learning Machines for Top-N Recommender Systems**

To promote online businesses and sales, e-commerce industry focuses on fulfilling users' demands by giving them top set of recommendations which are ranked through different ranking measures. Deep learning based auto-encoder models have further shown improved performance in terms of giving Top-N recommendations. Therefore, in this chapter first deep learning based state-of-the-art auto-encoder models are presented for providing Top-N recommendations, then a users' rating-trend based denoising auto-encoder for Top-N recommendations is proposed.

This chapter primarily includes introduction to deep learning based recommendation models, then deep learning based standard auto-encoder models for recommender systems are presented. Lastly, the proposed users' rating-trend based denoising auto-encoder model is presented and the performance comparison of the suggested learning machine is made with standard deep learning based methods through simulations to prove the performance of proposed method for Top-N recommendations.

5.1 Deep Learning Approach

The objective of deep learning is to learn multiple levels of abstractions and representations from data. Different deep learning based neural designs have worked significantly for supervised as well as unsupervised problems [120]. Deep learning based recommendation models are mainly divided into two sub types [65] (1) Neural blocks based recommendation (2) Deep hybrid models based recommendation.

There are around nine important neural blocks based recommendation models also known as deep learning based models, that is, Auto-encoder (AE) [42][121][77][74][122], Restricted Boltzmann Machine (RBM) [123]–[127], Recurrent Neural Network (RNNs) [128]–[132], Multilayer Perceptron (MLP) [133]–[137], Convolutional Neural Network (CNNs) [112], [138]–[141], Neural Autoregressive Distribution Estimation (NADE) [142]–[144], Deep Reinforcement Learning (DRL) [145]–[149], Adversarial Networks (AN) [150]–[153] and Attention Models (AM) [154]–[158]. The applicability of the neural blocks based recommendation models is determined by the method employed in the recommendation model.

Deep hybrid models are the combination of two or more deep learning based neural models [137], [159]–[161]. The objective of deep hybrid models is to utilize the characteristics of two or more models in a single but more powerful neural model. Different combinations of nine deep learning models are possible to make an influential deep hybrid model [162]–[167] but all mixtures of deep learning models have not been explored yet.

5.1.1 Notations

We denote matrices with upper-case bold letters $(\text{e.g.}, \mathbf{M})$ and vectors are represented by a lower case bold italic letters$(\text{e.g.}, \boldsymbol{m})$. The x^{th} row of a matrix $\mathbf{M}$ is denoted by $\boldsymbol{M}_x$ and $(x,y)^{th}$ entry of a matrix is represented by $\boldsymbol{M}_{xy}$ whereas x^{th} element of a vector $\boldsymbol{m}$ is symbolized by $\boldsymbol{m}_x$. We treat u as user index and i as item index throughout this chapter.

In this chapter, users and items sets are denoted by $\wp = \{1, \dots, p\}$ and $\mathbb{Q} = \{1, \dots, q\}$ respectively. A set of observed user-item pairs is denoted by $\mathbb{Z} = \{u, i, z_{ui}\}$ where z_{ui} represents a non-zero rating (for implicit feedback setting, $z_{ui} = 1$) of a user u for item i. An unobserved set of user-item is given by $\overline{\mathbb{Z}}$. The user-item pairs dataset comprises of some information taken from a subset S_{un} of $\overline{\mathbb{Z}}$ is known as augmented dataset [28] represented by $\overline{\overline{\mathbb{Z}}}$. Such augmented dataset is used to avoid trivial solution by optimizing point-wise objective function for implicit feedback setting where all observed 'likes' are treated as "1". For a specific user u, the set of ratings in training data is designated as $\mathbb{Z}_u$, while a set of unobserved ratings of a user u is represented by $\overline{\mathbb{Z}}_u$. The items which are to be suggested to a user u are the ones present in $\overline{\mathbb{Z}}_u$. Therefore, a subset of items from $\overline{\mathbb{Z}}_u$ are suggested by a recommender system to a user for top-N recommendations. Due to a large number of users and items in rating matrix, we take a subset of unobserved preferences of a user u termed as S_{un} from unobserved ratings set of a user $\overline{\mathbb{Z}}_u$ and calculate gradients for the items in $S_{un} \subset \overline{\mathbb{Z}}_u$ though back-propagation rather than calculating gradients on all outputs. The S_{un} set is also described as set of negative items in [28]. In this chapter we are actually using implicit feedback setting $\{0,1\}$ instead of numerical feedback setting that lies in the range of $\{1,5\}$.

Before going into the details of proposed deep learning based denoising auto-encoder model for Top-N recommendations, mathematical description of deep learning based standard denoising auto-encoder models for recommender systems are presented in the next section.

5.2 Auto-encoders

Auto-encoders are applied to recommender systems for two purposes [65] (1) For learning non-linear feature representations in the bottle-neck layer (2) For filling the rating entries in the rating matrix straight in the output layer (reconstruction layer). Neurons in the input layer of auto-encoders are exactly same in number to the reconstruction layer neurons. Different variants of auto-encoders include denoising auto-encoder [156][43], variational auto-encoder [168][169], marginalized auto-encoder [121] and contractive auto-encoder [170]. The architecture including mathematical details of a basic auto-encoder is already discussed in Chapter 2.

Furthermore, two variations of a basic auto-encoder such as denoising auto-encoder and collaborative denoising auto-encoder are described in the following two subsections.

5.2.1 Denoising Auto-encoder (DAE)

One of the variations of a basic auto-encoder is the denoising auto-encoder (DAE) [171]. In simple auto-encoder, network is trained to reconstruct input $\boldsymbol{t}$ at the output layer $\hat{\boldsymbol{t}}$, by exploiting the information learned in the hidden layer. Whereas, DAE is trained to reconstruct $\boldsymbol{t}$ from partially corrupted form of the input $\bar{\boldsymbol{t}}$. Denoising property of DAE, makes it more robust than basic variant since it also has ability to deal with corrupted forms of data [172]. For recommender systems, this models sparse input as a corrupted input and reconstructing the clean output helps in learning latent space of data at the hidden layer (bottle-neck layer). The network of DAE is graphically shown in Fig 5.1.

In DAE, input $\boldsymbol{t}$ is initially corrupted to get a partially corrupted form $\bar{\boldsymbol{t}}$ by using the concept of stochastic mapping $\bar{\boldsymbol{t}} \sim q(\bar{\boldsymbol{t}}|\boldsymbol{t})$ [171]. The corrupted form of the input is randomly drawn from a conditional distribution $p(\bar{t}|t)$. The standard input corruption options are additive Gaussian noise $p(\bar{t}|t) = \mathbb{N}(t, \Sigma)$ (where covariance matrix is not dependent on t) and mask-out corruption (where every dimension of t is randomly overwritten by 0 with a probability of q) [28][173].

$$P(\bar{t} = 0) = q \quad (5.1)$$

$$\epsilon = P(\bar{t} = 1/(1-q)t) = 1 - q \quad (5.2)$$

For the sake of making corruption unbiased, non-corrupted entries are set by a factor of $1/(1-q)$ which is a function of the original input value.

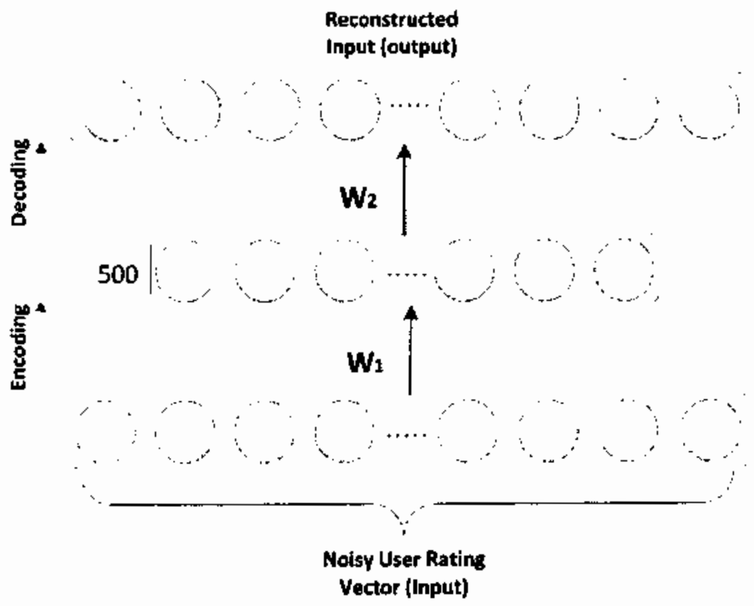

Reconstructed Input (output)
Decoding
W₂
500
Encoding
W₁
Noisy User Rating Vector (Input)

Fig. 5.1 Network diagram of standard DAE for top-N recommender systems

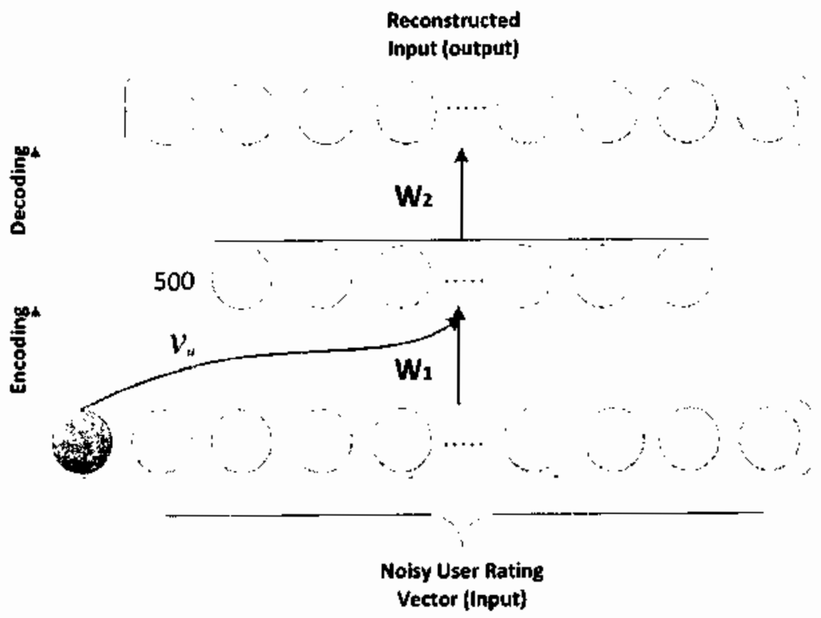
5.2.2 Collaborative Denoising Auto-encoder (CDAE)

The DAE is extended in [28] to a more improved form of collaborative filtering by introducing a user oriented node in the input layer of the denoising auto-encoder, known as collaborative denoising auto-encoder (CDAE). The idea of CDAE is to recover correlations among user and items through a corrupted version of given binary feedback. The difference between DAE and CDAE lies in the additional encoding vector $\boldsymbol{v}_u \in \mathbb{R}^K$ (user oriented) at the input layer for a user. CDAE proved to be an improved model than DAE for top-N recommendations. The network design of CDAE is given in Fig 5.2.

The hidden layer of CDAE with Sigmoid mapping function is represented as:

$$\mathcal{H}(\boldsymbol{t}) = Sigmoid(\mathbf{W}_1^{\mathrm{T}}\bar{\boldsymbol{t}} + \boldsymbol{v}_u + \boldsymbol{b}_{\mathcal{H}}) = \sigma(\mathbf{W}_1^{\mathrm{T}}\bar{\boldsymbol{t}} + \boldsymbol{v}_u + \boldsymbol{b}_{\mathcal{H}}) \tag{5.3}$$

Where, $\boldsymbol{b}_{\mathcal{H}} \in \mathbb{R}^K$, is the bias of hidden layer and matrix $\mathbf{W}_1 \in \mathbb{R}^{d\times K}$ denotes weights of input layer connected with hidden layer.



Reconstructed Input (output)
Decoding
W₂
500
Encoding
V_u
W₁
Noisy User Rating Vector (Input)

Fig. 5.2 Network design of CDAE for top-N recommender systems

Where $\boldsymbol{v}_u \in \mathbb{R}^K$ is the randomly generated Gaussian noise.The representation of output layer using sigmoid activation function for CDAE is similar to the AE, which is denoted as:

$$\hat{\boldsymbol{t}} = Sigmoid(\mathbf{W}_2^{\mathrm{T}}\mathcal{H}(\boldsymbol{t}) + \boldsymbol{b}_O) = \sigma(\mathbf{W}_2^{\mathrm{T}}\mathcal{H}(\boldsymbol{t}) + \boldsymbol{b}_O) \tag{5.4}$$

Where, $\boldsymbol{b}_O \in \mathbb{R}^d$, denotes bias of the output layer and matrix $\mathbf{W}_2 \in \mathbb{R}^{K\times d}$ represents weights linked between hidden layer and output layer.

The average reconstruction loss turn out to be expected average loss [28][173] using the condition $p(\bar{\boldsymbol{t}}|\boldsymbol{t})$. Therefore, loss function of CDAE for the training of parameters $\Phi = \{\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{v}_u, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_O\}$ is denoted as:

$$\min_{\Phi} \frac{1}{p}\sum_{u=1}^{p} \mathbb{E}_{p(\bar{t}|t)}\,[\mathcal{L}(\boldsymbol{t},\hat{\boldsymbol{t}})] + \Re\,(\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{v}_u\,, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_O) \tag{5.5}$$

Here, $\mathbb{E}_{p(\bar{t}|t)}\,[\mathcal{L}(\boldsymbol{t},\hat{\boldsymbol{t}})]$ represents expected average squared loss and $\Re$ is the regularization term that includes squared ℓ_2 norm of the training parameters represented as:

$$\Re\,(\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{v}_u\,, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_O) = \frac{\lambda}{2}(\|\mathbf{W}_1\|_2^2 + \|\mathbf{W}_2\|_2^2 + \|\boldsymbol{v}_u\|_2^2 + \|\boldsymbol{b}_{\mathcal{H}}\|_2^2 + \|\boldsymbol{b}_O\|_2^2) \tag{5.6}$$

As it is already proved in [28] that performance of CDAE in terms of mean average precision improves after adding one unique user oriented weight vector to the encoding layer. But that randomly generated specific vector is added to encoding layer for all input user vectors irrespective of users rating behavior. This provides an opportunity to use weight vectors according to users' rating-trend Therefore, in the next section a novel user's rating-trend based model is proposed. In our proposed design one user oriented weight vector is added to input layer for low rating-trend of a user as evidenced in [28] and two weight vectors are added for high rating-trend of a user.

5.3 User's Rating-Trend based Collaborative Filtering

Apart from modelling user vectors as corrupted inputs, an important aspect is to encode the rating trend of users. To learn latent features of users, a user who has a behavior of giving higher ratings should be learned. Similarly, for users having low rating trend should also be learned. This will help to generalize the recommendations for those items which don't have any recommendation for a user. In our proposed model, this rating-trend of users is incorporated using two additional nodes vectors in a basic DAE architecture. One of the node vectors encodes low-valued ratings and other encodes high-valued ratings. For a user with low-valued rating trend, low-trend user node is activated while for high-valued rating trend, high trend user node along with low-valued user node is activated. These user's rating trend encoding nodes can also be visualized as weight vectors encoding rating-trend of users. Users with high valued ratings are given more weights while learning their latent features.

Brief discussion on User's Rating-trend:

The rating-trend of a user actually exhibits the rating patterns of a user to indicate the preference of a user for a set of items. Rating patterns can be categorized into low and high rating patterns. If a user is habitual of giving high ratings for the items of his interest, then this habit of a user reflects the high rating-trend of a user for a set of items. Whereas, rating-trend of a user is said to be low, if a user frequently gives low ratings for the items. In order to calculate the actual rating-trend of a user, a threshold value is required to check whether the accumulative rating-trend of a user is greater or lesser than the threshold value (e.g. Threshold value = 30% of the total items set). The value of threshold plays a role in determining the overall performance of the proposed method. Lower threshold values improve the proposed model by activating both user oriented nodes of the UT-CDAE. However, the overall performance trend will remain same for different input corruption levels. The proposed UT-CDAE captures the rating-trend of a user by counting the ratings (number of 1's against numerical ratings 4 and 5) rated by a user in a rating matrix for a set of items through a binary feedback as given in Table 5.1. If the number of 1's are less than the decided threshold value then rating-trend of a user is said to be low otherwise rating-trend of a user is known as high rating-trend. The benefit of modelling rating-trend for Top-N recommendations is to recommend an unrated set of items to the user by approximating the dominating preference behavior (ratings) of a user for a rated set of items. The process of computation of user's rating-trend for a set of items through an intuitive example is given in Table 5.1.

Table 5.1: User's rating-trend calculation from binary feedback

	ITEMS											Val_{th} = **Threshold Value (30% of Items Set = 3)**
		I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	If 1's count greater than Val_{th} = High Trend otherwise Low Trend
USERS	U_1	1		1		0		0		1	1	High rating-trend
	U_2	1		0		0			1			Low rating-trend
	U_3		1		1		0			0	0	Low rating-trend
	U_4	0		0		0	1			1	0	Low rating-trend
	U_5		1	0	1		0		1		1	High rating-trend

By exploiting the concept of CDAE, we propose an additional value-added strategy for collaborative filtering by using the rating-trend of users. We call such variation as user's rating-trend based collaborative denoising auto-encoder (UT-CDAE). This idea of encoding user's rating trends in UT-CDAE has outperformed other denoising based methods (DAE and CDAE) while predicting top-N recommendations as given in simulations and results section of this chapter. The network architecture and mathematical details of proposed UT-CDAE are presented in the next sub-section.:

5.3.1 User's Rating-Trend based CDAE (UT-CDAE)

The network of the proposed learning machine is as follows:

Network Design/Architecture: The UT-CDAE network comprises of encoding and decoding layers. The encoding layer maps the data inputs from the input layer to the hidden layer. Decoding layer maps the encoded representation of data from hidden to output layer. An illustration of the network of UT-CDAE is presented in Fig 5.3.

The input layer of UT-CDAE consists of $iln + 2$ nodes, where iln denotes input layer nodes representing a user's preference for set of items. Each node in iln represents an item in $\mathbb{Z}_u$. The left and the right most nodes in the input layer excluding the iln nodes are distinct for each user and termed as "user rating-trend nodes". We call the left most node as **High-Trend-User-Node** ($HTUN$) and the right most node is known as **Low-Trend-User-Node** ($LTUN$). Weights connected to both nodes, i.e., ($HTUN$ *and* $LTUN$) are different and uniquely linked to preferences of each user($u \in \wp$).

