Privacy-Preserving Multi-Target Multi-Domain Recommender Systems with Assisted AutoEncoders

Enmao Diao
Duke University
Durham, US
enmao.diao@duke.edu

Vahid Tarokh
Duke University
Durham, US
vahid.tarokh@duke.edu

Jie Ding
University of Minnesota-Twin Cities
Minnesota, US
dingj@umn.edu

ABSTRACT

A long-standing challenge in Recommender Systems (RCs) is the data sparsity problem that often arises when users rate very few items. Multi-Target Multi-Domain Recommender Systems (MTMDR) aim to improve the recommendation performance in multiple domains simultaneously. The existing works assume that the data of different domains can be fully shared, and the computation can be performed in a centralized manner. However, in many realistic scenarios, separate recommender systems are operated by different organizations, which do not allow the sharing of private data, models, and recommendation tasks. This work proposes an MTMDR based on Assisted AutoEncoders (AAE) and Multi-Target Assisted Learning (MTAL) to help organizational learners improve their recommendation performance simultaneously without sharing sensitive assets. Moreover, AAE has a broad application scope since it allows explicit or implicit feedback, user- or item-based alignment, and with or without side information. Extensive experiments demonstrate that our method significantly outperforms the case where each domain is locally trained, and it performs competitively with the centralized training where all data are shared. As a result, AAE can effectively integrate organizations from different domains to form a community of shared interest.

KEYWORDS

Recommender Systems, Multi-Organization Learning, Distributed Machine Learning

ACM Reference Format:
Enmao Diao, Vahid Tarokh, and Jie Ding. 2021. Privacy-Preserving Multi-Target Multi-Domain Recommender Systems with Assisted AutoEncoders. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years, Recommender Systems (RSs) have become one of the most popular techniques in web applications that involve big data. This is because they can effectively extract helpful information for relevant users, e.g., in recommending restaurants, videos, and e-commerce products [1, 18]. However, a long-standing challenge in recommender systems is the data sparsity problem because the users usually rate very few items. To address this issue, the research direction of Cross-Domain Recommendation (CDR) has been developed to leverage ratings from a source domain where users may have relatively more information to improve the performance of a target domain [2]. A system that simultaneously improves the performance of both the source and target domains is referred to as Dual-Target CDR [33]. Since then, Multi-Target Multi-Domain Recommendation (MTMDR) has become a popular solution to improve the recommendation performance of multiple domains simultaneously. However, most existing works on this topic require that the data of different domains are fully shared, and the computation must be performed in a centralized manner [3, 28, 31, 34]. Since most recommender systems are built upon users' sensitive data, e.g., user profiles and usage history, and the model and task information are also proprietary to organizational learners [25], collaborations among organizations from different domains are often restricted by ethical, regulatory, and commercial constraints [27, 33]. Therefore, a privacy-preserving MTMDR, as shown in Figure 1, is the keystone for leveraging isolated data held by various domains.

This work proposes an MTMDR based on Assisted AutoEncoders (AAE) and Multi-Target Assisted Learning (MTAL), which help different organizations improve their recommendation performance simultaneously while preserving their local data, models, and task labels. In particular, each organization will calculate a set of 'residuals' and broadcast these to other organizations. These residuals approximate the fastest direction of reducing the training loss in hindsight. Subsequently, other organizations will fit the residuals using their local data, models, and objective functions and broadcast the fitted values back to each other. Each learner will then assign weights to its peers to approximate the fastest direction of learning. The prediction will be aggregated from the fitted values. The above procedure is repeated until all organizations accomplish a sufficient level of learning. Moreover, our approach can handle explicit or implicit feedback [9], user- or item-based alignment [33], and with or without side information [24]. We perform extensive experiments to demonstrate that our method significantly outperforms the case where each domain is locally trained. It performs competitively with centralized training where all data are shared. As a result, our method can effectively integrate organizations from different domains to form a community of shared interest, as shown in Figure 1. Our main contributions are summarized below.

