Meta-Learning for User Cold-Start Recommendation

Homanga Bharadhwaj
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Kanpur, India
homangab@cse.iitk.ac.in

***Abstract*—Recent studies in recommender systems emphasize the importance of dealing with the cold-start problem i.e. the modeling of new users or items in the recommendation system. Meta-learning approaches have gained popularity recently in the Machine Learning (ML) community for learning representations useful for a wide-range of tasks. Inspired by the generalizable modeling prowess of Model-Agnostic Meta Learning, we design a recommendation framework that is trained to be reasonably good enough for a wide range of users. During testing, to adapt to a specific user, the model parameters are updated by a few gradient steps. We evaluate our approach on three different benchmark datasets, from Movielens, Netflix, and MyFitnessPal. Through detailed simulation studies, we show that this framework handles the user cold-start model much better than state-of-the art benchmark recommender systems. We also show that the proposed approach performs well on the task of general recommendations to non cold-start users and effectively takes care of routine and eclectic preference trends of users.**

***Index Terms*—meta-learning, cold-start problem, recommender systems, deep neural network**

I. Introduction

Recommendation systems (RS) are ubiquitous now and the problem of recommending items to users is fairly common across numerous web-platforms [1]. Content-based filtering and collaborative filtering (CF) are two of the broad approaches adopted in most recommender systems. Content-based filtering relies on properties (features) of each item (for e.g. in case of movies, features can include its genre, year of release, names of the actors, a plot synopsis, an advertisement image etc.) [2] while collaborative filtering is based on inferring correlations between users and items - similar users prefer similar items and vice versa [3], [4]. Most modern RS are based on CF in part due to its higher accuracy and also because content-based methods require a lot of ad-hoc features about items which may not be easily available [1], [4].

The most popular CF method is called Matrix Factorization [5], [6], which models each user and item by individual vectors (of learned features) and computes a similarity score like dot-product for inferring the preference of users for items. These methods typically work well in the presence of large data and are notorious for performing poorly in cold-start recommendations [7], [8]. The user cold-start problem refers to the task of recommending items to a new user, whose previous item preferences are not present in the system. This is one of the most important challenges in RS and several methods have been proposed to tackle it [7], [9], [10]. However, they are not successful in tackling it robustly, especially in cases of sparse feedback - when there are large numbers of users and items and only a few users rate (or consume) each item [7], [8].

In this paper, we tackle the problem of user cold-start in RS, by applying model-agnostic meta learning (MAML) [11]. MAML is a general and very powerful technique proposed for meta-learning of deep neural networks and through this work we demonstrate its applicability in mitigating the cold-start problem. There have been a few existing works on applying meta-learning to recommender sustems, namely [8] and [12]. [8] considers the problem of item cold-start which has only very specific applications like in situations for tweet / facebook post recommendation. In most situations like movie, music, products recommendation, the problem of user cold-start is more acute, since there are many users who rate / consume a new item very quickly thus providing some signals for recommendation. Also, their method does not provide any guarantees on the speed at which test-time computations can be done. We benchmark our model against [8] and show that our method requires only a few gradient updates during test-time, for it to rapidly adapt to a new user. [12] does not consider the cold-start problem and focuses on the issue of distributed training and privacy preservation. Our method is a super-set of this technique in the sense that it alleviates the cold-start problem while maintaining the claims of privacy-preservation described in [12].

The key insight behind the design of our framework is to train a recommendation model offline that is suitable for a wide range of users. When a new user, who has rated /consumed only a few items comes along, our model requires very few gradient updates to its parameters for it to be suitable for recommending items to the new user. We specifically avoid overfitting the model on the new user by formulating the meta-learning problem through a series of tasks, wherein each task involves inferring which items in a group of items are relevant to a particular user [11], [13]. Through detailed simulation on standard Movielens [14], Netflix [15], and MyFitnessPal [16] datasets, we demonstrate the efficacy of our method in handling the cold-start problem, while maintaining

high recommendation accuracy.

II. Related Works

Recommendation systems based on deep learning models have become very popular in the recent past. Several deep learning techniques like Recurrent Neural Networks [17]–[19], Variational Auto Encoders [20], Denoising Auto Encoders [21], Restricted Boltzmann Machines [22], Generative Adversarial Networks [23], [24], and Convolutional Neural Networks [25], [26] have been used to build recommender system that perform very well on benchmark datasets and also produce reliable recommendations in practice. However, none of these methods explicitly tackle the cold-start problem which is very relevant in practical recommender systems that constantly witness new users joining the system and new items being added to the database.

There have been some recent attempts to address the cold-start problem in recommendations: [8] focuses on the item-cold start problem by using a meta-learning strategy, [27] also tackles the item-cold start problem, but adopts an active learning framework to balance exploration-exploitation trade-offs, [10] uses doc2vec for mitigating the cold-start problem of new items, while [28] uses demographic information of users for solving the user cold-start problem.