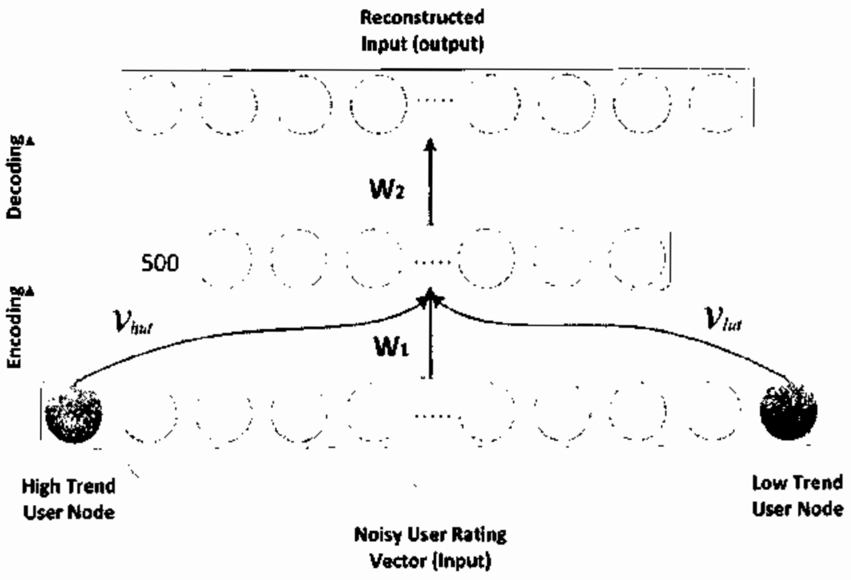


Fig. 5.3 Network architecture of proposed UT-CDAE for top-N recommender systems

The training set formed for the observed user-item interactions $\mathbb{Z}$ consists of p user vectors $\{t_1, t_2, t_3, \dots, t_p\}$, where $t_u = \{t_{u1}, t_{u2}, t_{u3}, \dots, t_{uiln}\}$ denotes a sparse, iln dimensional user's (u) preference based binary vector that merely contains the observed non-zero entries for items in the items set $\mathbb{Q}$, the value t_{ui} depends upon rating values of a user u for an item i. If a user does not like an item i then $t_{ui} = 0$, otherwise $t_{ui} = 1$.

The hidden layer is composed of K nodes which are fully connected and smaller in dimension than the input layer. In addition to the K nodes, there is an additional bias node present in the hidden layer, which is fully connected to the output layer and acts as an offset. The weight matrix between the input layer nodes iln and the hidden layer nodes K is represented by $\mathbf{W}_1 \in \mathbb{R}^{iln \times K}$. We represent the weight vectors of $HTUN$ and $LTUN$ associated with hidden layer nodes respectively as $\boldsymbol{v}_{hut} \in \mathbb{R}^K$ and $\boldsymbol{v}_{lut} \in \mathbb{R}^K$.

The output layer represents reconstructions of the input $\boldsymbol{t}_u$ with iln nodes. The output layer is densely connected with the hidden layer nodes through a weight matrix represented as $\mathbf{W}_2 \in \mathbb{R}^{iln \times K}$. As the dimension of both the input and output layers is iln, therefore, the weight vector associated with the bias node in hidden layer is denoted by $\boldsymbol{b}_1 \in \mathbb{R}^{iln}$. The input $\bar{\boldsymbol{t}}_u$ applied at the input layer of UT-CDAE is a corrupted version of the original input $\boldsymbol{t}_u$. The corruptions are incorporated randomly in the input vector using masking noise $\boldsymbol{\epsilon}$ as given in equation (5.2).

Problem formulation: Given $u \in \wp$ user vectors, the average reconstruction error w.r.t user's rating-trend is minimized through learning of parameters $\Phi = \{\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{v}_{hut}, \boldsymbol{v}_{lut}, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_O\}$. Hence, the expected average loss for UT-CDAE is given as:

$$\min_{\Phi} \frac{1}{p} \sum_{u=1}^{p} \mathbb{E}_{p(\bar{\boldsymbol{t}}_u | \boldsymbol{t}_u)} \left[\mathcal{L}(\boldsymbol{t}_u, \hat{\boldsymbol{t}}_u)\right] + \Re_{UT-CDAE} \tag{5.7}$$

We use cross-entropy loss as reconstruction loss because cross-entropy loss is appropriate for binary inputs. The reconstruction through cross-entropy loss ($\mathcal{L}$) is computed as:

$$\mathcal{L}(\boldsymbol{t}_u, \hat{\boldsymbol{t}}_u) = -\boldsymbol{t}_u{}^T \log(\hat{\boldsymbol{t}}_u) - (1 - \boldsymbol{t}_u)^T \log(1 - \hat{\boldsymbol{t}}_u) \tag{5.8}$$

Where, $\Re_{UT-CDAE}$ is the regularization term involved in average reconstruction loss of UT-CDAE represented as:

$$\begin{aligned}\Re_{UT-CDAE} = \gamma\Re_1 \left(\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{v}_{lut}, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_O\right) + \\ (1-\gamma)\Re_2 \left(\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{v}_{hut}, \boldsymbol{v}_{lut}, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_O\right)\end{aligned} \tag{5.9}$$

To handle the complexity of the proposed method, the regularization terms $\Re_1$ and $\Re_2$ in the expected average loss for higher and lower rating trend of a user are based on the trend value denoted by γ. For a higher rating-trend of a user, $\gamma = 0$ otherwise $\gamma = 1$. We compute squared l_2 norm of the parameters in the regularization terms as follows:

$$\Re_1(\cdot) = \frac{\lambda_1}{2}(\|\mathbf{W}_1\|_2^2 + \|\mathbf{W}_2\|_2^2 + \|\boldsymbol{v}_{lut}\|_2^2 + \|\boldsymbol{b}_{\mathcal{H}}\|_2^2 + \|\boldsymbol{b}_O\|_2^2) \tag{5.10}$$

$$\Re_2(\cdot) = \frac{\lambda_2}{2}(\|\mathbf{W}_1\|_2^2 + \|\mathbf{W}_2\|_2^2 + \|\boldsymbol{v}_{hut}\|_2^2 + \|\boldsymbol{v}_{lut}\|_2^2 + \|\boldsymbol{b}_{\mathcal{H}}\|_2^2 + \|\boldsymbol{b}_O\|_2^2) \tag{5.11}$$

Here parameters $\Phi = \{\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{v}_{hut}, \boldsymbol{v}_{lut}, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_O\}$ are updated through stochastic gradient descent. The stepwise pseudocode of the proposed method is given in Algorithm 5.1.

The learning of UT-CDAE involves projection of the input $\bar{\boldsymbol{t}}_u$ to hidden layer to discover the hidden representations. For a high user's rating-trend for observed items ($\gamma = 0$) i.e. (number of binary 1's representing numerical ratings of 4 and 5 in a user vector are more in number), $\mathcal{H}(\boldsymbol{t})$ is computed as:

$$\mathcal{H}(\boldsymbol{t}) = f\left(\mathbf{W}_1^{\mathrm{T}}\bar{\boldsymbol{t}}_u + \boldsymbol{v}_{hut} + \boldsymbol{v}_{lut} + \boldsymbol{b}_1\right) \tag{5.12}$$

However, if there is a low rating-trend of a user, then γ is equal to 1 and $\mathcal{H}(\boldsymbol{t})$ is evaluated as follows:

$$\mathcal{H}(\boldsymbol{t}) = f\left(\mathbf{W}_1^{\mathrm{T}}\bar{\boldsymbol{t}}_u + \boldsymbol{v}_{lut} + \boldsymbol{b}_1\right) \tag{5.13}$$

For reconstruction of the input, the hidden representation $\mathcal{H}(\boldsymbol{t})$ is again mapped to the output layer (O) to reconstruct the clean input without corruption. After applying $Sigmoid$ as activation function at the output layer, reconstructed value for the i^{th} node is solved as:

$$\hat{\boldsymbol{t}}_{ui} = g(\mathbf{W}_2^{\mathrm{T}}\mathcal{H}(\boldsymbol{t}) + \boldsymbol{b}_i) \tag{5.14}$$

$$\hat{\boldsymbol{t}}_{ui} = Sigmoid(\mathbf{W}_2^{\mathrm{T}}\mathcal{H}(\boldsymbol{t}) + \boldsymbol{b}_i) = \sigma(\mathbf{W}_2^{\mathrm{T}}\mathcal{H}(\boldsymbol{t}) + \boldsymbol{b}_i) \tag{5.15}$$

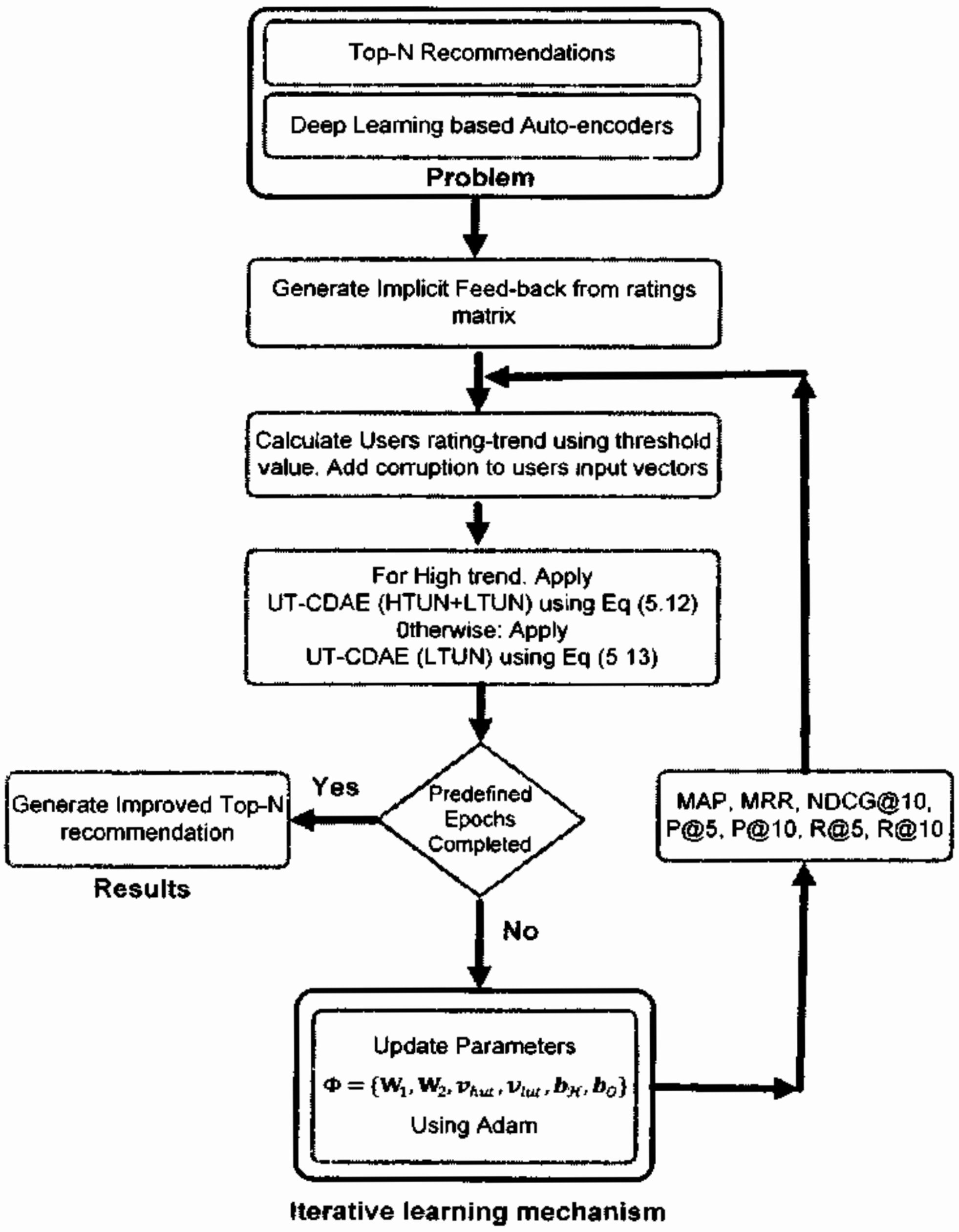
The overall graphical flow of the proposed UT-CDAE is given in Fig 5.4.

Algorithm 5.1: Pseudo code of proposed UT-CDAE algorithm for top-N recommender systems

Input: Corrupted User Preference Vectors
Output: Clean User preference vectors

```
1)  Initialize parameters randomly
2)  Set $epoch = 1$
3)  **While** $epoch < epochs$ **do**
4)      **for all** $u \in \wp$ **do**
5)          Add noise to input user vector $\bar{\boldsymbol{t}}_u \sim p(\bar{\boldsymbol{t}}_u | \boldsymbol{t}_u)$ through **Equation 5.9**
6)          Calculate rating-trend $\gamma$ for a User $u$
7)          **If** High user rating-trend i.e. $(\gamma = 0)$
8)              Compute $\mathcal{H}(\boldsymbol{t})$ from **Equation 5.12**
9)          **Else** $(\gamma = 1)$
10)             Compute $\mathcal{H}(\boldsymbol{t})$ from **Equation 5.13**
11)         Take negative samples $S_{un} \subset \bar{\mathbb{Z}}_u$
12)         **for all** $i \in \mathbb{Z}_u \cup S_{un}$ **do**
13)             Update parameters $\Phi = \{\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{v}_{hut}, \boldsymbol{v}_{lut}, \boldsymbol{b}_{\mathcal{H}}, \boldsymbol{b}_O\}$ through **back-propagation**
14)         **end for**
15)     **end for**
16)     $epoch = epoch + 1$
17) **end while**
```

A Comprehensive study including simulation results for performance comparison of proposed UT-CDAE with standard DAE and CDAE in terms of different evaluation metrics is given in the following section.



Top-N Recommendations
Deep Learning based Auto-encoders
Problem
Generate Implicit Feed-back from ratings matrix
Calculate Users rating-trend using threshold value. Add corruption to users input vectors
For High trend. Apply UT-CDAE (HTUN+LTUN) using Eq (5.12) Otherwise: Apply UT-CDAE (LTUN) using Eq (5 13)
Predefined Epochs Completed
Yes
No
Generate Improved Top-N recommendation
Results
MAP, MRR, NDCG@10, P@5, P@10, R@5, R@10
Update Parameters
$\Phi = \{W_1, W_2, v_{hut}, v_{lut}, b_{\mathcal{H}}, b_O\}$
Using Adam
Iterative learning mechanism

Fig. 5.4 The graphical flow of proposed UT-CDAE for Top-N Recommender systems

5.4 Simulations and Results

To compare the performance of proposed UT-CDAE with DAE and CDAE in detail, this section presents three subsections i.e., simulation details, mathematical description of evaluation metrics and discussion on results.

5.4.1 Simulation Details

In this sub-section we split ratings into 80% training set and 20% test set. We randomly choose the training examples from the dataset and treat the remaining examples as test data. We convert explicit numerical feedback (on a scale of 1-5, 1 for lowest and 5 for the highest rating) into binary implicit feedback and follow the convention that we don't keep the numerical ratings less than 4 and assign them a binary value '0' but we keep numerical ratings greater than and equal to 4 and assign a binary value of '1' as already done in [28].

To prove the effectiveness of our proposed method, we consider two datasets from MovieLens for the evaluation of methods in our experimentations. ML-100K and ML-1M are considered standard datasets for assessing the performance of recommender systems. The range of numerical ratings for both ML-100K and ML-1M is from 1 to 5. Minimum number of ratings per user for each dataset are 20. The particulars of both MovieLens datasets are summarized in Table 5.2.

Table 5.2: Datasets Particulars

Dataset	**Ratings (R)**	**Users(U)**	**Items (I)**	**Density (%) R / (U*I) * 100**	**Min(R/U)**
ML-1M	1M	6040	3706	4.47	20
ML-100K	100K	943	1682	6.30	20

We use grid search for tuning and selection of best hyper-parameters on the training datasets for all methods. We performed 5 fold cross validation for 100 epochs and report the average results for all algorithms. Algorithms are assessed for different values of learning rates [0.1, 0.01, and 0.001] and the best results for suitable learning rates are reported for each method. Standard value of regularization parameter **reg_rate1 (0.01)** is taken for the three methods (DAE, CDAE and UT-CDAE). Another regularization parameter is used only by the proposed method named as reg_rate2 and UT-CDAE is evaluated by using various values of (**reg_rate2** = 0.1, 0.3, 0.5, 0.7, 0.9). We use masking noise ϵ to corrupt input vectors and performance of methods is tested against different corruption levels. These levels are called input corruption ratio (ICR = 0.2, 0.5, 0.8). The values of ICR depicts proportion of corruption (where each entry of input is randomly overwritten by 0 with a probability of ICR) in the input. Here ICR = 0.8 indicates that 80 percent of total inputs are randomly overwritten by 0. Simulations are performed using a single hidden layer for the network with (K=50) latent dimensions. We use SGD as optimizer with a mini-batch size of 100 for ML-100K and 500 for ML-1M to learn the parameters for all the algorithms including the proposed one. During the learning of algorithms, Adam [174] is selected for the learning rate to adapt automatically.

Results in Tables and Figs for DAE, CDAE and UT-CDAE are presented based on the optimal learning rate hyper-parameter value **LR = 0.01** which is set to adapt automatically, with three input corruption ratio variations ICR = (0.2, 0.5, 0.8) and **(reg_rate1=0.01).** Values of **reg_rate2** used exclusively for the proposed method are **(reg_rate2= 0.9 for ICR= 0.2 and 0.5 and reg_rate2=0.7 for ICR = 0.8)**. The optimum hyper-parameters selection and settings are summarized in Table 5.3.

Table 5.3: Hyper-Parameters Description

Hyper-Parameter	**Notation**	**Tuning Parameter Values**	**Optimal Values UT-CDAE**	**Optimal Values CDAE**	**Optimal Values DAE**
Learning Rate	μ	0.1, 0.01, 0.001	0.01	0.01	0.01
Regularization Rate1	λ_1	0.01	0.01	0.01	0.01
Regularization Rate2	λ_2	0.01,0.1, 0.3, 0.5, 0.7, 0.9	0.7, 0.9	----	----
Noise Type	ϵ	Masking Noise	Masking Noise	Masking Noise	Masking Noise
Latent Dimensions	K	50	50	50	50

Simulations are performed in Spyder 3.3.2 release 2015 by means of Python 3.5 (64 bit) on Windows 10 Pro Education 2018 operating system (64 bit). Experimentations are completed on laptop with these specifications. (Core-i7-5600U @ 2.60 GHz) Processor and DDR2 16 GB Ram. We implemented all methods in python using tensorflow.