- We present a new paradigm of privacy-preserving Multi-Target Multi-Domain Recommendation (MTMDR), which can effectively improve the recommendation performance of different domains simultaneously, without sharing their local data, models, or task labels.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

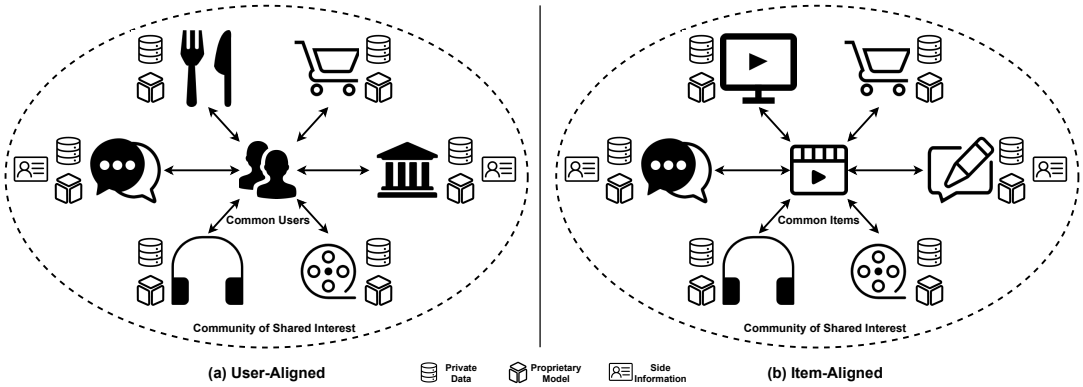


Figure 1: (a) User aligned (b) Item aligned Muti-Target Multi-Domain Recommender Systems. Organizations from different domains form a community of shared interest.

- We propose Assisted AutoEncoders (AAE) and Multi-Target Assisted Learning as a systematic solution to the proposed MTMDR scenario. It exchanges domain-specific information through pseudo-residuals fitted with local data and models. Our method covers broad application scenarios, including explicit or implicit feedback, user- or item-based alignment, and with or without side information.
- We conduct extensive experiments to demonstrate that our method can significantly outperform the case where each domain is locally trained and perform competitively with the centralized training where all data are shared. As a result, AAE can effectively integrate organizations from different domains to form a community of shared interest.

2 RELATED WORK

2.1 Recommender Systems

Recommender Systems (RCs) predict users' preference on items and provide personalized recommendations for users [1, 18]. Recommendation approaches are mainly classified into three categories [1, 10], namely collaborative filtering, content-based recommendation, and hybrid systems. Specifically, collaborative filtering learns from user-item interactions, while the content-based recommendation is primarily based on side information. Hybrid systems leverage both user-item interactions and side information. Our proposed method is a hybrid recommender system that can integrate 1) user-item interactions, 2) either explicit feedback (e.g., user's previous ratings) or implicit feedback (e.g., browsing history), and 3) side information.

Cross-Domain Recommender Systems In most realistic scenarios, users usually provide very few ratings among many items [19]. It results in a highly sparse interaction matrix which hinders the recommendation performance. To address the data sparsity problem, cross-domain recommendation (CDR) [2] has been proposed to utilize relatively richer information from the source domain to improve the recommendation performance in the target domain. Recent CDR methods use the latent factors obtained from a source domain for a target domain [6, 16, 29, 30, 32]. Furthermore, Dual-Target CDR was proposed to improve the performance of both source and target domains [33]. A natural extension of Dual-Target CDR is Multi-Target Multi-Domain Recommendation (MTMDR), which aims to improve the recommendation performance of multiple domains simultaneously [28, 34]. In this research direction, the existing works assume that different data domains can be trained in a centralized manner. However, in many practical scenarios, separate recommender systems are operated by various organizations, prohibiting the sharing of private data and model parameters. To address this challenge, we will propose a method to enhance the performance of multiple domains simultaneously without transmitting local data and models.