However, there are numerous drawbacks of these works: [8] is computationally very expensive as it requires training of two separate neural networks, the first network learns a common representation of all items, while the second network learns a single representation per class. Also, test-time predictions are time consuming as the model is not compatible with recent gradient based meta-learning approaches [11] that require only a few gradient updates for rapid adaptation. [27] requires a lot of additional attributes of items for solving the cold-start problem and hence suffers from similar drawbacks of content-based filtering. This method cannot be used in situations of sparse feedback, where item attributes are not available. [10] focuses on the specific task of job recommendation and is not generalizable to other recommendation situations. The method in [28] relies on sensitive demographic information of users, which may not be always available for solving the cold-start problem. In addition, the prediction model is based on simple hand-crafted features and is not generalizable to different tasks. The RNN-GAN based approach in RecGAN [9] has been empirically shown to do well in the cold-start recommendation problem but there is no theoretical motivation for the same.

Meta-learning is a recently popularized paradigm for training easily generalizable machine learning models that are capable of *learning to learn*. While the basic understanding behind meta-learning has existed in the Artificial Intelligence and Machine Learning communities for a long time [29], [30], recent advances in deep learning has pushed research in this front significantly. It has been used in a myriad of fields ranging from robotics [31] to speech recognition [32]. Older approaches to meta-learning train memory-augmented models on a plethora of tasks, by training a recurrent learner to adapt to an ensemble of tasks [33], [34]. However, recurrent architecture based models are bulky, and the major thrust of modern meta-learning research has been towards low cost gradient-based methods [35]. These methods learn a better initialization of the meta-learner's parameters such that they can be easily updated through a few gradient steps to adapt to any task [11], [36], [37].

In this work we leverage recent advances in meta-learning [11], [13], [38] that are based on gradient descent for rapid test-time predictions. Our method does not require any additional information apart from user id, item id and the ratings (or consumption information) of users on items. Our method is task-agnostic, in the sense that it can be used for any recommendation task and does not rely on collecting sensitive information from users.

III. The Proposed Approach

Inspired by other algorithm formulations in the domain of recommender systems [8], we consider the task of recommendation to be a binary classification problem for each user. Each item is either *preferred* (label 1) or *not preferred* (label 0) by an user. For recommendation situations, where each user rates items on a scale (say 0 to 10 scale), then by thresholding (say rating value < 6 is considered *not preferred* while rating value > 6 is considered *preferred*), the binary classification scheme can be reliably applied. In situations, where the feedback mechanism is not explicit via ratings, preference can be elicited by measuring *engagement*, for example in music recommendation, engagement can be defined as the user listening to a particular song.

Let $\mathcal{U}$ denote the set of all users and $\mathcal{I}$ denote the set of all items. The prediction task in this model is to infer the probability of user u_i preferring item v_j:

$$\mathbb{P}(r_{ij} = 1 | u_i, v_j) \quad \forall i \in \mathcal{U} \quad \forall j \in \mathcal{I} \tag{1}$$

Here, $r_{ij} \in \{0, 1\}$ denotes the rating of user u_i on item v_j. Using this, users can be recommended items whose probability of being preferred by the user are highest. For meta-learning, we need to formulate the definition of each "task" in the context of our problem. Since the aim of our meta-learning strategy is to learn a recommendation model, *general enough* for all users, that by minor fine tuning can adapt to specific users, we define each task $\mathcal{T}_u$ to be the problem of recommending items to one user given some examples of both positive and negative examples i.e. some items preferred and some not preferred by the user.

We parametrize the recommendation algorithm by ϕ and denote it by $\mathcal{F}_\phi(\cdot)$. We use this to model the probability of preference of item v_j for user u_i:

$$\mathbb{P}(r_{ij} = 1 | u_i, v_j) = \mathcal{F}_\phi(u_i, v_j) \tag{2}$$

$\mathcal{F}_\phi(\cdot)$ in general can be any function approximator. In the Experiments section, we consider $\mathcal{F}_\phi(\cdot)$ to be a linear model, a single layer feedforward neural network and a deep feedforward neural network and perform comparative analyses on their performance.