5.4.2 Evaluation Metrics for Top-N Recommendations

In top-N items ranking approach, a set of top-N ranked items are selected as recommendations to a user. A set of items that is recommended to the user is denoted by $\Psi(N)$, which is equal in size to the recommended list of N items. Suppose H be the set of all possible relevant items for a user, then for any list of top-N recommendations, following ranking based metrics at N are defined as follows [1]:

Precision (P@N): Precision is the percentage of recommended items that are also relevant out of all possible recommended items for a user u in the top-N recommendation set.

$$P@N = \left|\frac{\Psi(K) \cap \mathrm{H}}{|\Psi(N)|}\right| \times 100 \qquad (5.16)$$

Recall (R@N): Recall is the percentage of relevant items that are also recommended out of all possible relevant items for a user u in the top-N recommendations set.

$$R@N = \left|\frac{\Psi(N) \cap \mathrm{H}}{|\mathrm{H}|}\right| \times 100 \qquad (5.17)$$

MAP: The ranking metric MAP is meant to evaluate the fraction of relevant items from recommendation set for a user. Different equally spaced sizes of recommendation sets are considered for MAP for p users and the mean of precision is taken for all sets with different sizes.

$$MAP = \frac{1}{p}\sum_{u=1}^{p}(AP@Hits)_u \qquad (5.18)$$

AP is the average precision of relevant items of a user u for all hits from a recommendation set.

$$AP@Hits = \frac{1}{\mathrm{H}}\sum_{t=1}^{Hits} P(t).relv(t) \qquad (5.19)$$

Here, $relv(t)$ indicates whether the relevance of t^{th} item is true$(relv(t) = 1)$ or false$(relv(t) = 0)$.

NDCG: The Normalized cumulative discounted gain is the ratio of discounted cumulative gain to the ideal value of discounted cumulative gain (IDCG). The IDCG is computed by repetitive computations for DCG by arranging all the items in the test set in an ideal order after normalization.

$$NDCG = \frac{DCG}{IDCG} \tag{5.20}$$

$$DCG = \frac{1}{p}\sum_{u=1}^{p}\sum_{i=1}^{S_u}\frac{f_{ui}}{\log_2(w_i+1)} \tag{5.21}$$

Here, The set of items rated by user u, which is concealed from recommender system before estimation is denoted by S_u and the relevance of item i for a user u is $relv_{ui}$. The utility of user u towards item i is represented by f_{ui} and w_i is the rank of item i in the test set S_u.

Where,

$$f_{ui} = 2^{relv_{ui}} - 1 \tag{5.22}$$

Here, the relevance of item i for user is $relv_{ui}$.

NDCG@N: It is also possible to evaluate the DCG over a recommendation set of length $\Psi(N)$ which is given as:

$$DCG = \frac{1}{p}\sum_{u=1}^{p}\sum_{i\epsilon S_u,\, w_i=1}^{\Psi(N)}\frac{f_{ui}}{\log_2(w_i+1)} \tag{5.23}$$

MRR: The mean reciprocal rank is also known as average reciprocal hit rate (ARHR). It is defined as the average reciprocal hit rate over all p users. The range of mean reciprocal rank lies between (0, 1).

$$MRR = \frac{\sum_{u=1}^{p} ARHR(u)}{p} \tag{5.24}$$

The average reciprocal hit rate ($ARHR$) is another evaluation metric, which is used for implicit feedback setting of datasets, where rating of a user u for an item i (r_{ui}) lies in (0, 1). In such implicit feedback setting $r_{ui} = 1$ shows a "Hit" which means a user rated an item and $r_{ui} = 0$ means that a user has not rated an item. While, missing values are treated as "0". Thus, $ARHR$ is defined as the role of item $i \in S_u$ to its utility.

$$ARHR(u) = \sum_{i=1}^{S_u} \frac{r_{ui}}{w_i} \tag{5.25}$$

Here, r_{ui} is the rating of a rating of a user u for an item i and w_i is the rank of item i in the test set S_u. Therefore, $\frac{r_{ui}}{w_i}$ denotes the collect utility of an item based on its rank.

5.4.3 Results Description

For ML-100K dataset, performance comparison of all three methods for various values of ICR (0.2, 0.5, 0.8) in terms of different evaluation metrics is shown in Table 5.4 and Figs. 5.5 to 5.9. For all evaluation metrics listed in Table 5.4, Figs. 5.5 to 5.9 show that our proposed method outperforms DAE and CDAE for ICR values 0.2 and 0.5. However, for ICR = 0.8, the proposed UT-CDAE has better performance that DAE but comparable with that of CDAE.

Primarily we assess performance of UT-CDAE in terms of MRR, MAP and NDCG measures and results are presented in Table 5.4, Figs. 5.5 and 5.6. It is observed that for corruption levels (ICR = 0.2, and 0.5), UT-CDAE consistently shows superior performance than other techniques (CDAE, DAE) from 15 to 100 epochs. For ICR = 0.8, performance of UT-CDAE is comparable with CDAE but UT-CDAE outperforms DAE. A small increase in scores for ICR = 0.8 is achieved by UT-CDAE after 100 epochs than CDAE. Such difference in results is due to the extra regularization of the proposed objective function, which is based on user's rating-trend.

Table 5.4: Performance Comparison of Methods w.r.t Input Corruption Ratios for ML-100K Dataset

METRICS	Corruption Ratio = 0.2			Corruption Ratio = 0.5			Corruption Ratio = 0.8		
	UTCDAE	**CDAE**	**DAE**	**UTCDAE**	**CDAE**	**DAE**	**UTCDAE**	**CDAE**	**DAE**
P@10	0.2034	0.1809	0.1807	0.2065	0.1945	0.1911	0.2159	0.2106	0.2071
R@10	0.2238	0.1955	0.1950	0.2315	0.2177	0.2058	0.2365	0.2388	0.2237
P@5	0.2480	0.2211	0.2165	0.2509	0.2335	0.2315	0.2654	0.2569	0.2501
R@5	0.1426	0.1247	0.1223	0.1473	0.1384	0.1324	0.1531	0.1551	0.1394
MAP	0.2003	0.1763	0.1732	0.2073	0.1939	0.1878	0.2182	0.2159	0.2030
MRR	0.4849	0.4549	0.4418	0.4925	0.4714	0.4633	0.5193	0.5067	0.4880
NDCG	0.5078	0.4842	0.4799	0.5144	0.4999	0.4952	0.5263	0.5224	0.5096
NDCG@5	0.2841	0.2548	0.2484	0.2895	0.2706	0.2655	0.3076	0.2991	0.2868
NDCG@10	0.2799	0.2497	0.2460	0.2868	0.2693	0.2618	0.3007	0.2955	0.2833

The algorithms' performance is further verified with reference to NDCG@5 and NDCG@10 for same ICR variations and the outcomes are as shown in Fig. 5.7. It can be seen in Fig. 5.7 that the relative improvement in performance of proposed UT-CDAE for ICR = 0.2 and 0.5 is far better than other methods. This enhancement in performance is due to the flexibility in architecture of the proposed method. UT-CDAE attains similar performance as that of CDAE for ICR= 0.8 after 100 epochs but it performs significantly better than DAE with the same settings.

Precision curves for the competing methods are shown as P@5 and P@10 in Fig. 5.8 for same ICR settings. It is observed in Fig. 5.8 (a)-(f) that after 10 epochs, our proposed UT-CDAE achieves outstanding performance when compared to that of CDAE and DAE for P@5 and P@10 with two ICR variations (0.2, 0.5). Noticeable increase in precision score of UT-CDAE at P@5 for ICR = 0.5 is seen when compared to other methods. This boost in performance is achieved because of the users' rating-trend based decision of adding weight vectors to the users input vectors. It can also be seen that with ICR = 0.8, our proposed method attains better final score in terms of P@5 and P@10 after 100 epochs but its performance is comparable to CDAE for all epochs. It also outperforms DAE for all epochs with ICR = 0.8.

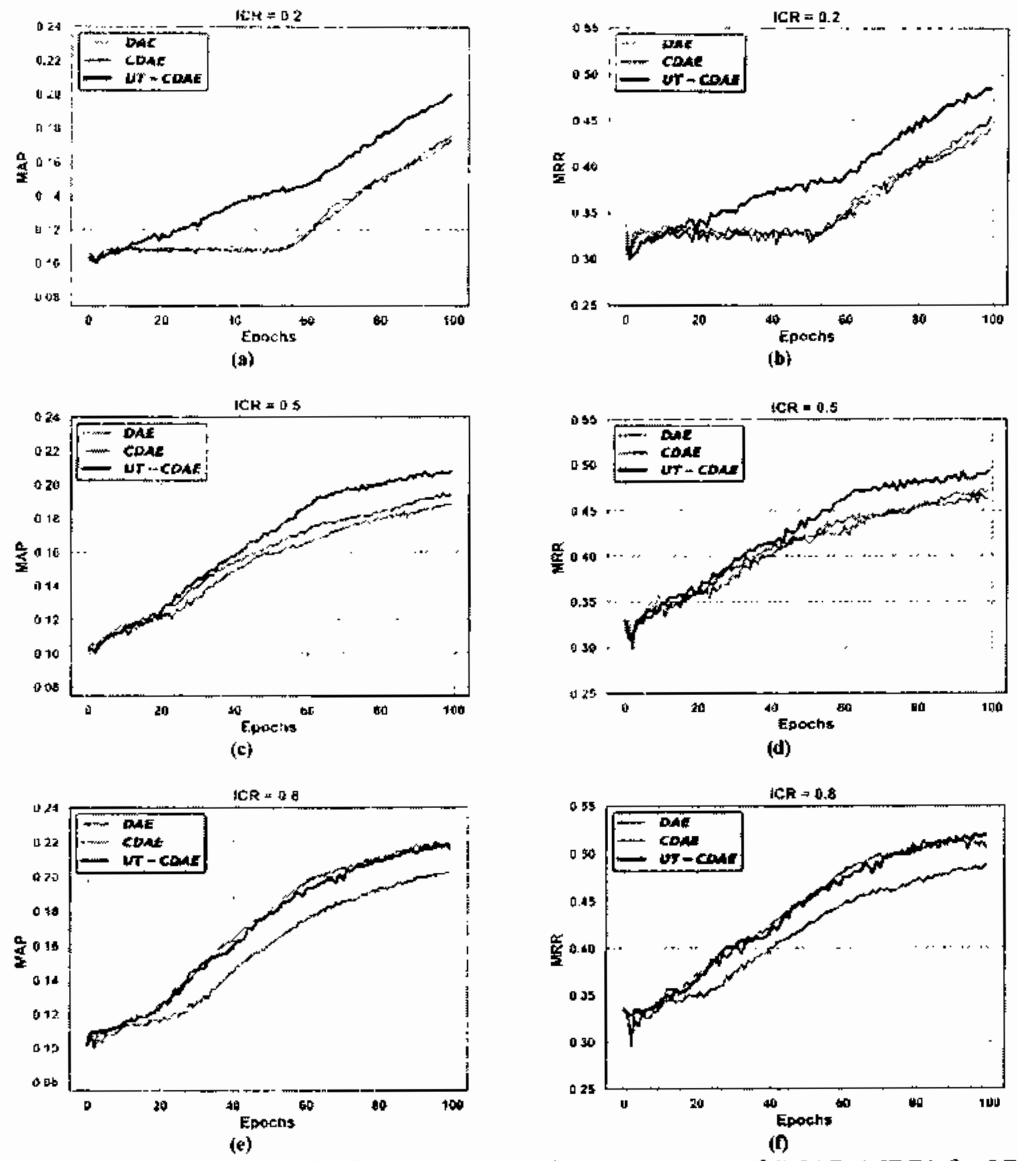
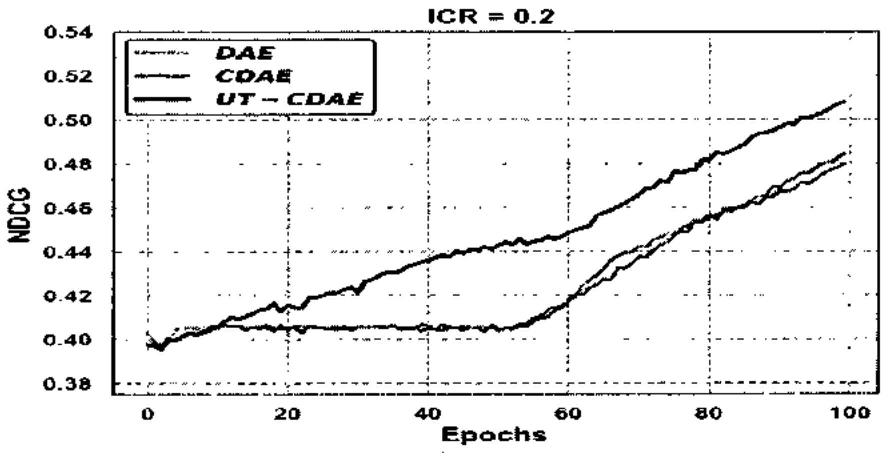
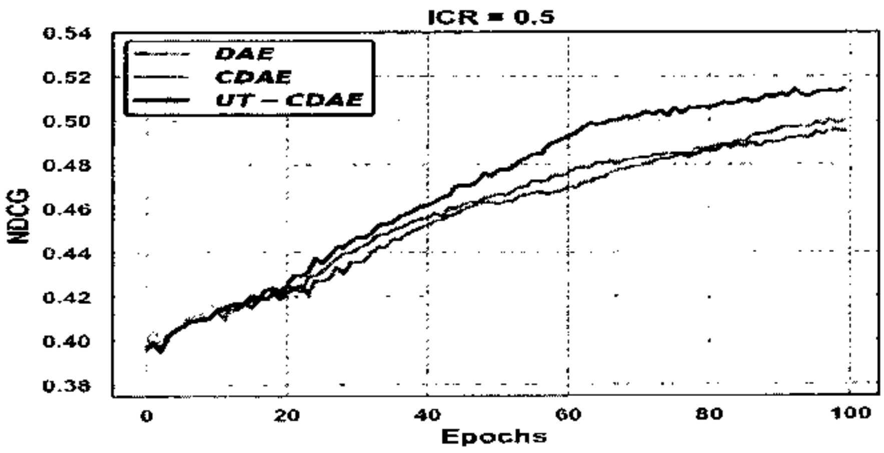


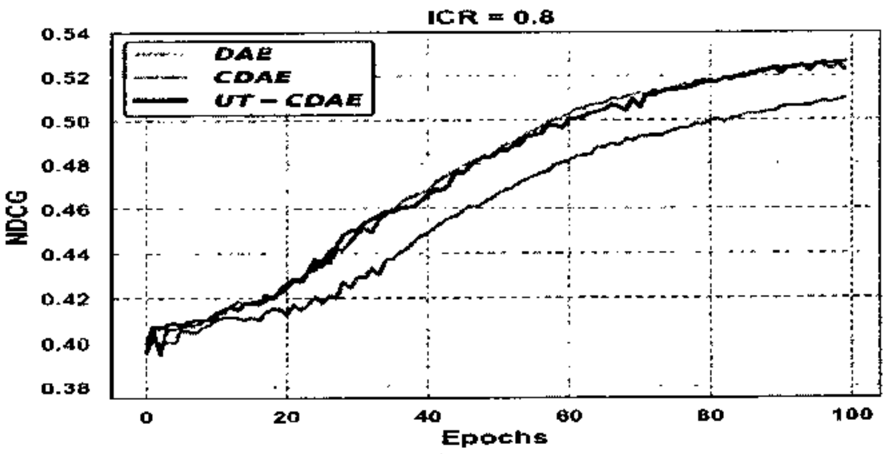
Fig. 5.5 Comparison of methods on ML-100K dataset in terms of (MAP, MRR) for ICR variations



(a)



(b)



(c)

Fig. 5.6 Comparison of methods on ML-100K dataset in terms of NDCG for ICR variations

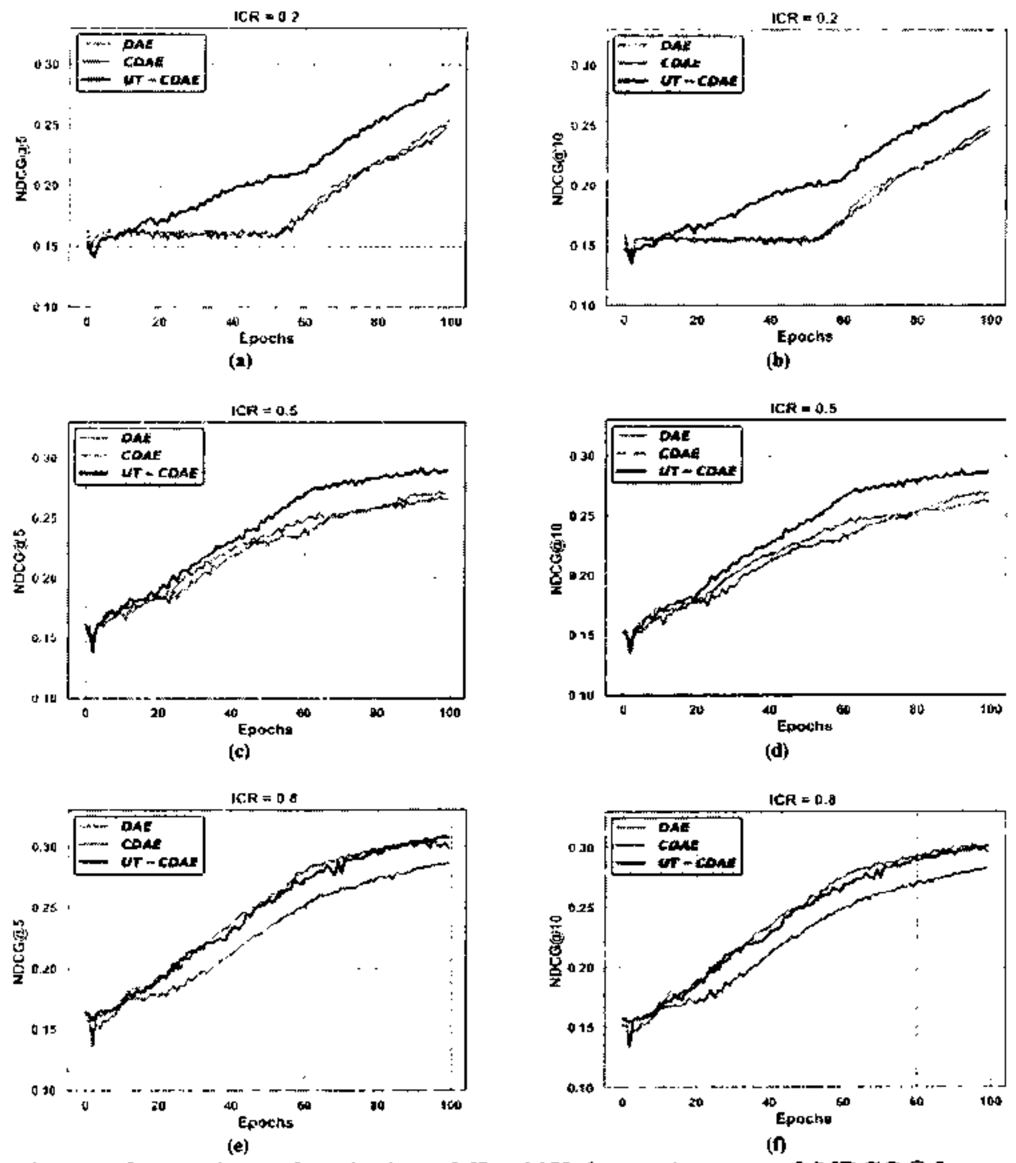


Fig. 5.7 Comparison of methods on ML-100K dataset in terms of (NDCG@5, NDCG@10) for ICR variations