Autoencoder-based Recommender Systems AutoRec [20] applies AutoEncoders (AE) to RCs and has found many successful applications. It takes user vectors or item vectors as input and reconstructs them in the output layer. Several recent studies improve the performance of AutoRec by using denoising AE [13, 22], variational AE [14, 15], and Dropout [5, 12, 21]. Side information can also be leveraged with autoencoders to tackle the cold start problem [23]. AutoEncoder-based recommender systems have two variants, namely user-based and item-based. Our method can leverage user-based and item-based AutoEncoders to handle user-based and item-based alignment in MTMDR, respectively.

2.2 Assisted Learning

Assisted Learning (AL) [26] is a collaborative learning framework where organizations being assisted or assisting others do not share private local data and models. The recently proposed Gradient Assisted Learning (GAL) [4] generalizes AL from a sequential protocol to parallel aggregation across multiple organizations. This work develops and generalizes the GAL algorithm for privacy-preserving MTMDR based on a novel design of autoencoder-based recommender systems.

3 PROBLEM FORMULATION

Recommender Systems Let $\mathcal{U} = \{u_1, \ldots, u_m\}$ and $\mathcal{V} = \{v_1, \ldots, v_n\}$ denote respectively the set of users and items, where m is the number of users and n is the number of items. We have a user-item interaction or rating matrix $\mathcal{R} \in \mathbb{R}^{m \times n}$, where $r_{i,j} \in \mathcal{R}$ denotes the rating that user u_i gives to item v_j. A recommender system

$F(\cdot)$ predicts the rating $\hat{r}_{i,j}$ given a pair of user and item (u_i, v_j), i.e. $\hat{r}_{i,j} = F(u_i, v_j)$. The recommender system can also incorporate side information such as user profile $s_{u,i} \in \mathcal{S}_u$ and item attributes $s_{v,j} \in \mathcal{S}_v$ that are associated with the user u_i and item v_j, so that $\hat{r}_{i,j} = F(u_i, v_j, s_{u,i}, s_{v,j})$. We train a recommender system by minimizing an average of loss values in the form of $L(\hat{r}_{i,j}, r_{i,j})$.

Multi-Target Multi-Domain Recommender Systems (MTMDR) Suppose that there are K observed domains including the user sets $\{\mathcal{U}^1, \ldots, \mathcal{U}^K\}$ and the item sets $\{\mathcal{V}^1, \ldots, \mathcal{V}^K\}$. We have a set of rating matrices $\{\mathcal{R}^1, \ldots, \mathcal{R}^K\}$, where domain k has m_k users and n_k items. Domain k can train a separate recommender system $\tilde{F}^k(\cdot)$. However, in order to resolve the data sparsity problem, a multi-domain recommender system $F^k(\cdot)$ aims to improve the performance of the locally trained recommender system by leveraging the common users $\mathcal{U}_c^k$ or the common items $\mathcal{V}_c^k$ that appear in other sets of users or items. The common users or items are those shared between a pair of domains, described by

$$\mathcal{U}_c^k = \{\mathcal{U}^k \cap \mathcal{U}^1, \ldots, \mathcal{U}^k \cap \mathcal{U}^K\} \tag{1}$$

$$\mathcal{V}_c^k = \{\mathcal{V}^k \cap \mathcal{V}^1, \ldots, \mathcal{V}^k \cap \mathcal{V}^K\}. \tag{2}$$

As shown in Figure 1, multi-domain recommender systems can be categorized based on their alignment. Depending on the application scenario, a *user-aligned* multi-domain recommender system leverages the common users, while an *item-aligned* one leverages the common items.