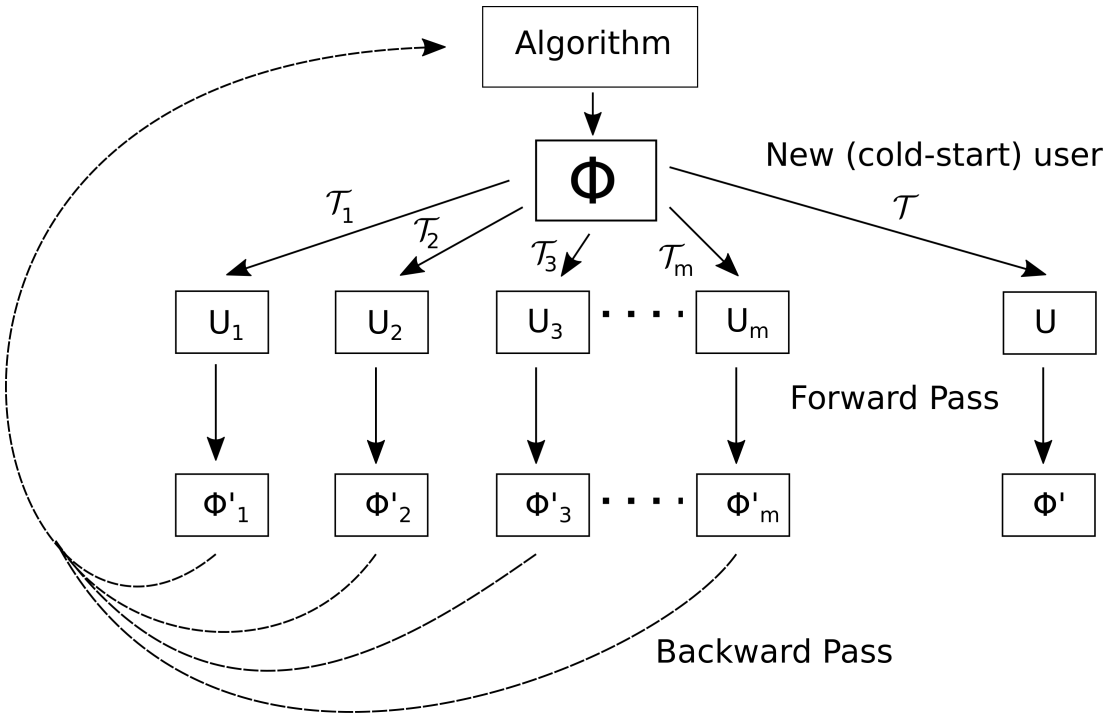


Fig. 1: The overall meta-learning scheme for recommendation. The forward pass is used to update the task-specific parameters by a few gradient updates. The backward pass is used to update the parameter of the algorithm based on the test losses from each task. During recommendation for a new (cold-start) user, only a few gradient updates on the overall parameters of the algorithm are required to adapt the model for recommending items to the new user.

Algorithm 1 Training: Meta-Learning for Recommendation

Require: $p(\mathcal{T})$: Task distribution
Require: Fix the value of β_2: meta-step size
Randomly initialize θ and β_1
while not done **do**
  Sample batch $\mathcal{B}$ of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ in $\mathcal{B}$ **do**
    Sample K_1 data samples $\mathscr{D}_i = \{v_j, r_{ij}\}$ from $\mathcal{T}_i$
    Evaluate $\nabla_\phi \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_\phi)$ using $\mathscr{D}_i$ and $\mathscr{L}_{\mathcal{T}_i}$ as described in Section III-B
    Compute gradient updates for parameters of task $\mathcal{T}_i$: $\phi_i^{'} = \phi - \beta_1 \nabla_\phi \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_\phi)$
    Sample K_2 data samples $\mathscr{D}_i^{'} = \{v_j, r_{ij}\}$ from $\mathcal{T}_i$ for meta-update
  end for
  Update $\phi = \phi - \beta_2 \nabla_\phi \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_{\phi_i^{'}})$ using each $\mathscr{D}_i'$ and $\mathscr{L}_{\mathcal{T}_i}$ as described in Section III-C
  Update $\beta_1 = \beta_1 - \beta_2 \nabla_{\beta_1} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_{\phi_i^{'}})$ using each $\mathscr{D}_i'$ and $\mathscr{L}_{\mathcal{T}_i}$ as described in Section III-C
end while

Algorithm 2 Testing: Meta-Learning for Recommendation

Require: $p(\mathcal{T})$: Task distribution
Require: The trained value of step size β_1 and ϕ
Sample batch $\mathcal{B}_{test}$ of new tasks $\mathcal{T}_i \sim p(\mathcal{T})$
//These new tasks correspond to new (cold-start) users
for all $\mathcal{T}_i$ in $\mathcal{B}_{test}$ **do**
  Let $\mathscr{D}_i = \{v_j, r_{ij}\}$ be the set of items rated by the cold-start user u_i (Task $\mathcal{T}_i$)
  Evaluate $\nabla_\phi \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_\phi)$ using $\mathscr{D}_i$ and $\mathscr{L}_{\mathcal{T}_i}$ as described in Section III-B
  Compute gradient updates for parameters of task $\mathcal{T}_i$: $\phi_i^{'} = \phi - \beta_1 \nabla_\phi \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_\phi)$
end for

A. Task description

Let $p(\mathcal{T})$ denote the probability distribution over all tasks. Each task $\mathcal{T}_i$ generates K_1 i.i.d. positive and negative items from the dataset $\mathcal{D}$ for user u_i (training set for $\mathcal{T}_i$) and K_2 i.i.d. positive and negative items (test set for $\mathcal{T}_i$) . The dataset $\mathcal{D} = (u_i, v_j, r_{i,j})$ consists of (user id, item id, rating) tuples for every user-item pair. Task $\mathcal{T}_i$ corresponds to user u_i. A new user (user cold-start) is denoted by a new task $\mathcal{T}$.

The problem description of each task $\mathcal{T}_i$ is to infer the ratings of each of the K_2 items in the test set for user u_i. When adapting the recommendation model to task $\mathcal{T}_i$, the model's parameters ϕ become $\phi_i^{'}$. Similar to MAML, $\phi_i^{'}$ is computed from ϕ by a few gradient updates based on the loss function value for task $\mathcal{T}_i$.