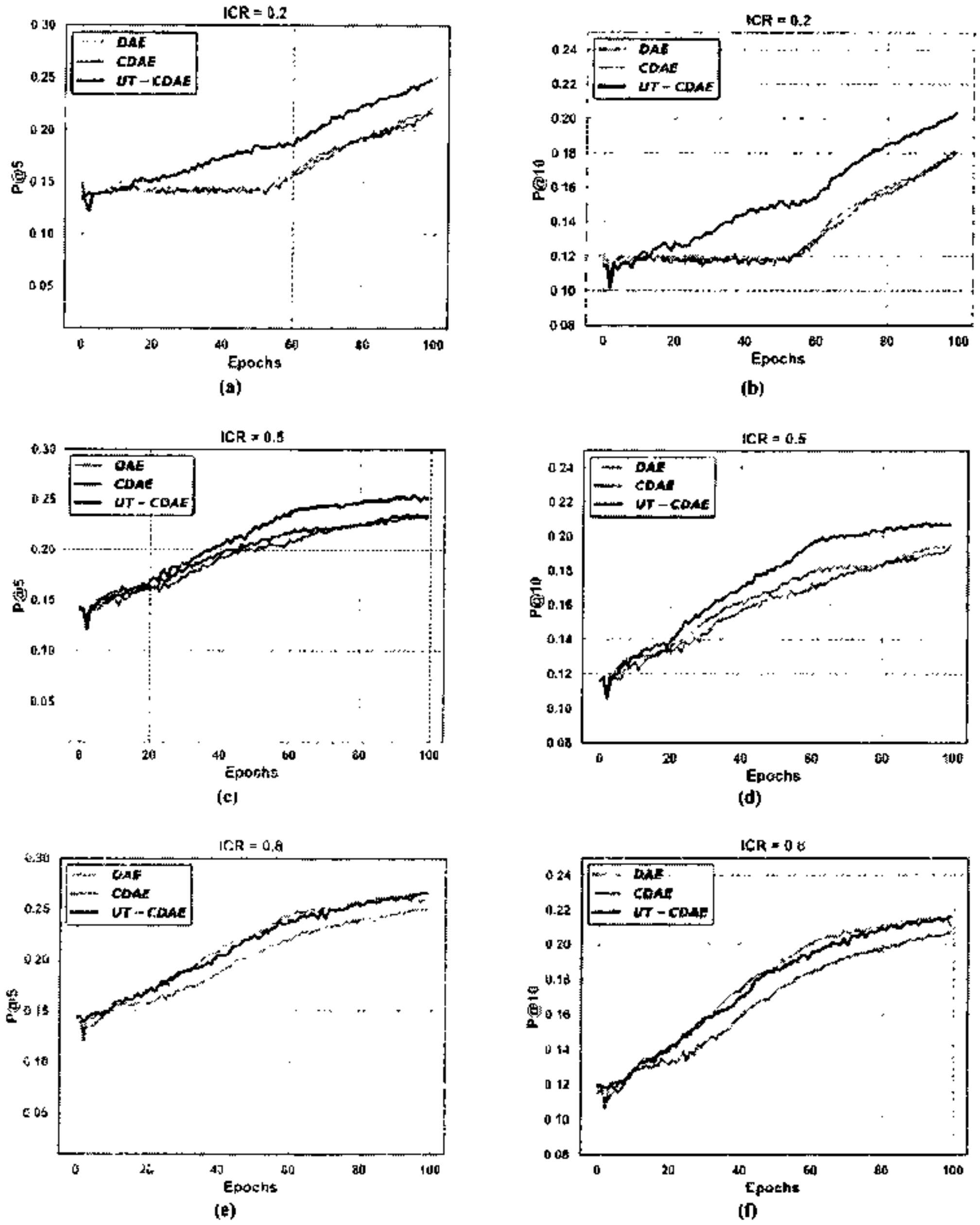


Fig. 5.8 Comparison of methods on ML-100K dataset in terms of Precision (P@5, P@10) for ICR variations

Fig. 5.9 presents recall (R@5 and R@10) results for DAE, CDAE and our UT-CDAE method. It is clearly depicted in Fig. 5.9 (a)-(f) that DAE shows worst performance than CDAE and UT-CDAE in terms of R@5 and R@10 for all epochs with all ICR variations. This is due to lack of user oriented weight vectors for each user. Moreover, Fig. 5.9 portrays that our proposed method performs significantly better than counterparts for ICR = 0.2 and 0.5. It is also observed that for ICR = 0.8, R@5 scores for UT-CDAE and CDAE are comparable but CDAE score for R@10 exceeds UT-CDAE after 30 epochs.

In Fig. 5.10, relative comparison of the proposed UT-CDAE with other two variants is made using bar-chart for ML-100K dataset. Fig. 5.10 (a) shows the scores based on P@5 for DAE, CDAE and suggested UT-CDAE methods for different values of input corruption ratio (ICR = 0.2, 0.5 and 0 .8). It is observed that UT-CDAE performs significantly well with precision (0.2509) for ICR = 0.5 than other methods. UT-CDAE shows appreciable performance (0.2480) for ICR = 0.2. However, UT-CDAE exhibits similar precision scores (0.2654) as CDAE (0.2569) and DAE (0.2501) with ICR = 0.8.

P@10 scores of DAE, CDAE and proposed UT-CDAE are presented in Fig. 5.10 (b) against ICR variations. The precision value (0.2034) achieved by UT-CDAE as compared to other counterparts for ICR = 0.2 is much better than ICR = 0.5 and 0.8. It is also seen that for ICR = 0.5, UT-CDAE leads both CDAE and DAE with score (0.2065) but CDAE and DAE perform in a similar fashion. For ICR=0.8, there is a slight improvement in performance of proposed method (0.2159) than CDAE (0.2106) whereas CDAE performs slightly better than DAE.

A similar performance trend in terms of R@5 of the proposed method is observed as that of P@10 for ICR = 0.2 and 0.5 than other methods as shown in Fig. 5.10 (c). The comparable behavior in recall score of UT-CDAE and CDAE is noticed for ICR = 0.8, where CDAE performs slightly better than proposed UT-CDAE with recall scores of (0.1551) and (0.1531) respectively but both CDAE and UT-CDAE show improved performance than DAE with recall score of (0.1394). A comparison between UT-CDAE concerning recall score for top-10 recommendations (R@10) with other methods is represented in Fig. 5.10 (d). The bars graphs for UT-CDAE with ICR = 0.2 and 0.5 depict superior performance of proposed scheme than other methods with recall scores of (0.2238) and (0.2315) respectively. For ICR = 0.8, proposed strategy (0.2365) slightly lags behind CDAE (0.2388) and both methods lead DAE for R@10.

Bar graphs for NDCG@5 and NDCG@10 in Fig. 5.10 (e) and (f) show improved performance of the proposed method than CDAE and DAE. For ICR 0.2 and 0.5, UT-CDAE NDCG values show improvement with good margins than other competing methods.

For all of the above discussed evaluation metrics, UT-CDAE has improved performance for ICR values of 0.2 and 0.5 as compared to other methods and comparable performance for ICR=0.8 to CDAE. This is because higher values of ICR randomly overwrites large number of input vector values with zeros, thus making low valued rating-trend dominant while learning latent features at the encoding node. This approximates UT-CDAE to CDAE and hence only one weight vector at the input layer is activated. This also validates our proposed method for modeling user rating-trend using two weight vectors $\boldsymbol{v}_{hut}$ and $\boldsymbol{v}_{lut}$ nodes.

Investigation through latent dimensions:

To observe the effect of latent features on the performance of the proposed method, the best performance of UT-CDAE with ICR = 0.8 is also tested for number of hidden dimensions (K). The performance of UT-CDAE in terms of MAP, MRR and NDCG with variations in latent dimensions is given in Fig. 5.11. It is witnessed from the Fig. 5.11 (a)-(c), that with increase in number of latent dimensions up to K = 50, performance of proposed method increases with K. When K is further increased beyond 50, performance of UT-CDAE starts declining. The cause of decrease beyond K = 50 is over-fitting. It is realized that proposed method performs much better for lesser number of latent dimensions. It is realized that the proposed method shows increasing trend in performance up to 50 latent dimensions and decreasing trend in performance afterwards. The percentage increase in performance of the UT-CDAE in terms of MAP, MRR and NDCG over CDAE and DAE against 50 latent dimensions with ICR = 0.8 using ML-100K dataset are, 1.07%, 2.49%, 0.75% and 7.49%, 6.41%, 3.28% respectively.

ML-1M Data Set: To further validate the performance of the UT-CDAE, we also evaluate it on ML-1M dataset in comparison with other methods.

Performance comparison of the three methods on ML-1M dataset is presented in Table 5.5. With reference to evaluation metrics i.e. MAP, MRR and NDCG, it is seen in Figs. 5.12 and 5.13 that for ICR = 0.2 UT-CDAE performs significantly better than counterparts. While, for ICR = 0.5 and 0.8, performance of UT-CDAE is comparable to CDAE. In addition to NDCG, performance of UT-CDAE in connection with NDCG@5 and NDCG@10 for ICR values against CDAE and DAE is presented in Fig. 5.14. The difference in performance of UT-CDAE than CDAE with regard to NDCG@5 and

NDCG@10 with ICR = 0.2 is substantial than with ICR = 0.8. However, UT-CDAE shows similar performance to that of CDAE for ICR = 0.5.

Fig. 5.15 represents performance of proposed UT-CDAE in terms of precision (P@5 and P@10). It is observed in Fig. 5.15 (a)-(b) and (e)-(f) that UT-CDAE achieves improved results than CDAE with ICR 0.2 and 0.8 but comparable results in Fig. 5.15 (c)-(d) to CDAE with ICR = 0.5. On the other hand, UT-CDAE performs significantly better than DAE for all ICR values. A similar behavior of the three methods can be seem for recall (R@5 and R@10) in Fig. 5.16 (a)-(f).

The bar charts in Fig. 5.17 (a)-(f) demonstrate the overall comparison of UT-CDAE, CDAE and DAE for various evaluation metrics against different ICR values on ML-1M dataset. It is perceived from Fig. 5.17 (a) that UT-CDAE shows better performance in terms of P@5 than other counterparts and the best score attained by UT-CDAE with ICR = 0.8 is (0.2712). Whereas, CDAE exhibits comparable and a bit improved behavior with score (0.2597) than UT-CDAE only with ICR = 0.5. The behavior of UT-CDAE regarding P@10 in Fig. 5.17 (b) is similar to that of P@5 in Fig. 5.17 (a) against CDAE and DAE. The maximum score attained by UT-CDAE is (0.2264) with ICR = 0.8. UT-CDAE only lags behind CDAE for ICR = 0.5 and it outperforms DAE for all ICR values.

The comparison of methods using bar charts for recall (R@5, R@10) and normalized discounted gain (NDCG@5, NDCG@10) scores with different ICR values are given in Fig. 5.17 (c)-(f). The comparative trend in performance of UT-CDAE in terms of recall and normalized discounted gain against CDAE and DAE is same as that of precision scores given in Fig. 5.17 (a)-(b). Fig. 5.17 (c) represents best R@5 score (0.1079) achieved by UT-CDAE and the improved R@10 score (0.1715) attained by UT-CDAE is presented in

Fig. 5.17 (d). However, finest scores in terms of NDCG@5 and NDCG@10 accomplished by UT-CDAE are (0.2970) and (0.2807), which are respectively given in Fig. 5.17 (e) and (f).

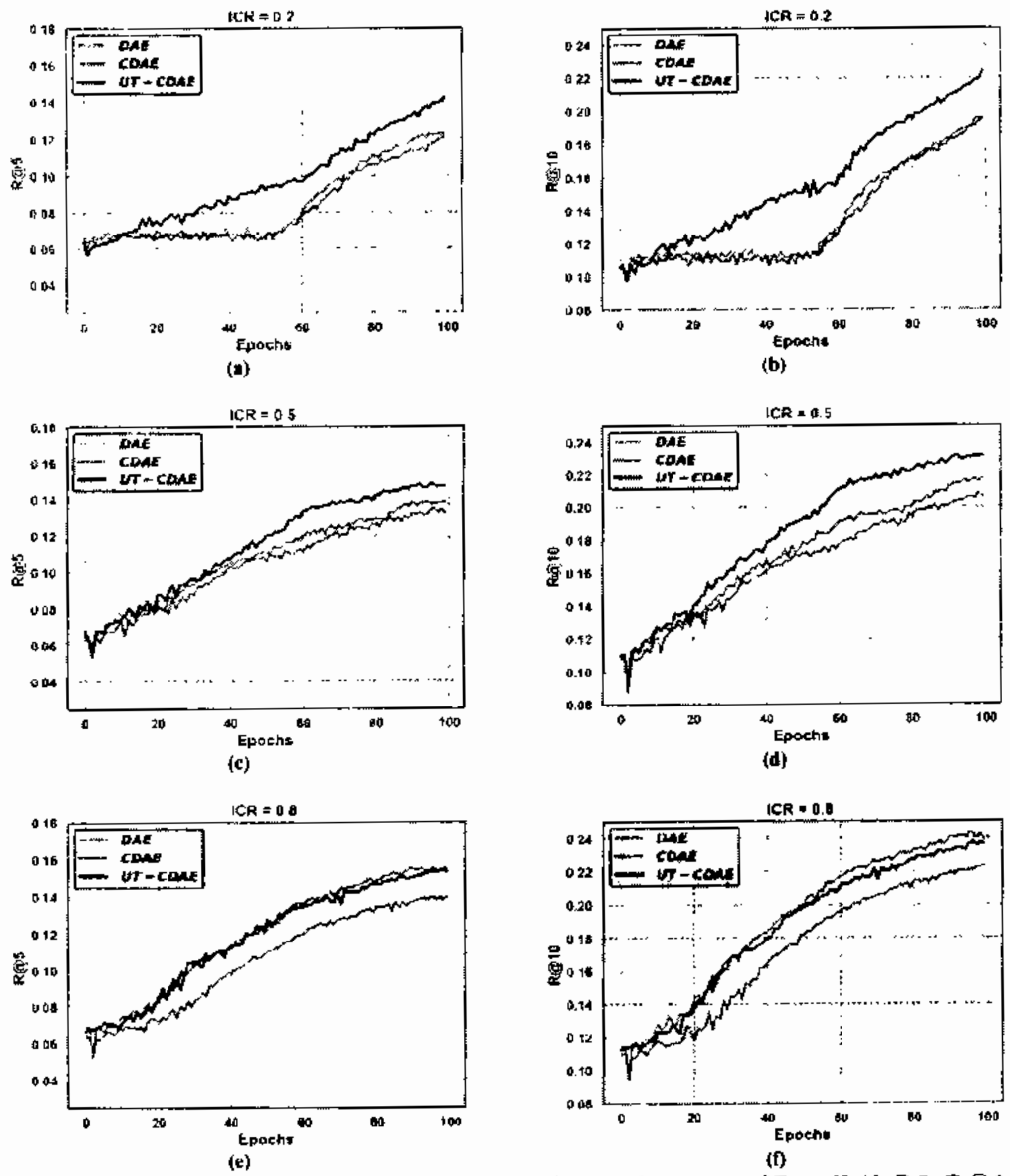


Fig. 5.9 Comparison of methods on ML-100K dataset in terms of Recall (R@5, R@10) for ICR variations

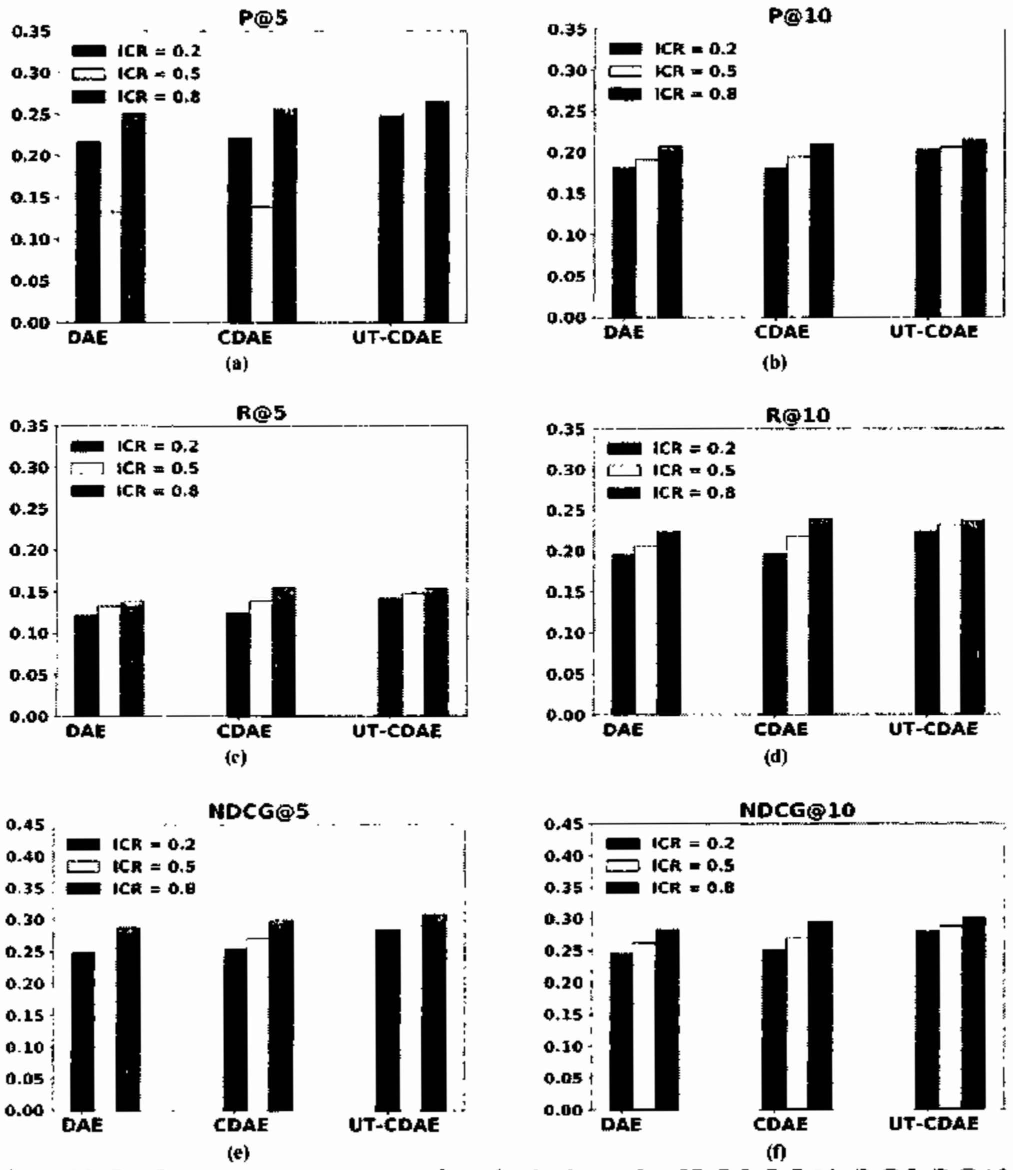
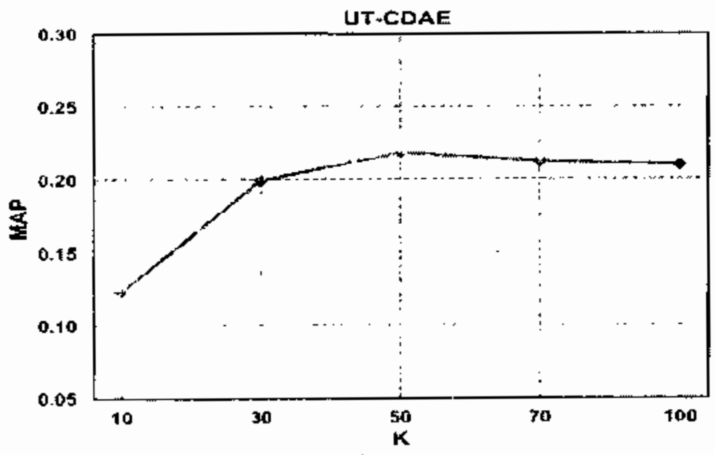
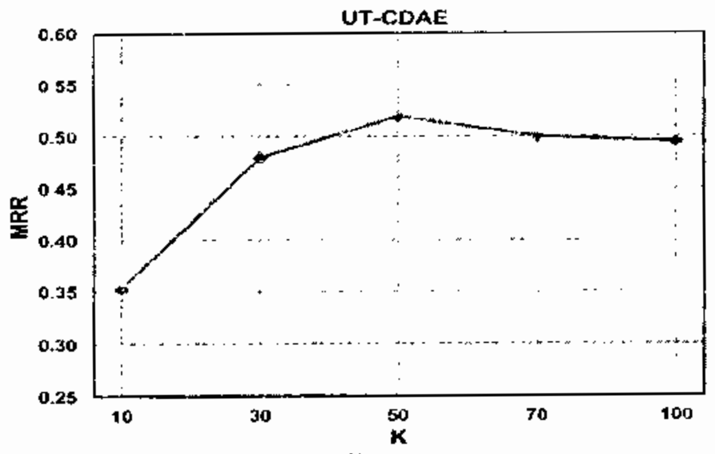