The general goal is to develop a multi-domain recommender system that significantly outperforms each locally trained recommender system, and that performs competitively with the recommender system $F(\cdot)$ jointly trained from all user and item sets. In other words,

$$\mathbb{E}\{L(F^k(u_i^k, v_j^k), r_{i,j}^k)\} \ll \mathbb{E}\{L(\tilde{F}^k(u_i^k, v_j^k), r_{i,j}^k)\}, \tag{3}$$

$$\mathbb{E}\{L(F^k(u_i^k, v_j^k), r_{i,j}^k)\} \approx \mathbb{E}\{L(F(u_i^k, v_j^k), r_{i,j}^k)\}, \tag{4}$$

where the expectation $\mathbb{E}$ is over test data. To achieve the above objective for all data domains whose data are distributed in practice, we develop a multi-target multi-domain recommender system that does not share the rating matrix $\mathcal{R}^k$, user profile $\mathcal{S}_u^k$, item attribute $\mathcal{S}_v^k$, model $F^k(\cdot)$, and objective function $L^k(\cdot)$.

4 METHOD

4.1 Model

AutoEncoders An autoencoder is a network consisting of an encoder $z = E(x) : R^{d_{\text{in}}} \to R^d$ and a decoder $\hat{x} = D(x) : R^d \to R^{d_{\text{out}}}$, where d_{in} and d_{out} are dimensions of the input vector x and output vector $\hat{x}$, respectively. Unlike Collaborative Filtering, the input of an autoencoder-based recommender system is not a pair of user u_i and item v_j. Instead, the input of a user-based autoencoder is a partially observed vector $\mathbf{r}_i = (r_{i,1}, \ldots, r_{i,n}) \in \mathbb{R}^n$ which represents the ratings of user u_i giving to all items $v_1 \ldots v_n$. We represent the partially observed vector $\mathbf{r}_i$ as a sparse vector, where the unknown ratings are zeros. Similarly, the input of an item-based autoencoder is $\mathbf{r}_j = (r_{1,j}, \ldots, r_{m,j}) \in \mathbb{R}^m$. The encoder transforms the observed vector into a dense lower-dimensional code, namely $\mathbf{z}_i = E(\mathbf{r}_i)$ or $\mathbf{z}_j = E(\mathbf{r}_j)$. The decoder produces the output vector $\hat{\mathbf{r}}_i = D(E(\mathbf{r}_i)) \in \mathbb{R}^n$ and $\hat{\mathbf{r}}_j = D(E(\mathbf{r}_j)) \in \mathbb{R}^m$, where n is the number of items, and m is the number of users. The dimension of the input vector is the same as that of the output vector for an autoencoder-based reocommender system. We train an autoencoder-based recommender system by minimizing the average loss between the input and output vectors. When computing the loss function, we mask out the unobserved output ratings.

Assisted AutoEncoders Our model is inspired by a series of autoencoder-based recommender systems [5, 12, 20], but it has an unusual architecture. Suppose that there are K observed domains. Each domain can train its own Assisted AutoEncoder (AAE). The input of a user-based AAE is a partially observed vector $\mathbf{r}_i^k = (r_{i,1}^k, \ldots, r_{i,n_k}^k) \in \mathbb{R}^{n_k}$ that represents the ratings of user u_i^k giving to all items $v_1^k \ldots v_{n_k}^k$ in domain k, where n_k is the number of items in domain k. We represent the partially observed vector $\mathbf{r}_i^k$ as a sparse vector, where the unknown ratings are zeros. Similarly, the input of an item-based AAE is $\mathbf{r}_j^k = (r_{1,j}^k, \ldots, r_{m_k,j}^k) \in \mathbb{R}^{m_k}$, where m_k is the number of users in domain k. The key difference between previous works and ours is that the output vector of a user-based and item-based AAE has the dimension of total number of items or users, i.e. $\hat{\mathbf{r}}_i^k = D(E(\mathbf{r}_i^k)) \in \mathbb{R}^n$ and $\hat{\mathbf{r}}_j^k = D(E(\mathbf{r}_j^k)) \in \mathbb{R}^m$, where n is the total number of items, and m is the total number of users, across all domains.