B. Optimization for each task

Here, we describe the inner loop optimization of Algorithm 1. We first sample a batch of tasks from $p(\mathcal{T})$, and then for each task $\mathcal{T}_i$, we update the recommendation model $\mathcal{F}_\phi(\cdot)$ by one or more gradient updates. The idea is to train the model $\mathcal{F}_\phi(\cdot)$ to be general enough for all tasks, such that by only a few gradient updates, it can be made good enough for a new task $\mathcal{T}_i$, which corresponds to a new (cold-start) user. The gradient update for task $\mathcal{T}_i$ can be described as follows:

$$\phi_i^{'} = \phi - \beta_1 \nabla_\phi \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_\phi) \tag{3}$$

The above denotes one gradient update, but in practice we found that 10 to 20 updates perform best. These implementation details are mentioned in the Experiments section. The step size β is a hyper-parameter which we meta-learn in the outer loop of optimization as described in the next sub-section.

Since we have formulated the recommendation problem as binary-classification, we define the loss function $\mathscr{L}_{\mathcal{T}_i}$ to be a cross-entropy loss of the following form:

$$\mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_\phi) = \sum_{v_j, r_{i,j} \sim \mathcal{T}_i} [r_{ij} \log \mathcal{F}_\phi(u_i, v_j) + (1 - r_{ij}) \log(1 - \mathcal{F}_\phi(u_i, v_j))]$$

As described in sub-section III-A, we sample K_1 i.i.d. positive and negative items ($v_j, r_{i,j} \sim \mathcal{T}_i$) for the above loss function during training. We then sample K_2 i.i.d. positive and negative items ($v_j, r_{i,j} \sim \mathcal{T}_i$) for the meta-update (which corresponds to test of particular tasks) as described in the next sub-section.

C. Meta-optimization

The meta-optimization corresponds to updates of the parameters ϕ based on the test loss for all tasks $\mathcal{T}_i \sim p(\mathcal{T})$. As is standard in gradient-based meta learning techniques like MAML, the meta-objective is formalized as:

$$\min_\phi \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_{\phi_i^{'}}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathscr{L}_{\mathcal{T}_i}\left(\mathcal{F}_{\phi - \beta \nabla_\phi \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_\phi)}\right) \tag{4}$$

It is important to take note of the fact that meta-optimization is executed over parameters of the recommendation model ϕ, while the objective function above is calculated using the individual parameters $\phi_i^{'}$ of each task $\mathcal{T}_i$. We perform the meta-optimization by ADAM [39] or SGD [40] methods, based on prior recommender system models that yield good performance using these methods [1]. Any other gradient-descent based optimization scheme can also be used here. If SGD is used, than the gradient update step can be described as follows:

$$\phi = \phi - \beta_2 \nabla_\phi \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_{\phi_i^{'}}) \tag{5}$$

Here, β_2 is the meta-step size which is a hyperparameter fixed apriori. In addition to θ, we also meta-update the value of β_1, which is the step size for updating the gradient of each task.

$$\beta_1 = \beta_1 - \beta_2 \nabla_{\beta_1} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathscr{L}_{\mathcal{T}_i}(\mathcal{F}_{\phi_i^{'}}) \tag{6}$$

Learning the value of β_1 from data in this fashion makes our method more robust and less susceptible to getting trapped in local optima due to improper initialization of its value.

IV. Experiments

For notational convenience, we name our model MetaCS (Short for Meta Learning based Cold-Start Recommender System).

We evaluate our method on the tasks of movie recommendation and food recommendation by performing experiments on three benchmark datasets, namely Movielens 100k [14], Netflix Challenge, [15] and MyFitness Pal [16]. Through our experiments, we seek to answer three major research questions (RQs):

- **RQ1:** How much (quantitatively) does the proposed meta-learning based approach mitigate the user cold-start problem in recommender systems?
- **RQ2:** Does the proposed method also help in improving the recommendation performance of non cold-start users (those that have been using the system for a long time)?
- **RQ3:** How well does the proposed model generalize as compared to baselines for users that have diverse (eclectic) or routine preferences?

To answer these questions we do a comparative analysis of our method against state-of-the art baseline recommendation models on all the three datasets.

A. Baselines

For comparative analysis, we consider different benchmark models as well as variations of our proposed approach. For our model, we specifically consider the following variations of $\mathcal{F}_\phi$:

- **A Linear model:** $\mathcal{F}_\phi(u_i, v_j) = \sigma(W_0 u_i + W_1 v_j + b)$. Here, $\phi = \{W_0, W_1, b\}$ denote the trainable weights of the linear regression model. The sigmoid function $\sigma(\cdot)$ ensures that the final output is squashed in the range $(0, 1)$ so as to be representative of a probability value, $\mathbb{P}(r_{ij} = 1 | u_i, v_j)$. This model is denoted as MetaCS-L.
- **A non-linear SVM:** This is a very simple non-linear model. We use a Gaussian Radial Basis Function (RBF) as the kernel for non-linear classification in SVM. The specific form of this kernel is:

$$k(u_i, v_j) = \exp\left(-\gamma |u_i - v_j|^2\right) \tag{7}$$

  Here, $\gamma > 0$ is a hyperparameter tuned by grid-search. This model is denoted as MetaCS-NN.
- **Deep Neural Network:** We perform analyses with DNNs of different depths to understand the tradeoffs between depth of the network and recommendation accuracy. The networks are feedforward models with sigmoid non-linearity. In the subsequent sections, we denote by MetaCS-DNN, the DNN based model with depth and

width optimized for best performance as described in Section IV-H.