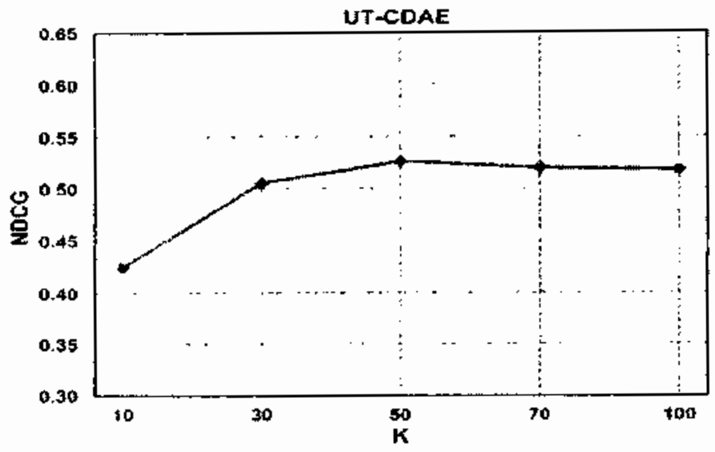
Fig. 5.10 Performance comparison of methods through of P@5, P@10, R@5, R@10, NDCG@5 and NDCG@10 for ICR variations on ML-100K dataset



(a)



(b)



(c)

Fig. 5.11 Performance of UT-CDAE 0n ML-100K dataset in terms of MAP, MRR and NDCG for Latent dimensions (K)

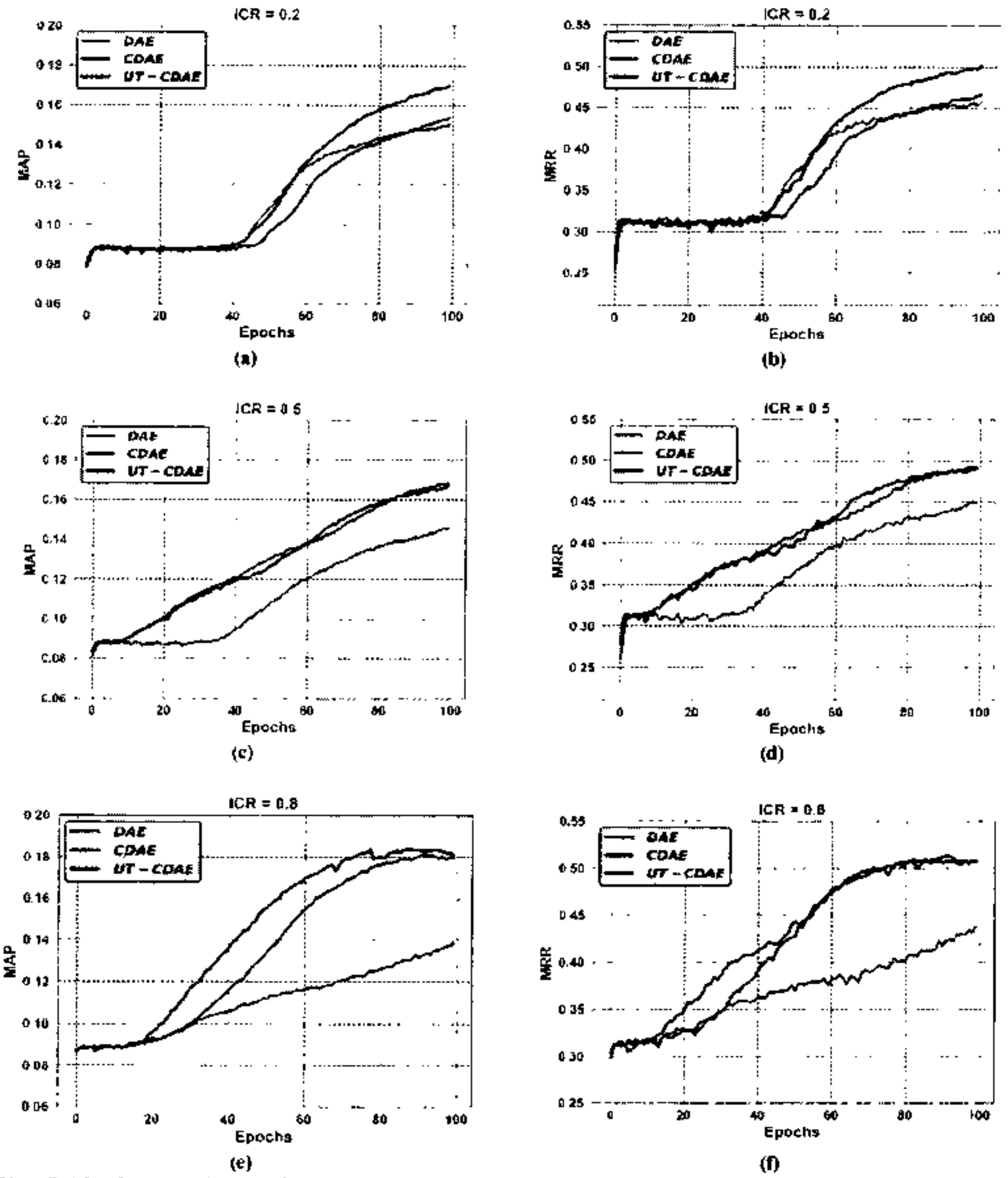
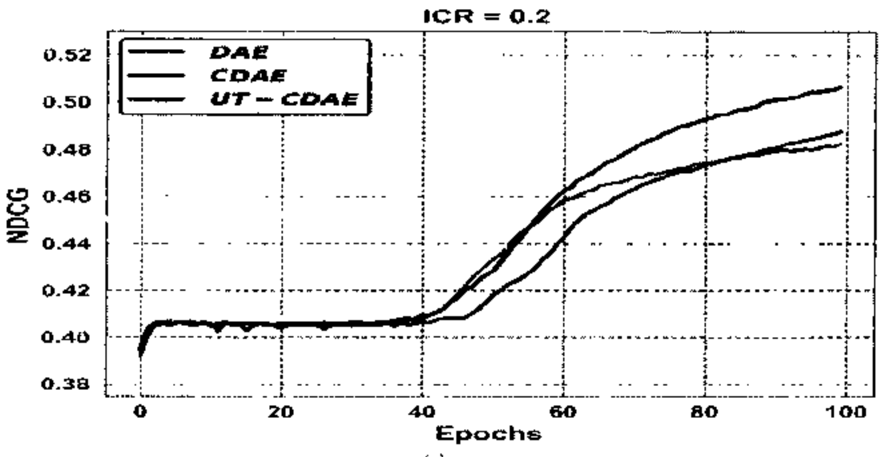
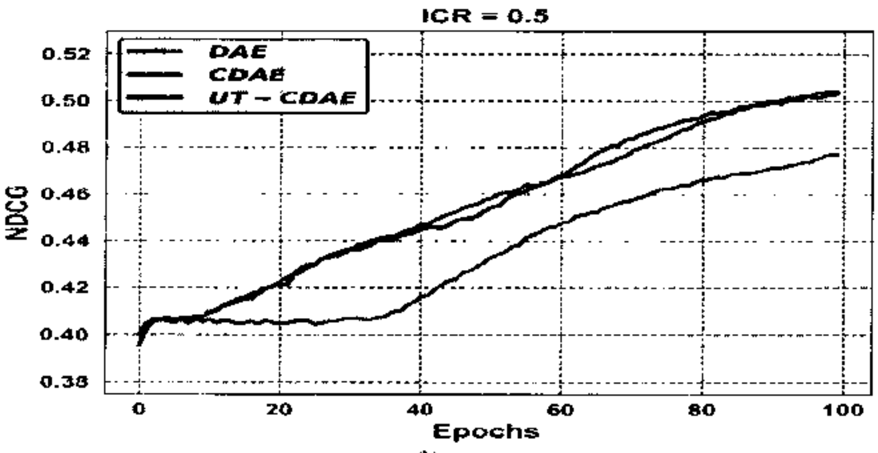


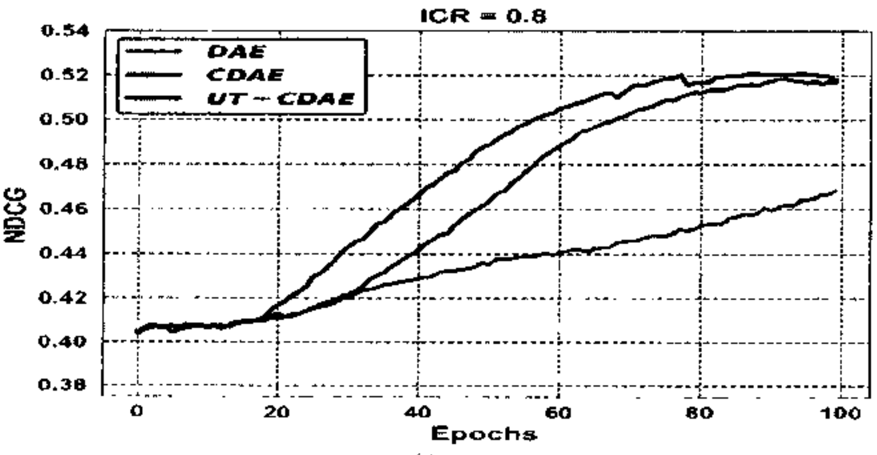
Fig. 5.12 Comparison of methods using ML-1M dataset through (MAP, MRR) against ICR variations



(a)



(b)



(c)

Fig. 5.13 Comparison of methods using ML-1M dataset through NDCG against ICR variations

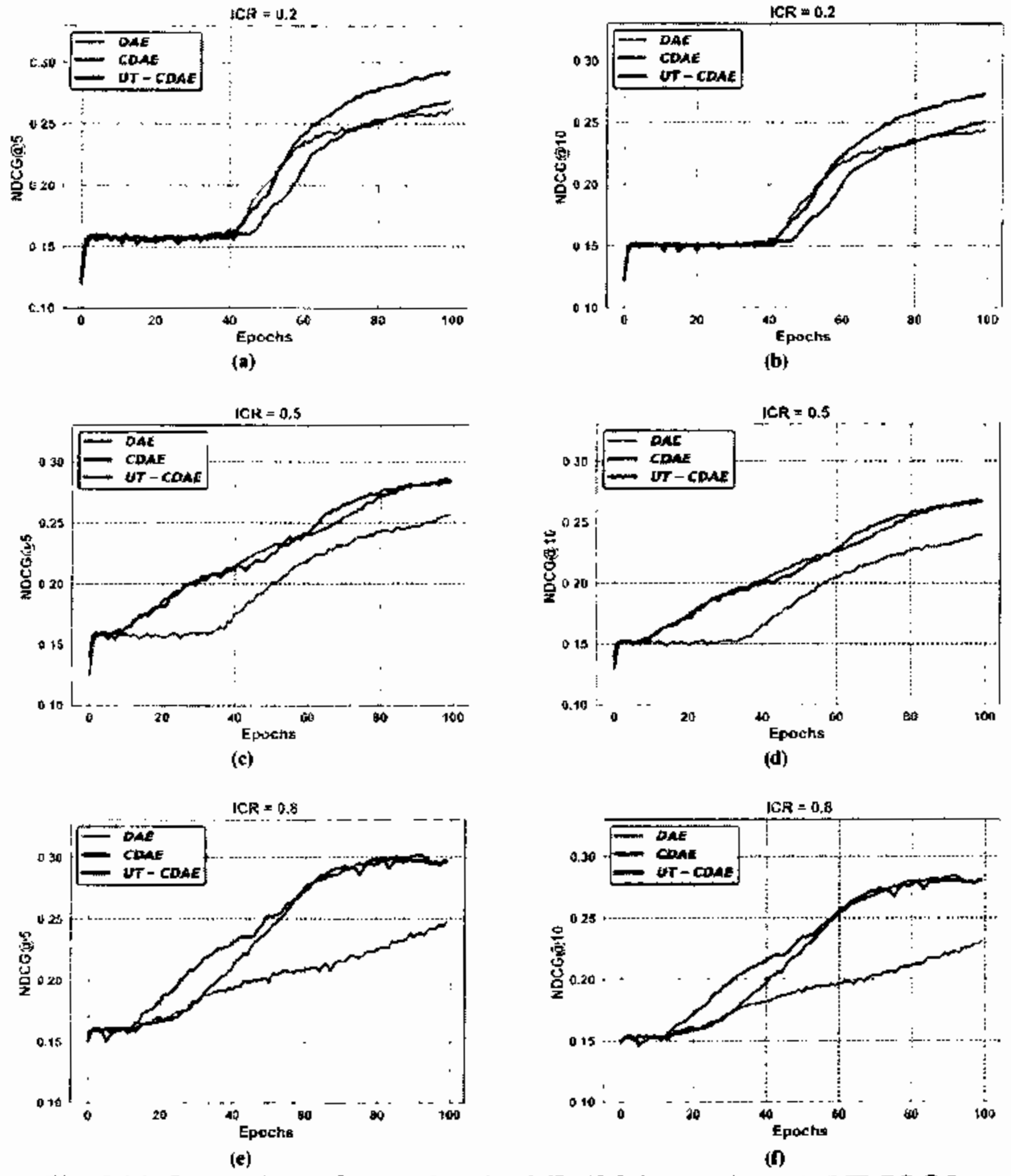


Fig. 5.14 Comparison of methods using ML-1M dataset through (NDCG@5, NDCG@10) against ICR variations

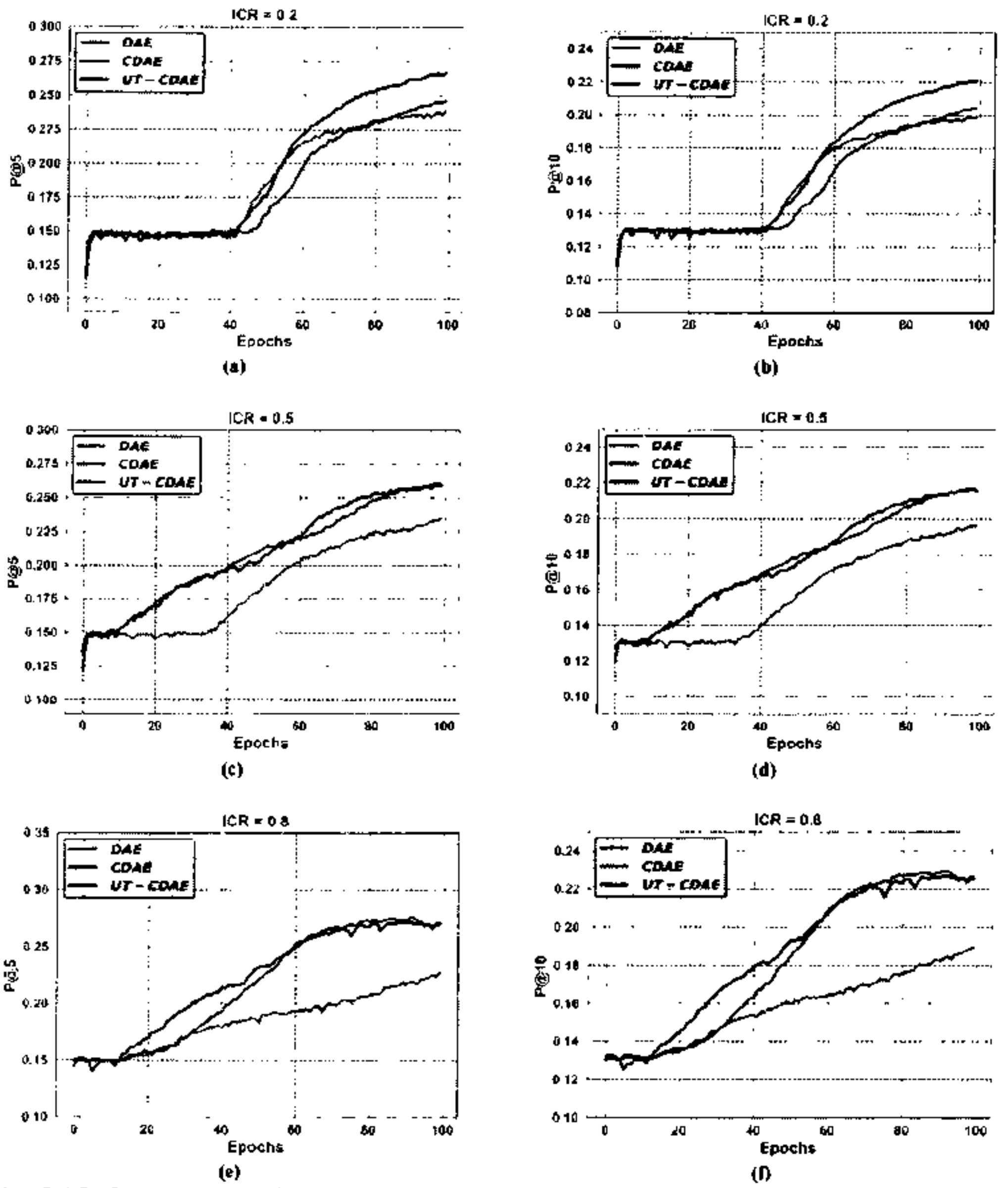


Fig. 5.15 Comparison of methods on ML-1M dataset through Precision (P@5, P@10) against ICR variations

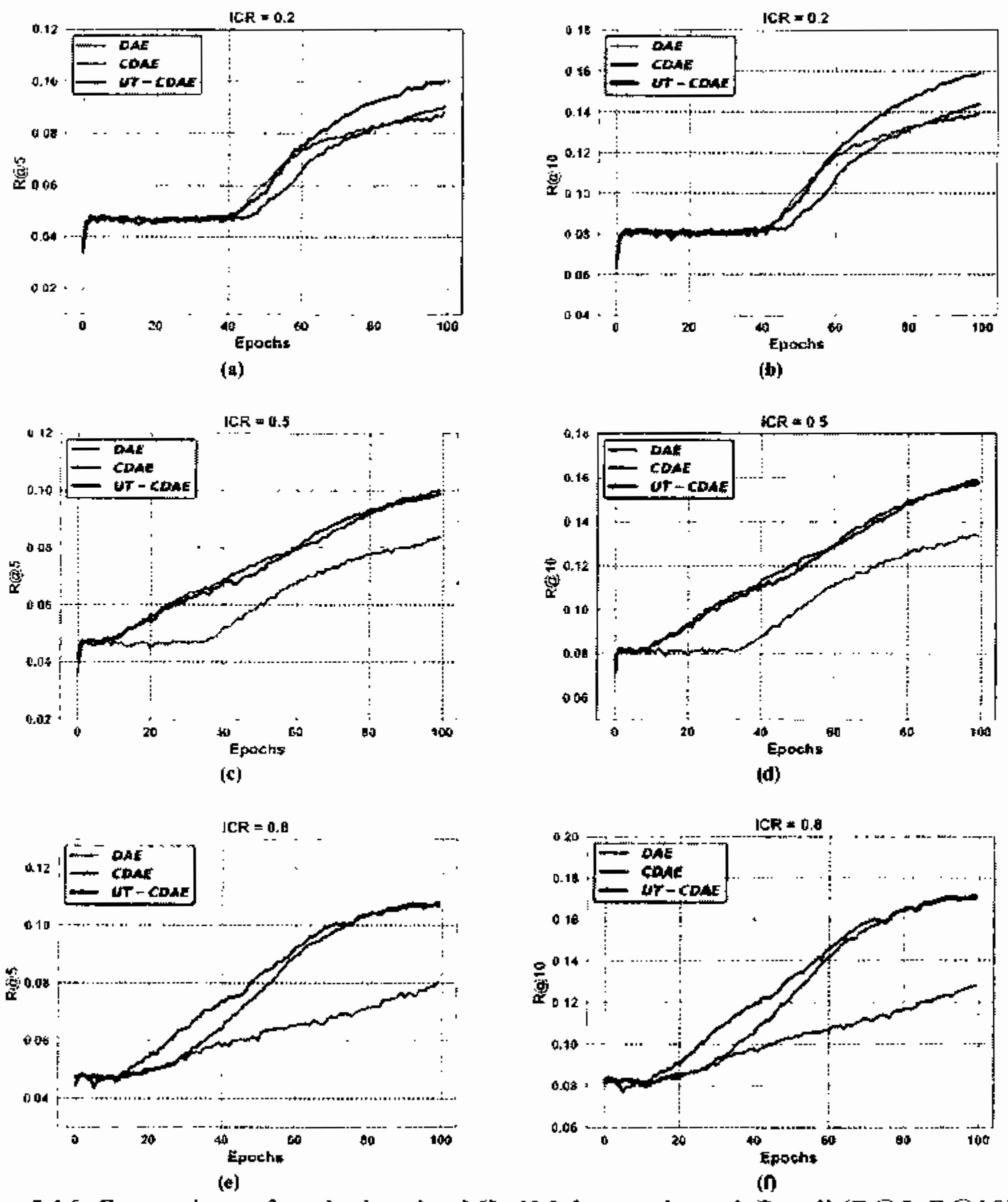


Fig. 5.16 Comparison of methods using ML-1M dataset through Recall (R@5, R@10) against ICR variations

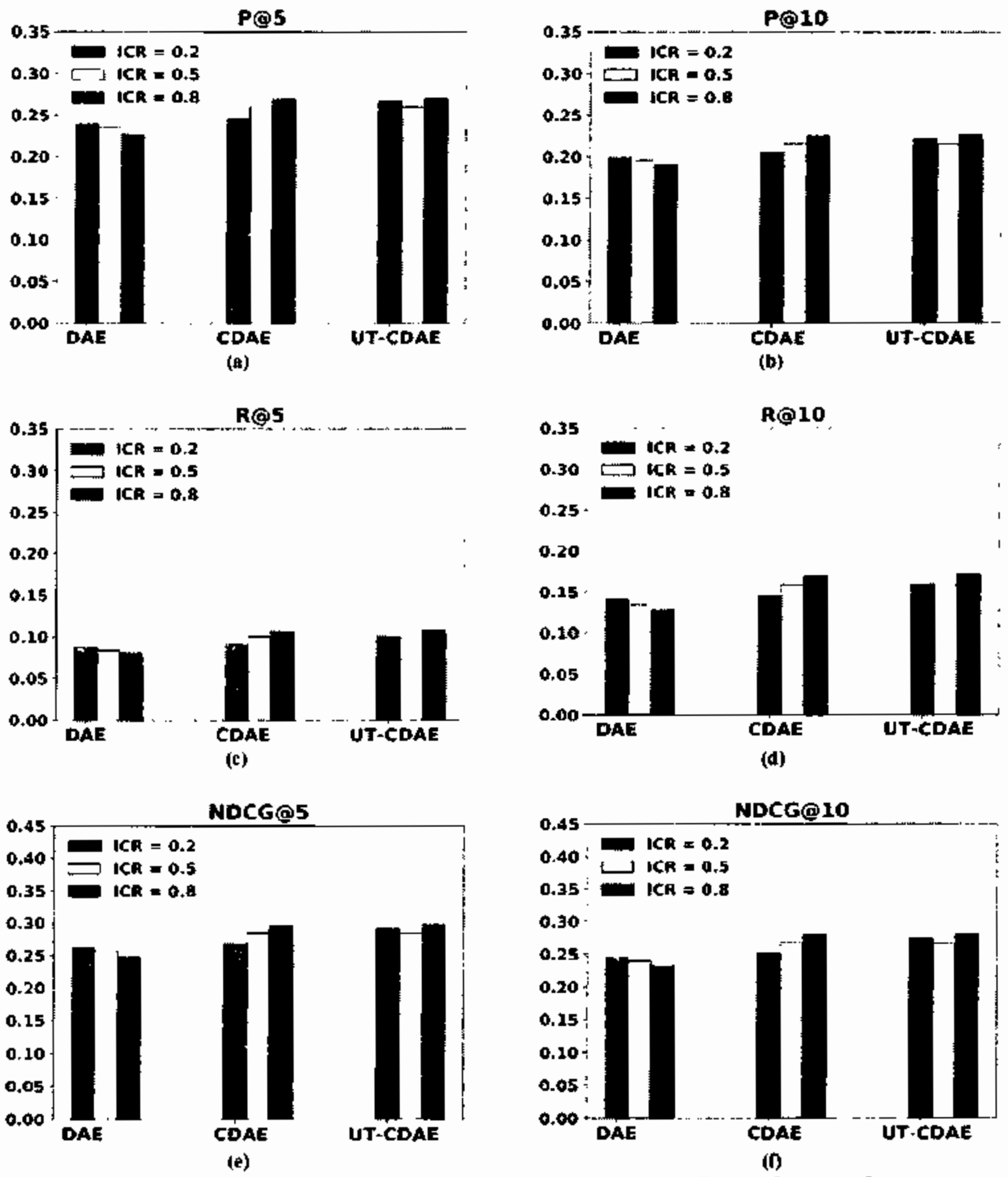


Fig. 5.17 Performance comparison of methods through P@5, P@10, R@5, R@10, NDCG@5 and NDCG@10 for ICR variations on ML-1M dataset

Table 5.5: Performance comparison of methods w.r.t input corruption ratios for ML-1M dataset

METRICS	Corruption Ratio = 0.2			Corruption Ratio = 0.5			Corruption Ratio = 0.8			Caser
	UTCDAE	CDAE	DAE	UTCDAE	CDAE	DAE	UTCDAE	CDAE	DAE	
P@10	0.2213	0.2048	0.1993	0 2154	0.2162	0.1960	0.2264	0.2254	0.1894	0.1991
R@10	0.1593	0.1441	0.1400	0.1573	0.1583	0.1338	0.1715	0.1700	0.1280	0.1121
P@5	0.2666	0.2459	0.2388	0.2593	0.2597	0.2350	0.2712	0.2698	0.2271	0.2175
R@5	0.1001	0.0904	0.0880	0.0987	0.1000	0.0837	0.1079	0.1067	0.0803	0.0632
MAP	0.1699	0.1539	0.1504	0.1665	0.1679	0.1457	0.1806	0.1794	0.1388	0.1507
MRR	0.5010	0.4666	0.4569	0.4897	0.4906	0.4493	0.5074	0.5081	0.4377	0.0925
NDCG	0.5065	0.4876	0.4826	0.5032	0.5042	0.4771	0.5183	0.5168	0.4685	---
NDCG@5	0.2927	0.2686	0.2610	0.2835	0.2844	0.2564	0.2970	0.2961	0.2480	---
NDCG@10	0.2732	0.2507	0 2442	0.2657	0.2670	0.2388	0.2807	0.2799	0 2306	---

Performance comparison of deep learning based methods:

To prove the usefulness of the proposed method, we have also compared the performance of UT-CDAE with few recent deep learning based methods apart from other competing methods of DAE and CDAE, for **ML-1M dataset**. The results of each deep learning based method are reported with best hyper-parameter settings. Brief introduction of the deep learning based baselines is as follows:

- **Caser** [175][176]: Caser is proposed to model Users' sequential patterns and general preferences. Caser is meant to generalize some standard approaches in a combined framework. Sequential patterns and skip behaviors are captured by Caser through vertical and horizontal convolutional filters. Caser optimize its network through cross-entropy.

- **NCF** [22]: The user and item preferences for Top-N recommendations are learned through Neural Collaborative Filtering (NCF) method. The development of NCF is accomplished by integrating matrix factorization and multi-layer perceptron (MLP).
- **ECAE** [22]: Enhanced Collaborative Auto-encoder (ECAE) is suggested to learn robust information for recommendation from produced soft targets (data) using knowledge distillation method. In ECAE, states generated from generation and retraining networks are combined to build an integrated framework. Using such unified network the soft targets can be tuned by propagating the training errors of retraining network to reduce noise for retaining useful information.

The proposed UT-CDAE outperforms deep learning based method **Caser** [175][176] in terms of different evaluation metrics i.e. **(P@10, R@10, P@5, R@5, MAP and MRR)** and results are presented in right most column of **Table 5.5**. To demonstrate the effectiveness of the proposed work, performance of the UT-CDAE is further matched with other two deep learning based methods **(NCF and ECAE)** [22] with regard to **NDCG@10.** NDCG@10 scores are given in **Table 5.6**. It is observed that UT-CDAE achieved significant improvement in terms of NDCG@10 against both methods. This behavior confirms the effectiveness of UT-CDAE for providing TOP-N recommendations.

5.4.4 In-Depth Analysis

To improve the interpretability of the proposed model, the in-depth analysis of results achieved by UT-CDAE is also presented through the following different performance measures.

Scalability and Robustness: To demonstrate the effectiveness of our proposed UT-CDAE model, we simulate our model with two MovieLens datasets i.e. ML-100K and ML-1M and found that UT-CDAE outperforms denoising variants i.e. CDAE and DAE for both datasets. Since ML-1M is much bigger dataset than ML-100K, which proves that suggested UT-CDAE is a scalable model which has the ability to show improved performance even for MovieLens larger dataset i.e. ML-10M as that of CDAE [28].

Furthermore, to validate the robustness of proposed UT-CDAE, the evaluation of suggested model is performed in terms of various performance measures such as P@10, R@10, P@5, R@5, MAP, MRR, NDCG, NDCG@5 and NDCG@10. The significant performance of UT-CDAE in comparison with CDAE and DAE for all metrics confirms that UT-CDAE is robust and its capability of modelling rating-trend improves the Top-N recommendations to the users.

Average Improvements of UT-CDAE relative to ML-100K and Ml-1M datasets:

The higher average improvements in performance of UT-CDAE are observed with ICR = 0.2. Hence, average results for ICR = 0.2 are of greater importance and are discussed below.

ML-100K: The average improvements of UT-CDAE in terms of P@10 and R@10 over CDAE and DAE are 12.44%, 14.48% and 12.56%, 14.77 respectively. Whereas, the percentage increase in performance with respect to P@5 and R@5 against CDAE and DAE are 12.17%, 14.35% and 14.55%, 16.60% respectively. Apart from precision and recall, the average increase in performance of UT-CDAE relative to MAP, MRR and NDCG as compared to CDAE and DAE are 13.61%, 6.59%, 4.87%, and 15.65%, 9.76%, 5.81% respectively. The percentage rise in performance of proposed method in terms of normalized discounted gain NDCG@5 and NDCG@10 in contrast to CDAE and DAE are respectively given as 11.50%, 12.09% and 14.37%, 13.78%.

ML-1M: A noticeable increase in average performance is achieved by UT-CDAE for ML-1M dataset with ICR = 0.2 than for ICR = 0.5 and 0.8 over CDAE and DAE in connection with evaluation metrics presented in the Table 5.5 for ML-1M dataset. It is observed that the average improvements obtained by UT-CDAE for ML-1M dataset are similar to that of ML-100K dataset. The considerable average improvements accomplished by UT-CDAE with ICR = 0.2 in comparison with CDAE for P@10, R@10, P@5, R@5, MAP, MRR, NDCG, NDCG@5 and NDCG@10 are respectively given as 8.06%, 10.55%, 8.42%, 10.73%, 10.40%, 7.37%, 3.88%, 8.97% and 8.97%. Moreover, it is observed that the percentage increase in performance of UT-CDAE than DAE relative to evaluation metrics such as P@10, R@10, P@5, R@5, MAP, MRR, NDCG, NDCG@5 and NDCG@10 with ICR = 0.2 is substantial than CDAE and the average increase in performance for these metrics against DAE are respectively given as 11.04%, 13.79%, 11.64%, 13.75%, 12.97%, 9.65%, 4.95%, 12.15% and 11.88%.

Analysis with respect to ICR variations: For all of the above discussed evaluation metrics, UT-CDAE has improved performance for ICR values of 0.2 and 0.5 as compared to other methods and comparable performance for ICR=0.8 to CDAE. This is because higher values of ICR randomly overwrites large number of input vector values with zeros, thus making low valued rating-trend dominant while learning latent features at the encoding node. This approximates UT-CDAE to CDAE and hence only one weight vector at the input layer is activated. This also validates our proposed method for modeling user rating-trend using two weight vectors $\boldsymbol{v}_{hut}$ and $\boldsymbol{v}_{lut}$ nodes.

Table 5.6: Performance Comparison of Methods with respect to NDCG@10 Scores achieved with ML-1M dataset

METHODS	NDCG@10 SCORES
NCF	0.1991
ECAE	0.2063
UT-CDAE (ICR = 0.2)	**0.2732**
UT-CDAE (ICR = 0.5)	**0.2657**
UT-CDAE (ICR = 0.8)	**0.2807**

5.5 Summary

In this chapter, we have proposed a novel variant of de-noising auto-encoder for top-N recommender systems. We call it users rating-trend based denoising auto-encoder (UT-CDAE). Furthermore, this chapter also includes basic denoising auto-encoder (DAE) and collaborative denoising auto-encoder (CDAE) to compare the performance of the suggested method in terms of various ranking evaluation metrics. The evaluation of the proposed UT-CDAE is accomplished through precision, recall, mean reciprocal rank, normalized discounted gain and mean average precision as ranking based evaluation measures for top-N recommendations.

Chapter 6.

Conclusion and Future Work

This chapter presents conclusions deduced from the suggested matrix factorization and deep learning based models discussed in the previous chapters. Apart from conclusions, the chapter also includes guidelines for scholars interested in doing future research by applying proposed methods or different variations of suggested methods in different engineering fields.

6.1 Conclusions

Conclusions inferred from the proposed work are as follows:

- A detailed investigation is made in this research work for developing fractional gradient based novel learning machines for recommender systems by exploiting strong mathematical foundations of fractional calculus.
- A deep learning based new learning machine is proposed in this thesis, which presents a novel architecture of denoising auto-encoders by exploiting users' rating-trend in providing precise Top-N recommendations of items to users.
- The work presented in this research regarding proposed fractional learning machines and proposed user's trend based deep learning machine is an innovation towards the development of novel, precise, convergent, robust and a non-linear

computing paradigm with novel application to solve recommender system problem effectively.

- The fractional order based SGD learning machines of fractional SGD (FSGD), momentum fractional SGD (mF-SGD) and normalized fractional SGD (NF-SGD) are proposed for efficient matrix factorization of recommender systems.
- The convergence speed and accuracy of recommendations are improved using FSGD, however convergence rate of FSGD is further improved in mF-SGD by incorporating the proportion of previous gradients efficiently. Moreover, to investigate the adaptive behavior of learning rate in FSGD for smoother convergence, NF-SGD is developed.
- The accuracy of the proposed fractional order based learning models is verified by applying it to standard recommender system datasets of MovieLens-100k and MovieLens-1M. The effectiveness of the proposed methods is established through comparisons with standard stochastic gradient descent variants.
- The quantitative evaluation of the proposed fractional gradient-based learning machines is performed through RMSE and it is observed that proposed learning machines have outperformed for RMSE with all variations of fractional orders, while higher value of fractional order yields better results.
- The percentage improvement achieved by F-SGD over SGD is 24.55% for ML-100K. Whereas, mF-SGD has achieved 16.05% and 0.41% better performance as compared to F-SGD and mSGD respectively for ML-100K dataset. Furthermore, for ML-1M dataset mF-SGD also has gained improved average performance of 1.11% and 4.62% as compared to standard SGD counterparts.