Apart from these variations, we consider other benchmark models, namely Probabilistic Matrix Factorization (PMF) [6], RecGAN [23], IRGAN [24], MetaNeurIPS [8] and T-SVD++ [41]. Although all results are reported on the test data, for selection of network configuration only the training data was used. In particular, we employed k-fold cross validation with $k = 5$.

B. Datasets

Here we discuss the statistics of the three datasets and the relevant pre-processing used for our experiments.

- **Movielens:** The Movielens 100k dataset contains 100,000 ratings for 1682 movies from 943 users for the seven month duration from September, 1997 to April, 1998. Here, each user has rated atleast 20 movies. Out of these, 85% users' data are randomly selected for the training set and the remaining users' data is used during testing alone.
- **Netflix:** The Netflix Prize dataset contains 100 million ratings from 480 thousand anonymized Netflix users for 17 thousand movie titles collected between October, 1998 and December, 2005. The ratings are integral values within the range 1 to 5. We denote a rating value > 3 as an indication of preference (i.e. binary label 1), and < 0 as binary label 0 in our evaluation. 85% users are randomly selected for the training set and the remaining users' data is used during testing alone.
- **MyFitnessPal (MFP):** The MFP dataset contains 587,187 days of food diary records logged by 9,900 MFP users from September, 2014 through April, 2015. If an user consumed an item, we assign this the binary rating 1, while for items not consumed the binary rating is 0. In our evaluation, 85% users are randomly selected for the training set and the remaining users' data is used during testing alone.

C. Evaluation Metrics

We use the following standard metrics which are widely used in the Machine Learning (ML) and Information Retrieval (IR) communities:

- **Precision@k:** This refers to the number of correct recommendations in the top k recommendations of the model [42].
- **AUROC:** This refers to the area under the Receiver Operating Characteristic (ROC) curve [43], [44] for the task of recommendation over all test users [42].
- **Mean Reciprocal Rank (MRR):** This is defined as the multiplicative inverse of the rank of the first relevant recommendation, averaged across all recommendation tasks [42].

D. Setup

We implement the proposed approach by using the Pytorch library [45] in Python 3 and use ADAM optimizer [39] for training in the meta-udate step (Algorithm 1). The only hyper-parameter in the model is β_2, which is tuned by a grid-search [46] with 10 points in the range $(0, 1)$ via k-fold (where k =5) cross validation [47] on the training set. The optimal value of β_2 for MetaCS-L, MetaCS-NN, and MetaCS-DNN respectively are 0.5, 0.6, and 0.4. We apply dropout [48] for effectively training the MetaCS-DNN model. The value of p for dropout is kept at 0.6.

We implement the baseline models PMF, MetaNeurIPS, IRGAN, RRN, T-SVD++ and RecGAN based on open-sourced codes and details provided in the respective papers. We evaluate these models on the three datasets and fine-tune the hyper-parameters for each model to evaluate their optimal performance on each dataset.

E. Cold-Start Recommendation (RQ1)

Here, during training, we exclude the data from 15% of the users. We then evaluate the trained model on the 15% of users, who essentially are 'new' to the system. Table I illustrates the performance of different models on the test-set of new (cold-start) users. It is evident that MetaCS-DNN i.e. our proposed approach with a Deep Neural Network as the recommendation algorithm performs much better than all other benchmark models. It is interesting to note that the performance with even a single-layer Neural Network, MetaCS-NN is much better than other other benchmark models. This is indicative of the fact that our overall meta-learning framework is effective in dealing with the cold-start problem irrespective of the depth of the function approximator used. Hence, the proposed approach is robust.

Figure 2 shows the variation of Precision@1 with the number of gradient updates required to achieve optimal performance during testing. This corresponds to the testing Algorithm 2 described in Section III. It is evident that only a few gradient updates (less than 30) are required to fine-tune the trained model (Algorithm 1) to adapt to a new cold-start user during testing. This indicates that the proposed approach is fast enough to be deployed in online interactive recommender systems to mitigate the user cold-start problem.

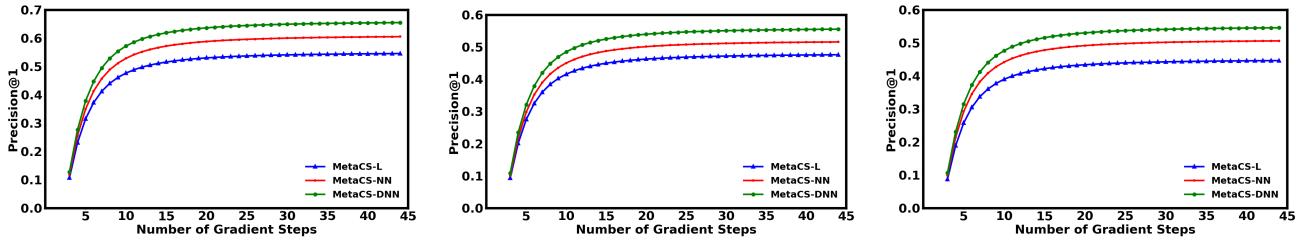
F. Performance in non cold-start scenario (RQ2)

In this section we aim to establish that the proposed MetaCS appraoch is also effective in the general recommendation setting, i.e. in recommending items to non cold-start users who have been using the system for a long time. For training, we consider 85% of the data in each dataset such that some items per user are left held-out during training. The remaining 15% of the data consists of ratings on held-out items corresponding to each user. The recommedation task is to infer the ratings of users on these held-out items.