- The average improvements in performance for ML-100K in terms of RMSE accomplished by NF-SGD over NSGD and F-SGD are respectively given as 2.05% and 1.85%. While, for ML-1M dataset suggested NF-SGD obtained 3.7% and 0.11% better performance as compared to NSGD and F-SGD respectively.
- The results obtained through variations of standard SGD methods and proposed fractional learning machines improve by increasing number of features k, however, the proposed fractional variants provide better results than standard ones for different k variations.
- It is concluded to compute recommendations with large number of features as increasing the number of features increases accuracy at the cost of increase in computational complexity. This is because running time mainly depends upon number of features k and optimum number of runs (epochs).
- A new learning machine termed as users' rating-trend based denoising auto-encoder (UT-CDAE) is suggested in this thesis by exploiting the strength of deep learning for ranking based top-N predictions for a user.
- The trend based addition of weights provides an opportunity for UT-CDAE to learn more robust and non-linear latent representations, which helps to produce improved ranking based predictions at the output layer.
- The proposed learning machine has outperformed other de-noising auto-encoder based methods (DAE and CDAE) for relatively lower values of ICR (0.2 and 0.5). Low ICR values help in retaining a discernible proportion of users' rating trend in the data set and hence provide valuable information to UT-CDAE to model users' rating behavior. The greatest average improvements in performance of proposed

learning machine (UT-CDAE) as compared to the baselines (CDAE and DAE) are observed with smallest proportion of input noise such as ICR = 0.2.

- For ML-100K dataset, the percentage increase in performance of the proposed learning machine as compared to baselines with respect to MAP, MRR and NDCG are 13.61%, 6.59%, 4.87%, and 15.65%, 9.76%, 5.81% respectively. Moreover, the significant average improvements achieved by suggested learning machine for ML-1M dataset with ICR = 0.2 in comparison with standard counter parts for MAP, MRR and NDCG are respectively given as 10.40%, 7.37%, 3.88% and 12.97%, 9.65%, 4.95%.
- Comparison of the proposed learning model with DAE and CDAE in terms of various evaluation metrics show improved performance and robustness of the UT-CDAE for proposing top-N recommendation to the users.
- The proposed learning model not only provides additional flexibility in terms of regularization but also incorporates the features of other well established techniques which helps to predict improved top-N recommendations.

6.2 Future Work

The useful guidelines for future research for interested scholars in this area are as follows:

- One may explore in developing fractional gradient based novel adaptive strategies using new fractional derivatives for improved performance in recommender systems [177][178][179][180][181].

- Moreover, global search methodologies based on fractional swarming and evolutionary computing paradigms [182][183][184] looks promising to be exploited for solving complex recommender systems problems arising in different industrial applications such as [3][5].
- One future direction is to implement our proposed fractional order based MF models for recommender systems with both explicit and implicit feedback [185] for rating prediction and item ranking tasks.
- The possible and direct extension of the proposed UT-CDAE is to stack the model as already done in the stacked denoising auto-encoder [172].
- Another encouraging direction is to apply and explore other neural architectures like GAN, RNN and CNN in the context of users rating-trend based collaborative filtering.
- Attention models (Pre deep learning) being an interesting future research direction are not only able to improve performance but also provide greater interpretability and explainability of results to users [65].
- Moreover, we can explore Deep Multi-task Learning as a promising direction for future research for traditional recommender systems [186] because we can avoid overfitting by learning several tasks at a time with Deep Multi-task Learning. Moreover, sparsity problem in recommender systems can be alleviated by using Deep Multi-task Learning through implicit data augmentation.

BIBLIOGRAPHY

[1] C. C. Aggarwal, "An Introduction to Recommender Systems," in *Recommender Systems*, Cham: Springer International Publishing, 2016, pp. 1–28.

[2] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Syst.*, vol. 46, pp. 109–132, Jul. 2013.

[3] I. Heimbach, J. Gottschlich, and O. Hinz, "The value of user's Facebook profile data for product recommendation generation," *Electron. Mark.*, vol. 25, no. 2, pp. 125–138, Jun. 2015.

[4] M. Karimi, D. Jannach, and M. Jugovac, "News recommender systems – Survey and roads ahead," *Inf. Process. Manag.*, vol. 54, no. 6, pp. 1203–1227, Nov. 2018.

[5] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: A survey," *Decis. Support Syst.*, vol. 74, pp. 12–32, Jun. 2015.

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

[7] Z. Wang, N. Yu, and J. Wang, "Collaborative Filtering Recommendation Algorithm Based on Matrix Factorization and User Nearest Neighbors," in *Communications in Computer and Information Science*, vol. 643, Springer, Singapore, 2016, pp. 199–207.

[8] S. Köhler, T. Wöhner, and R. Peters, "The impact of consumer preferences on the accuracy of collaborative filtering recommender systems," *Electron. Mark.*, vol. 26, no. 4, pp. 369–379, Nov. 2016.

[9] C. He, D. Parra, and K. Verbert, "Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities," *Expert Syst. Appl.*, vol. 56, pp. 9–27, Sep. 2016.

[10] J. Ben Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative Filtering Recommender Systems," in *The Adaptive Web*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 291–324.

[11] T. Cunha, C. Soares, and A. C. P. L. F. de Carvalho, "Metalearning and Recommender Systems: A literature review and empirical study on the algorithm selection problem for Collaborative Filtering," *Inf. Sci. (Ny).*, vol. 423, pp. 128–144, Jan. 2018.

[12] S. Kant and T. Mahara, "Nearest biclusters collaborative filtering framework with fusion," *J. Comput. Sci.*, vol. 25, pp. 204–212, Mar. 2018.

[13] J. Salter and N. Antonopoulos, "CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering," *IEEE Intell. Syst.*, vol. 21, no. 1, pp.

35–41, Jan. 2006.

[14] M. J. Pazzani and D. Billsus, "Content-Based Recommendation Systems," in *The Adaptive Web*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 325–341.

[15] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer (Long. Beach. Calif).*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[16] J. Ben Schafer, J. A. Konstan, and J. Riedl, "E-Commerce Recommendation Applications," in *Applications of Data Mining to Electronic Commerce*, vol. 5, no. 1/2, Boston, MA: Springer US, 2001, pp. 115–153.

[17] 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 - SIGIR '06*, 2006, p. 501.

[18] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization.," in *Neural Information Processing Systems*, 2007, pp. 1257–1264.

[19] S. Wang, J. Tang, Y. Wang, and H. Liu, "Exploring Hierarchical Structures for Recommender Systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 6, pp. 1022–1035, Jun. 2018.

[20] L. Gao and C. Li, "Hybrid Personalized Recommended Model Based on Genetic Algorithm," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, 2008, pp. 1–4.

[21] S. Wang, M. Gong, H. Li, J. Yang, and Y. Wu, "Memetic algorithm based location and topic aware recommender system," *Knowledge-Based Syst.*, vol. 131, pp. 125–134, Sep. 2017.

[22] Y. Pan, F. He, and H. Yu, "A novel Enhanced Collaborative Autoencoder with knowledge distillation for top-N recommender systems," *Neurocomputing*, vol. 332, pp. 137–148, Mar. 2019.

[23] M.-H. Park, J.-H. Hong, and S.-B. Cho, "Location-Based Recommendation System Using Bayesian User's Preference Model in Mobile Devices," in *Ubiquitous Intelligence and Computing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1130–1139.

[24] J. Zhong and X. Li, "Unified collaborative filtering model based on combination of latent features," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 5666–5672, Aug. 2010.

[25] X. Luo, Y. Xia, and Q. Zhu, "Incremental Collaborative Filtering recommender based on Regularized Matrix Factorization," *Knowledge-Based Syst.*, vol. 27, pp. 271–280, Mar. 2012.

[26] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable Collaborative Filtering Approaches for Large Recommender Systems," *J. Mach. Learn. Res.*, vol. 10, no.

Mar, pp. 623–656, 2009.

[27] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec," in *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*, 2015, pp. 111–112.

[28] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative Denoising Auto-Encoders for Top-N Recommender Systems," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining - WSDM '16*, 2016, pp. 153–162.

[29] X. Luo, Y. Xia, and Q. Zhu, "Applying the learning rate adaptation to the matrix factorization based collaborative filtering," *Knowledge-Based Syst.*, vol. 37, pp. 154–164, Jan. 2013.

[30] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering Categories and Subject Descriptors," in *Proceedings of KDDCup.07*, 2007, pp. 39–42.

[31] Y.-F. Pu, J.-L. Zhou, Y. Zhang, N. Zhang, G. Huang, and P. Siarry, "Fractional Extreme Value Adaptive Training Method: Fractional Steepest Descent Approach," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 26, no. 4, pp. 653–662, Apr. 2015.

[32] A. Bouzeriba, A. Boulkroune, and T. Bouden, "Projective synchronization of two different fractional-order chaotic systems via adaptive fuzzy control," *Neural Comput. Appl.*, vol. 27, no. 5, pp. 1349–1360, Jul. 2016.

[33] H. A. Jalab, R. W. Ibrahim, and A. Ahmed, "Image denoising algorithm based on the convolution of fractional Tsallis entropy with the Riesz fractional derivative," *Neural Comput. Appl.*, vol. 28, no. S1, pp. 217–223, Dec. 2017.

[34] H. Baskonus, T. Mekkaoui, Z. Hammouch, and H. Bulut, "Active Control of a Chaotic Fractional Order Economic System," *Entropy*, vol. 17, no. 12, pp. 5771–5783, Aug. 2015.

[35] J. A. T. Machado, "Fractional order description of DNA," *Appl. Math. Model.*, vol. 39, no. 14, pp. 4095–4102, Jul. 2015.

[36] Y. Hu, Y. Fan, Y. Wei, Y. Wang, and Q. Liang, "Subspace-based continuous-time identification of fractional order systems from non-uniformly sampled data," *Int. J. Syst. Sci.*, vol. 47, no. 1, pp. 122–134, Jan. 2016.

[37] N. I. Chaudhary, M. Ahmed, Z. A. Khan, S. Zubair, M. A. Z. Raja, and N. Dedovic, "Design of normalized fractional adaptive algorithms for parameter estimation of control autoregressive autoregressive systems," *Appl. Math. Model.*, vol. 55, pp. 698–715, Mar. 2018.

[38] N. I. Chaudhary, S. Zubair, and M. A. Z. Raja, "A new computing approach for power signal modeling using fractional adaptive algorithms," *ISA Trans.*, vol. 68,

pp. 189–202, May 2017.

[39] S. M. Shah, R. Samar, N. M. Khan, and M. A. Z. Raja, "Design of fractional-order variants of complex LMS and NLMS algorithms for adaptive channel equalization," *Nonlinear Dyn.*, vol. 88, no. 2, pp. 839–858, Apr. 2017.

[40] M. S. Aslam and M. A. Z. Raja, "A new adaptive strategy to improve online secondary path modeling in active noise control systems using fractional signal processing approach," *Signal Processing*, vol. 107, pp. 433–443, Feb. 2015.

[41] L. Zhang, T. Luo, F. Zhang, and Y. Wu, "A Recommendation Model Based on Deep Neural Network," *IEEE Access*, vol. 6, pp. 9454–9463, 2018.

[42] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, "A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[43] F. Strub *et al.*, "Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs," *NIPS Work. Mach. Learn. eCommerce, Dec 2015*, pp. 1–9, 2015.

[44] R. Logesh, V. Subramaniyaswamy, D. Malathi, N. Sivaramakrishnan, and V. Vijayakumar, "Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method," *Neural Comput. Appl.*, pp. 1–24, Nov. 2018.

[45] R. Katarya and O. P. Verma, "Recommender system with grey wolf optimizer and FCM," *Neural Comput. Appl.*, vol. 30, no. 5, pp. 1679–1687, Sep. 2018.

[46] H. Yin, W. Wang, L. Chen, X. Du, Q. V. Hung Nguyen, and Z. Huang, "Mobi-SAGE-RS: A sparse additive generative model-based mobile application recommender system," *Knowledge-Based Syst.*, vol. 157, pp. 68–80, Oct. 2018.

[47] H. Zhou and K. Hirasawa, "Evolving temporal association rules in recommender system," *Neural Comput. Appl.*, pp. 1–15, Oct. 2017.

[48] R. Katarya, "Movie recommender system with metaheuristic artificial bee," *Neural Comput. Appl.*, vol. 30, no. 6, pp. 1983–1990, Sep. 2018.

[49] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola, "Maximum-Margin Matrix Factorization," in *NIPS*, 2004, vol. 17, pp. 1329–1336.

[50] R. M. Bell and Y. Koren, "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 2007, pp. 43–52.

[51] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 89–115, Jan. 2004.

[52] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "TR 00-043 Application of Dimensionality Reduction in Recommender System - A Case Study," *No. TR-00-043. Minnesota Univ Minneap. Dept Comput. Sci.*, 2000.

[53] M. Kurucz, a a Benczur, and K. Csalogány, "Methods for large scale SVD with missing values," *Proc. KDD Cup Work.*, vol. 12, pp. 31–38, 2007.

[54] S. Funk, "Netflix Update: Try This at Home," 2006. [Online]. Available: https://sifter.org/simon/journal/20061211.html.

[55] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-Scale Parallel Collaborative Filtering for the Netflix Prize," in *Algorithmic Aspects in Information and Management*, vol. 5034 LNCS, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–348.

[56] P.-L. Chen *et al.*, "A linear ensemble of individual and blended models for music rating prediction," *KDDCup 2011*, pp. 21–60, Jun. 2011.

[57] H.-F. Yu, C.-J. Hsieh, S. Si, and I. Dhillon, "Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems," in *2012 IEEE 12th International Conference on Data Mining*, 2012, pp. 765–774.

[58] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon, "Parallel matrix factorization for recommender systems," *Knowl. Inf. Syst.*, vol. 41, no. 3, pp. 793–819, Dec. 2014.

[59] W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin, "A Fast Parallel Stochastic Gradient Method for Matrix Factorization in Shared Memory Systems," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 1, pp. 1–24, Mar. 2015.

[60] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, 2011, p. 69.

[61] S. Zubair, N. I. Chaudhary, Z. A. Khan, and W. Wang, "Momentum fractional LMS for power signal parameter estimation," *Signal Processing*, vol. 142, pp. 441–449, Jan. 2018.

[62] S. Cheng, Y. Wei, Y. Chen, Y. Li, and Y. Wang, "An innovative fractional order LMS based on variable initial value and gradient order," *Signal Processing*, vol. 133, pp. 260–269, Apr. 2017.

[63] G. E. Hinton, "Reducing the Dimensionality of Data with Neural Networks," *Science (80-.).*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[64] D.-K. Chae, J. A. Shin, and S.-W. Kim, "Collaborative Adversarial Autoencoders: An Effective Collaborative Filtering Model Under the GAN Framework," *IEEE Access*, vol. 7, pp. 37650–37663, 2019.

[65] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep Learning Based Recommender System," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, Feb. 2019.

[66] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, "A review on deep learning for recommender systems: challenges and remedies," *Artif. Intell. Rev.*, pp. 1–37, Aug. 2018.

[67] M. He, Q. Meng, and S. Zhang, "Collaborative Additional Variational Autoencoder for Top-N Recommender Systems," *IEEE Access*, vol. 7, pp. 5707–5713, 2019.

[68] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized Denoising Autoencoders for Domain Adaptation," *arXiv Prepr. arXiv*, Jun. 2012.

[69] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Nature, 2016.

[70] H. Steck, "Evaluation of recommendations," in *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, 2013, pp. 213–220.

[71] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the fourth ACM conference on Recommender systems - RecSys '10*, 2010, p. 39.

[72] F. Strub, R. Gaudel, and J. Mary, "Hybrid Recommender System based on Autoencoders," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems - DLRS 2016*, 2016, pp. 11–16.

[73] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi, "Sequential Variational Autoencoders for Collaborative Filtering," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining - WSDM '19*, 2019, pp. 600–608.

[74] Y. Ouyang, W. Liu, W. Rong, and Z. Xiong, "Autoencoder-Based Collaborative Filtering," Springer, Cham, 2014, pp. 284–291.

[75] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative Deep Learning for Recommender Systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, 2015, pp. 1235–1244.

[76] X. S. D. Hao Wang, "Supplementary Material : Relational Stacked Denoising Autoencoder for Tag Recommendation," *AAAI*, pp. 3052–3058, Feb. 2015.

[77] X. Li and J. She, "Collaborative Variational Autoencoder for Recommender Systems," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, 2017, pp. 305–314.

[78] H. Ying, L. Chen, Y. Xiong, and J. Wu, "Collaborative Deep Ranking: A Hybrid Pair-Wise Recommendation Algorithm with Implicit Feedback," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9652 LNAI, Springer, Cham, 2016, pp. 555–567.

[79] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," *Proc. twenty-fifth Conf. Uncertain. Artif. Intell.*, pp. 452–461, 2012.

[80] M. S. Aslam, N. I. Chaudhary, and M. A. Z. Raja, "A sliding-window

approximation-based fractional adaptive strategy for Hammerstein nonlinear ARMAX systems." *Nonlinear Dyn.*, vol. 87, no. 1, pp. 519–533, Jan. 2017.

[81] M. A. Z. Raja and N. I. Chaudhary, "Two-stage fractional least mean square identification algorithm for parameter estimation of CARMA systems," *Signal Processing*, vol. 107, pp. 327–339, Feb. 2015.

[82] J. Sabatier, O. P. Agrawal; and J. A. Tenreiro MacHado, *Advances in Fractional Calculus*. Dordrecht: Springer Netherlands, 2007.

[83] J. T. Machado, V. Kiryakova, and F. Mainardi, "Recent history of fractional calculus," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 16, no. 3, pp. 1140–1153, Mar. 2011.

[84] Y.-F. Pu, Z. Yi, and J.-L. Zhou, "Fractional Hopfield Neural Networks: Fractional Dynamic Associative Recurrent Neural Networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 10, pp. 2319–2333, Oct. 2017.

[85] S. M. A. Pahnehkolaei, A. Alfi, and J. A. Tenreiro Machado, "Dynamic stability analysis of fractional order leaky integrator echo state neural networks," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 47, pp. 328–337, Jun. 2017.