Table II illustrates the performance of the proposed model in comparison to the performance of the baseline models. It is evident that the proposed model, in partiular MetaCS-DNN outperforms all the other baseline models in all the metrics. This strongly indicates that out proposed approach does not

TABLE I: Evaluation of MetaCS against benchmark models for cold-start recommendation. All values reported are on the test set for new users. The values of Precision@k, AUROC, and MRR lie in the range $(0, 1)$. Higher is better for all metrics.

Dataset	Metrics	MetaCS-L	MetaCS-NN	MetaCS-DNN	PMF	T-SVD++	RecGAN	RRN	MetaNeurIPS	IRGAN
MovieLens 100k	*Precision@1*	0.55	0.61	**0.66**	0.53	0.51	0.56	0.53	0.57	0.52
	Precision@3	0.46	0.51	**0.54**	0.42	0.43	0.45	0.44	0.44	0.42
	AUROC	0.85	0.87	**0.92**	0.86	0.80	0.85	0.85	0.83	0.79
	MRR	0.49	**0.58**	**0.58**	0.45	0.49	0.51	0.50	0.47	0.46
Netflix Prize	*Precision@1*	0.48	0.52	**0.55**	0.45	0.45	0.51	0.46	0.46	0.45
	Precision@3	0.38	0.43	**0.46**	0.36	0.35	0.40	0.40	0.41	0.38
	AUROC	0.84	0.85	**0.91**	0.77	0.75	0.83	0.82	0.78	0.77
	MRR	0.48	0.55	**0.57**	0.45	0.46	0.48	0.49	0.47	0.45
MyFitnessPal	*Precision@1*	0.45	0.51	**0.56**	0.43	0.41	0.46	0.44	0.46	0.42
	Precision@3	0.36	0.42	0.42	0.37	0.39	**0.43**	0.42	0.37	0.35
	AUROC	0.75	0.77	**0.82**	0.70	0.70	0.74	0.75	0.73	0.74
	MRR	0.39	0.38	**0.48**	0.34	0.33	0.39	0.40	0.35	0.36



(a) Movielens 100k (b) Netflix Prize (c) MyFitnessPal

Fig. 2: Variation of Precision@1 during testing with the number of new user specific gradient updates. It is evident that only a small number of gradient updates are required to achieve convergence to a high Precision value.

TABLE II: Evaluation of MetaCS against benchmark models for non cold-start recommendation situations. All values reported are on the test set for held-out ratings of existing users. The values of Precision@k, AUROC, and MRR lie in the range $(0, 1)$. Higher is better for all metrics.

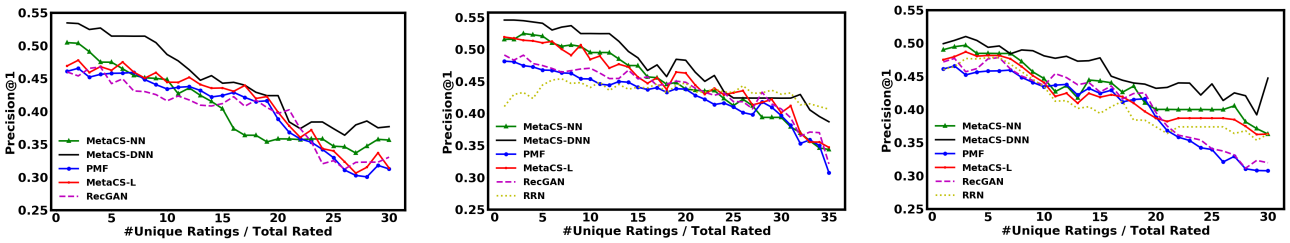
Dataset	Metrics	MetaCS-L	MetaCS-NN	MetaCS-DNN	PMF	T-SVD++	RecGAN	RRN	MetaNeurIPS	IRGAN
MovieLens 100k	*Precision@1*	0.62	0.65	**0.69**	0.59	0.57	0.61	0.63	0.60	0.59
	Precision@3	0.56	0.61	**0.64**	0.53	0.54	0.55	0.55	0.56	0.53
	AUROC	0.87	0.89	**0.93**	0.88	0.83	0.86	0.87	0.88	0.83
	MRR	0.59	0.67	**0.69**	0.56	0.59	0.62	0.60	0.57	0.56
Netflix Prize	*Precision@1*	0.51	0.54	**0.59**	0.48	0.49	0.51	0.51	0.50	0.48
	Precision@3	0.39	0.44	**0.46**	0.37	0.35	0.40	0.37	0.38	0.38
	AUROC	0.85	0.89	**0.92**	0.79	0.82	0.84	0.84	0.79	0.78
	MRR	0.52	0.58	**0.62**	0.47	0.49	0.50	0.48	0.47	0.47
MyFitnessPal	*Precision@1*	0.47	**0.56**	**0.56**	0.44	0.42	0.46	0.46	0.45	0.43
	Precision@3	0.39	0.45	**0.48**	0.35	0.35	0.39	0.41	0.40	0.38
	AUROC	0.73	0.76	**0.82**	0.70	0.72	0.74	0.74	0.74	0.71
	MRR	0.41	0.43	**0.48**	0.36	0.34	0.39	0.41	0.39	0.38