[86] S. M. Abedi Pahnehkolaei, A. Alfi, and J. A. T. Machado, "Uniform stability of Fractional Order Leaky Integrator Echo State Neural Network with multiple time delays," *Inf. Sci. (Ny).*, vol. 418–419, pp. 703–716, Dec. 2017.

[87] V. Badri and M. S. Tavazoei, "Some Analytical Results on Tuning Fractional-Order [Proportional–Integral] Controllers for Fractional-Order Systems," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 3, pp. 1059–1066, May 2016.

[88] S. M. A. Pahnehkolaei, A. Alfi, and J. A. Tenreiro Machado, "Chaos suppression in fractional systems using adaptive fractional state feedback control," *Chaos, Solitons & Fractals*, vol. 103, pp. 488–503, Oct. 2017.

[89] M. Rostami and M. Haeri, "Undamped oscillations in fractional-order Duffing oscillator," *Signal Processing*, vol. 107, pp. 361–367, Feb. 2015.

[90] M. D. Ortigueira and J. A. T. Machado, "Fractional calculus applications in signals and systems," *Signal Processing*, vol. 86, no. 10, pp. 2503–2504, Oct. 2006.

[91] M. D. Ortigueira, C. M. Ionescu, J. T. Machado, and J. J. Trujillo, "Fractional signal processing and applications," *Signal Processing*, vol. 107, no. 107, p. 197, Feb. 2015.

[92] B. Safarinejadian, M. Asad, and M. S. Sadeghi, "Simultaneous state estimation and parameter identification in linear fractional order systems using coloured measurement noise," *Int. J. Control*, vol. 89, no. 11, pp. 2277–2296, Nov. 2016.

[93] B. Safarinejadian, M. Asad, and A. T. Jahromi, "A Hierarchical Identification Method for SISO Fractional-order State-space Systems," 2014.

[94] C. Psychalinos, A. S. Elwakil, A. G. Radwan, and K. Biswas, "Guest Editorial: Fractional-Order Circuits and Systems: Theory, Design, and Applications," *Circuits, Syst. Signal Process.*, vol. 35, no. 6, pp. 1807–1813, Jun. 2016.

[95] A. Elwakil, "Fractional-Order Circuits and Systems: An Emerging Interdisciplinary Research Area," *IEEE Circuits Syst. Mag.*, vol. 10, no. 4, pp. 40–50, 2010.

[96] D. Chen, Y. Chen, and H. Sheng, "Fractional Variational Optical Flow Model for Motion Estimation," *Comput. Eng.*, vol. 2010, pp. 22–24, 2000.

[97] D. Chen, S. Sun, C. Zhang, Y. Chen, and D. Xue, "Fractional-order TV-L2 model for image denoising," *Open Phys.*, vol. 11, no. 10, pp. 1414–1422, Jan. 2013.

[98] D. Chen, Y. Chen, and D. Xue, "Fractional-order total variation image denoising based on proximity algorithm," *Appl. Math. Comput.*, vol. 257, pp. 537–545, Apr. 2015.

[99] G.-C. Wu, D. Baleanu, and Z.-X. Lin, "Image encryption technique based on fractional chaotic time series," *J. Vib. Control*, vol. 22, no. 8, pp. 2092–2099, May 2016.

[100] Y. Chen, H. Malek, and S. Dadras, "Fractional order equivalent series resistance modelling of electrolytic capacitor and fractional order failure prediction with application to predictive maintenance," *IET Power Electron.*, vol. 9, no. 8, pp. 1608–1613, Jun. 2016.

[101] H. M. Baskonus and H. Bulut, "On the numerical solutions of some fractional ordinary differential equations by fractional Adams-Bashforth-Moulton method," *Open Math.*, vol. 13, no. 1, Jan. 2015.

[102] H. Bulut, H. M. Baskonus, and Y. Pandir, "The Modified Trial Equation Method for Fractional Wave Equation and Time Fractional Generalized Burgers Equation," *Abstr. Appl. Anal.*, vol. 2013, pp. 1–8, Sep. 2013.

[103] N. I. Chaudhary, M. A. Z. Raja, and A. U. Rehman Khan, "Design of modified fractional adaptive strategies for Hammerstein nonlinear control autoregressive systems," *Nonlinear Dyn.*, vol. 82, no. 4, pp. 1811–1830, Dec. 2015.

[104] M. Geravanchizadeh and S. Ghalami Osgouei, "Speech enhancement by modified convex combination of fractional adaptive filtering," *Iran. J. Electr. Electron. Eng.*, vol. 10, no. 4, pp. 256–266, 2014.

[105] M. A. Z. Raja and N. I. Chaudhary, "Adaptive strategies for parameter estimation of Box–Jenkins systems," *IET Signal Process.*, vol. 8, no. 9, pp. 968–980, Dec. 2014.

[106] N. I. Chaudhary, M. S. Aslam, and M. A. Z. Raja, "Modified Volterra LMS algorithm to fractional order for identification of Hammerstein non-linear system," *IET Signal Process.*, vol. 11, no. 8, pp. 975–985, Oct. 2017.

[107] N. I. Chaudhary, M. A. Manzar, and M. A. Z. Raja, "Fractional Volterra LMS algorithm with application to Hammerstein control autoregressive model identification," *Neural Comput. Appl.*, pp. 1–14, Feb. 2018.

[108] M. D. Ortigueira, "Fractional Calculus for Scientists and Engineers," vol. 84. Springer Netherlands, Dordrecht, p. 154, 2011.

[109] J. J. Shynk and S. Roy, "The LMS algorithm with momentum updating," in *1988., IEEE International Symposium on Circuits and Systems*, 2003, pp. 2651–2654.

[110] F. M. Harper and J. A. Konstan, "The MovieLens Datasets," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Dec. 2015.

[111] K. Li, X. Zhou, F. Lin, W. Zeng, and G. Alterovitz, "Deep Probabilistic Matrix Factorization Framework for Online Collaborative Filtering," *IEEE Access*, vol. 7, pp. 56117–56128, 2019.

[112] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional Matrix Factorization for Document Context-Aware Recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16*, 2016, pp. 233–240.

[113] D. M. Nguyen, E. Tsiligianni, and N. Deligiannis, "Learning Discrete Matrix Factorization Models," *IEEE Signal Process. Lett.*, vol. 25, no. 5, pp. 720–724, May 2018.

[114] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online Learning: A Comprehensive Survey," Feb. 2018.

[115] F. Lin, X. Zhou, and W. Zeng, "Sparse Online Learning for Collaborative Filtering," *Int. J. Comput. Commun. Control*, vol. 11, no. 2, p. 248, Jan. 2016.

[116] J. Huang, F. Nie, and H. Huang, "Robust discrete matrix completion," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, 2013, pp. 424–430.

[117] Z. Huo, J. Liu, and H. Huang, "Optimal discrete matrix completion," in *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2016, pp. 1687–1693.

[118] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep Matrix Factorization Models for Recommender Systems," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 3203–3209.

[119] D. M. Nguyen, E. Tsiligianni, and N. Deligiannis, "Extendable Neural Matrix Completion," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, vol. 2018-April, pp. 6328–6332.

[120] L. Deng, "Deep Learning: Methods and Applications," *Found. Trends® Signal Process.*, vol. 7, no. 3–4, pp. 197–387, 2014.

[121] S. Li, J. Kawale, and Y. Fu, "Deep Collaborative Filtering via Marginalized Denoising Auto-encoder," in *Proceedings of the 24th ACM International on*

Conference on Information and Knowledge Management - CIKM '15, 2015, pp. 811–820.

[122] Y. Pan, F. He, and H. Yu, "Trust-aware Collaborative Denoising Auto-Encoder for Top-N Recommendation," *arXiv Prepr. arXiv1703.01760*, Mar. 2017.

[123] K. Georgiev and P. Nakov, "A non-IID Framework for Collaborative Filtering with Restricted Boltzmann Machines," *Icml*, vol. 28, pp. 1–9, 2013.

[124] X. Jia, X. Li, K. Li, V. Gopalakrishnan, G. Xun, and A. Zhang, "Collaborative restricted Boltzmann machine for social event recommendation," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2016, pp. 402–405.

[125] X. Jia, A. Wang, X. Li, G. Xun, W. Xu, and A. Zhang, "Multi-modal learning for video recommendation based on mobile application usage," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015, pp. 837–842.

[126] X. Liu, Y. Ouyang, W. Rong, and Z. Xiong, "Item Category Aware Conditional Restricted Boltzmann Machine Based Recommendation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9490, Springer, Cham, 2015, pp. 609–616.

[127] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning - ICML '07*, 2007, pp. 791–798.

[128] T. Bansal, D. Belanger, and A. McCallum, "Ask the GRU," in *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16*, 2016, pp. 107–114.

[129] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Deep Coevolutionary Network: Embedding User and Item Features for Recommendation," Sep. 2016.

[130] T. Donkers, B. Loepp, and J. Ziegler, "Sequential User-based Recurrent Neural Network Recommendations," in *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17*, 2017, pp. 152–160.

[131] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based Recommendations with Recurrent Neural Networks," Nov. 2015.

[132] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, "Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16*, 2016, pp. 241–248.

[133] T. Alashkar, S. Jiang, S. Wang, and Y. Fu, "Examples-Rules Guided Deep Neural Network for Makeup Recommendation," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 941–947.

[134] C. Chen, P. Zhao, L. Li, J. Zhou, X. Li, and M. Qiu, "Locally Connected Deep Learning Framework for Industrial-scale Recommender Systems," in *Proceedings*

of the 26th International Conference on World Wide Web Companion - WWW '17 Companion, 2017, pp. 769–770.

[135] H.-T. Cheng *et al.*, "Wide &amp; Deep Learning for Recommender Systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems - DLRS 2016*, 2016, pp. 7–10.

[136] P. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16*, 2016, pp. 191–198.

[137] T. Ebesu and Y. Fang, "Neural Citation Network for Context-Aware Citation Recommendation," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, 2017, pp. 1093–1096.

[138] W.-T. Chu and Y.-L. Tsai, "A hybrid recommendation system considering visual information for predicting favorite restaurants," *World Wide Web*, vol. 20, no. 6, pp. 1313–1331, Nov. 2017.

[139] R. He and J. McAuley, "Ups and Downs," in *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, 2016, pp. 507–517.

[140] R. He and J. McAuley, "VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback," *Thirtieth AAAI Conf. Artif. Intell.*, Oct. 2015.

[141] D. Kim, C. Park, J. Oh, and H. Yu, "Deep hybrid recommender systems via exploiting document context and statistics of items," *Inf. Sci. (Ny).*, vol. 417, pp. 72–87, Nov. 2017.

[142] C. Du, C. Li, Y. Zheng, J. Zhu, and B. Zhang, "Collaborative Filtering With User-Item Co-Autoregressive Models," *Thirty-Second AAAI Conf. Artif. Intell.*, Apr. 2018.

[143] Y. Zheng, C. Liu, B. Tang, and H. Zhou, "Neural Autoregressive Collaborative Filtering for Implicit Feedback," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems - DLRS 2016*, 2016, pp. 2–6.

[144] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A Neural Autoregressive Approach to Collaborative Filtering," *Proc. 33rd Int. Conf. Mach. Learn. PMLR 48764-773*, pp. 764–773, May 2016.

[145] S.-Y. Chen, Y. Yu, Q. Da, J. Tan, H.-K. Huang, and H.-H. Tang, "Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18*, 2018, pp. 1187–1196.

[146] S. Choi, H. Ha, U. Hwang, C. Kim, J.-W. Ha, and S. Yoon, "Reinforcement Learning based Recommender System using Biclustering Technique," Jan. 2018.

[147] I. Munemasa, Y. Tomomatsu, K. Hayashi, and T. Takagi, "Deep reinforcement learning for recommender systems," in *2018 International Conference on Information and Communications Technology (ICOIACT)*, 2018, vol. 2018-Janua, pp. 226–233.

[148] X. Wang, Y. Wang, D. Hsu, and Y. Wang, "Exploration in Interactive Personalized Music Recommendation," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 11, no. 1, pp. 1–22, Sep. 2014.

[149] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *Proceedings of the 12th ACM Conference on Recommender Systems - RecSys '18*, 2018, pp. 95–103.

[150] J. Han, X. Cai, and L. Yang, "Generative Adversarial Network based Heterogeneous Bibliographic Network Representation for Personalized Citation Recommendation," in *[AAAI2018]Proceedings of the Thirtiy-second AAAI Conference on Artificial Intelligence*, 2018, no. 2016, pp. 5747–5754.

[151] X. He, Z. He, X. Du, and T.-S. Chua, "Adversarial Personalized Ranking for Recommendation," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval - SIGIR '18*, 2018, pp. 355–364.

[152] J. Wang *et al.*, "IRGAN," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, 2017, pp. 515–524.

[153] Q. Wang, H. Yin, Z. Hu, D. Lian, H. Wang, and Z. Huang, "Neural Memory Streaming Recommender Networks with Adversarial Training," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18*, 2018, pp. 2467–2475.

[154] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive Collaborative Filtering," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, 2017, pp. 335–344.

[155] Y. Gong and Q. Zhang, "Hashtag recommendation using attention-based convolutional neural network," in *IJCAI International Joint Conference on Artificial Intelligence*, 2016, pp. 2782–2788.

[156] Y. Jhamb, T. Ebesu, and Y. Fang, "Attentive Contextual Denoising Autoencoder for Recommendation," in *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval - ICTIR '18*, 2018, pp. 27–34.

[157] Y. Li, T. Liu, J. Jiang, and L. Zhang, "Hashtag Recommendation with Topical Attention-Based LSTM," in *Proceedings of the 26th International Conference on Computational Linguistics*, 2016, pp. 943–952.

[158] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data*

Mining - KDD '18, 2018, pp. 1831–1839.

[159] X. Chen, Y. Zhang, Q. Ai, H. Xu, J. Yan, and Z. Qin, "Personalized Key Frame Recommendation," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, 2017, pp. 315–324.

[160] J. Gao, P. Pantel, M. Gamon, X. He, and L. Deng, "Modeling Interestingness with Deep Neural Networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 2–13.

[161] H. Lee, Y. Ahn, H. Lee, S. Ha, and S. Lee, "Quote Recommendation in Dialogue using Deep Neural Network," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16*, 2016, pp. 957–960.

[162] C. Lei, D. Liu, W. Li, Z.-J. Zha, and H. Li, "Comparative Deep Learning of Hybrid Representations for Image Recommendations," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, vol. 2016-Decem, pp. 2545–2553.

[163] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam, "Neural Rating Regression with Abstractive Tips Generation for Recommendation," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, 2017, pp. 345–354.

[164] Y. S. Rawat and M. S. Kankanhalli, "ConTagNet," in *Proceedings of the 2016 ACM on Multimedia Conference - MM '16*, 2016, pp. 1102–1106.

[165] Y. Song, A. M. Elkahky, and X. He, "Multi-Rate Deep Learning for Temporal Recommendation," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16*, 2016, pp. 909–912.

[166] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative Knowledge Base Embedding for Recommender Systems," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016, pp. 353–362.

[167] H. Wang, X. Shi, and D.-Y. Yeung, "Collaborative Recurrent Autoencoder: Recommend while Learning to Fill in the Blanks," *Adv. Neural Inf. Process. Syst. 29*, pp. 415–423, Nov. 2016.

[168] Y. Chen and M. de Rijke, "A Collective Variational Autoencoder for Top-N Recommendation with Side Information," in *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems - DLRS 2018*, 2018, pp. 3–9.

[169] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational Autoencoders for Collaborative Filtering," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, 2018, pp. 689–698.

[170] S. Zhang, L. Yao, and X. Xu, "AutoSVD++," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, 2017, pp. 957–960.

[171] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008, pp. 1096–1103.

[172] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *J. Mach. Learn. Res.*, vol. 11, no. 3, pp. 3371–3408, 2010.

[173] M. Chen, K. Weinberger, F. Sha, and Y. Bengio, "Marginalized Denoising Auto-encoders for Nonlinear Representations," in *International Conference on Machine Learning*, 2014, vol. 32, pp. 1476–1484.

[174] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv Prepr. arXiv*, Dec. 2014.

[175] J. Tang and K. Wang, "Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining - WSDM '18*, 2018, vol. 2018-Febua, pp. 565–573.

[176] S. Zhang, Y. Tay, L. Yao, and A. Sun, "Next Item Recommendation with Self-Attention," *AAAI*, 2018.

[177] A. Atangana, "On the new fractional derivative and application to nonlinear Fisher's reaction–diffusion equation," *Appl. Math. Comput.*, vol. 273, pp. 948–956, Jan. 2016.

[178] A. Atangana and D. Baleanu, "Caputo-Fabrizio Derivative Applied to Groundwater Flow within Confined Aquifer," *J. Eng. Mech.*, vol. 143, no. 5, p. D4016005, May 2017.

[179] A. Atangana and J. F. Gómez-Aguilar, "Fractional derivatives with no-index law property: Application to chaos and statistics," *Chaos, Solitons & Fractals*, vol. 114, pp. 516–535, Sep. 2018.

[180] X.-J. Yang, F. Gao, J. A. Tenreiro Machado, and D. Baleanu, "A new fractional derivative involving the normalized sinc function without singular kernel," *Eur. Phys. J. Spec. Top.*, vol. 226, no. 16–18, pp. 3567–3575, Dec. 2017.

[181] M. A. Firoozjaee, H. Jafari, A. Lia, and D. Baleanu, "Numerical approach of Fokker–Planck equation with Caputo–Fabrizio fractional derivative using Ritz approximation," *J. Comput. Appl. Math.*, vol. 339, pp. 367–373, Sep. 2018.

[182] E. J. Solteiro Pires, J. A. Tenreiro Machado, P. B. de Moura Oliveira, J.

Boaventura Cunha, and L. Mendes, "Particle swarm optimization with fractional-order velocity," *Nonlinear Dyn.*, vol. 61, no. 1–2, pp. 295–301, Jul. 2010.

[183] M. S. Couceiro, R. P. Rocha, N. M. F. Ferreira, and J. A. T. Machado, "Introducing the fractional-order Darwinian PSO," *Signal, Image Video Process.*, vol. 6, no. 3, pp. 343–350, Sep. 2012.

[184] Y. Mousavi and A. Alfi, "Fractional calculus-based firefly algorithm applied to parameter estimation of chaotic systems," *Chaos, Solitons & Fractals*, vol. 114, pp. 202–215, Sep. 2018.

[185] S. Chen and Y. Peng, "Matrix factorization for recommendation with explicit and implicit feedback," *Knowledge-Based Syst.*, vol. 158, pp. 109–117, Oct. 2018.

[186] X. Ning and G. Karypis, "Multi-task learning for recommender systems," in *Journal of Machine Learning Research*, 2010, vol. 13, pp. 269–284.