optimize cold-start recommendation at the cost of non cold-start user behavior. Hence, the proposed approach performs well both in the cold-start scenario and also in the general recommendation setting.

G. Routine vs. eclectic users (RQ3)

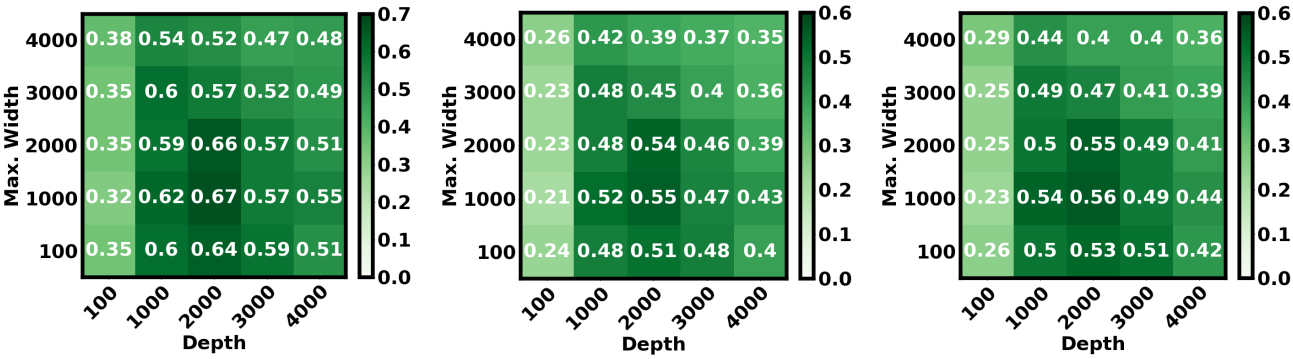
In this section, we perform analyses of recommendation accuracy with respect to different user-behaviors. It has been well-established that some users are more routine in their preferences, while others tend to prefer a diverse range of items and hence are eclectic in their choices [18], [19], [23]. To this end, we analyze the behavior of recommendation accuracy with respect to the ratio of number of unique items rated (or consumed) by an user and the total number of ratings (or consumptions) of that user. We then averaage across users for each dataset. This analysis is akin to similar analyses done in benchmark papers on recommender systems [18], [19], [23].

Figure 3 displays a distinct decreasing trend as the ratio of the number of unique items rated to the total number of items rated. This should be expected because of a search-space blowup owing to large combinations of potentially consumable items. In this plot, routine behavior is indicated by a small ratio on the x-axis because it represents relatively invariant user preferences. In contrast, eclectic behavior is indicated by



(a) Movielens 100k (b) Netflix Prize (c) MyFitnessPal

Fig. 3: Variation of Precision@1 during testing with the ratio of number of unique items rated and the total items rated by each user, avergaed over all users in the corresponding dataset. Low values in the x-axis correspond to routine users while high values in the x-axis correspond to eclectic users.



(a) Movielens 100k (b) Netflix Prize (c) MyFitnessPal

Fig. 4: Variation of Precision@1 during testing with the height and width of the DNN used in the proposed approach MetaCS-DNN.

a diverse consumption of items and hence a large ratio on the x-axis. From Figure 3 it is evident that MetaCS-DNN has high values of Precision@1 both in the low ratio regions and high ratio regions of the plot. This indicates that the proposed model is capable of modeling both routine and diverse preference trends of users.

We postulate that MetaCS's greater robustness in handling these preference trends of users is due to the meta-learning framework that learns to generalize well to different users. The model learnt is general enough to be readily suitable for a wide range of users who have different preference trends.

H. Depth and Width analysis

In this section we analyze the variation of the depth and width of the MetaCS-DNN framework on recommendation accuracy. If the proposed approach is truly robust, then increasing the depth beyond a point should not lead to significant increases in recommendation accuracy. We perform experiments to infer whether this hypothesis is indeed true.

Figure 4 shows the variation in Precision@1 of the proposed approach MetaCS-DNN with the depth (height) of the network and the maximum width of any layer. It is evident that the maximum recommendation accuracy (in terms of Precision@1) occurs when both the depth and the maximum width of the network is approximately in the range 2000 to 3000. The accuracy drops when the network becomes both deeper and wider. This is because, since recommendation datasets are intrinsically sparse (users consume only a small fraction of the entire set of consumable items), a larger network is susceptible to overfit during training and perform poorly during testing. When the network is shallower, it does not suffice to accurately capture the latent preference trends of users for items.

V. Conclusion

In this paper, we described the design of a powerful meta-learning based recommendation model that effectively tackles the cold-start problem. Specifically, we designed a recommendation model that is general enough to be reasonable good enough for a wide distribution of users. The trained model can be readily utilized for a specific user by performing a small number of gradient updates during the testing phase. Through detailed simulation studies, we show that very less gradient updates are required to make the trained model suitable for a cold-start (new) user and the recommendation accuracy for such users significantly outperforms state-of-the art baseline models. In addition, we empirically showed that this benefit does not come at the cost of the general recommendation performance for non cold-start users. The next step which is

a part of our future work is to design an interactive platform for implementing our algorithm and testing its efficacy in real-time recommendations to physical users.

REFERENCES

[1] R. Burke, A. Felfernig, and M. H. Göker, "Recommender systems: An overview," *Ai Magazine*, vol. 32, no. 3, pp. 13–18, 2011.

[2] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web*. Springer, 2007, pp. 325–341.

[3] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, no. 1, pp. 76–80, 2003.

[4] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 271–280.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[6] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2008, pp. 1257–1264.

[7] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing cold-start problem in recommendation systems," in *Proceedings of the 2nd international conference on Ubiquitous information management and communication*. ACM, 2008, pp. 208–211.

[8] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *Advances in Neural Information Processing Systems*, 2017, pp. 6904–6914.

[9] H. Bharadhwaj, H. Park, and B. Y. Lim, "Recgan: recurrent generative adversarial networks for recommendation systems," in *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 2018, pp. 372–376.

[10] Y. Zhu, J. Lin, S. He, B. Wang, Z. Guan, H. Liu, and D. Cai, "Addressing the item cold-start problem by attribute-driven active learning," *arXiv preprint arXiv:1805.09023*, 2018.

[11] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," 2017.

[12] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *arXiv preprint arXiv:1802.07876*, 2018.

[13] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4077–4087.

[14] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, p. 19, 2016.

[15] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, p. 13, 2016.

[16] P. Achananuparp and I. Weber, "Extracting food substitutes from food diary via distributional similarity," *arXiv preprint arXiv:1607.08807*, 2016.

[17] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015.

[18] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proceedings of the tenth ACM international conference on web search and data mining*. ACM, 2017, pp. 495–503.

[19] R. Devooght and H. Bersini, "Long and short-term recommendations with recurrent neural networks," in *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. ACM, 2017, pp. 13–21.

[20] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," *arXiv preprint arXiv:1802.05814*, 2018.

[21] 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*. ACM, 2016, pp. 153–162.

[22] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 791–798.

[23] H. Bharadhwaj, H. Park, and B. Y. Lim, "Recgan: recurrent generative adversarial networks for recommendation systems," in *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 2018, pp. 372–376.

[24] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 515–524.

[25] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 191–198.

[26] H. Bharadhwaj, "Layer-wise relevance propagation for explainable recommendations," *arXiv preprint arXiv:1807.06160*, 2018.

[27] N. Houlsby, J. M. Hernández-Lobato, and Z. Ghahramani, "Cold-start active learning with robust ordinal matrix factorization," in *International Conference on Machine Learning*, 2014, pp. 766–774.

[28] A. K. Pandey and D. S. Rajpoot, "Resolving cold start problem in recommendation system using demographic approach," in *Signal Processing and Communication (ICSC), 2016 International Conference on*. IEEE, 2016, pp. 213–218.

[29] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn," Ph.D. dissertation, Technische Universität München, 1987.

[30] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei, "On the optimization of a synaptic learning rule," in *Preprints Conf. Optimality in Artificial and Biological Neural Networks*. Univ. of Texas, 1992, pp. 6–8.

[31] H. Bharadhwaj, Z. Wang, Y. Bengio, and L. Paull, "A data-efficient framework for training and sim-to-real transfer of navigation policies," *arXiv preprint arXiv:1810.04871*, 2018.

[32] O. Klejch, J. Fainberg, and P. Bell, "Learning to adapt: a meta-learning approach for speaker adaptation," *arXiv preprint arXiv:1808.10239*, 2018.

[33] T. Munkhdalai and H. Yu, "Meta networks," in *International Conference on Machine Learning (ICML)*, 2017.

[34] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," in *International Conference on Learning Representations (ICLR)*, 2016.

[35] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell, "Data-dependent initializations of convolutional neural networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[36] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," *arXiv preprint arXiv:1806.02817*, 2018.

[37] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn, "Bayesian model-agnostic meta-learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 7343–7353.

[38] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2016.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[41] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 447–456.

[42] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[43] K. A. Spackman, "Signal detection theory: Valuable tools for evaluating inductive learning," in *Proceedings of the sixth international workshop on Machine learning*. Elsevier, 1989, pp. 160–163.

[44] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[46] C.-W. Hsu, C.-C. Chang, C.-J. Lin, *et al.*, "A practical guide to support vector classification." Taipei, 2003.

[47] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[48] V. Tinto, "Dropout from higher education: A theoretical synthesis of recent research," *Review of educational research*, vol. 45, no. 1, pp. 89–125, 1975.