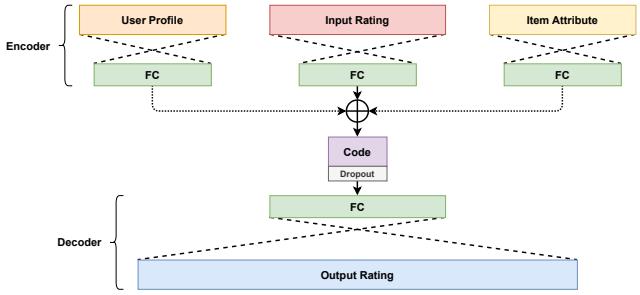


Figure 2: Assisted AutoEncoders (AAE) incorporate side information by summing up the codes encoded with two additional encoders corresponding to the user profile and item attribute. For each domain k, the input dimension is the number of users or items of that domain, while the output dimension is the total number of users or items of all domains. It uses $\tanh(\cdot)$ for nonlinear activation and Dropout for regularization.

Figure 2 demonstrates an example of the AAE network. Both encoder and decoder consist of fully connected (FC) layers. We use $\tanh(\cdot)$ as our nonlinear activation function because it is important for hidden layers to contain negative parts [5, 12]. We adopt Dropout [21] at the encoded space as suggested by [12]. We also consider side information of domain k such as user profile $\mathcal{S}_u^k \in \mathbb{R}^{m_k \times d_{s,u}^k}$ and item attributes $\mathcal{S}_v^k \in \mathbb{R}^{n_k \times d_{s,v}^k}$, where $d_{s,u}^k$ and $d_{s,v}^k$ denote the feature dimension of user profile and item attribute at domain k, respectively. We can transform non-structural side information such as text, image, and video into dense feature representations [24]. AAE can train two additional encoders for the user profile and item attribute. For a user-based AAE, the input of the encoder for user profile is $\mathbf{s}_{u,i}^k \in \mathbb{R}^{d_{s,u}^k}$, and the input of the

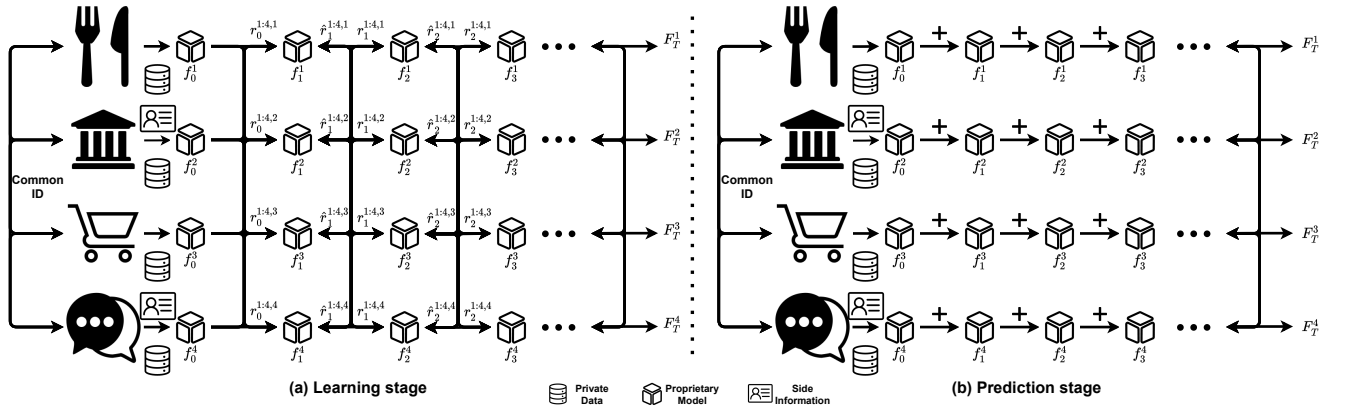


Figure 3: Learning and Prediction stages for Multi-Target Assisted Learning (MTAL). Organizations from different domains construct the set of common users or items. The domain k learns local models f_k^t with received pseudo-residuals $r_t^{1:K,k}$ and predicted outputs $\hat{r}_t^{1:K,k}$ from all the domains 1 to K.

encoder for item attribute is the summation of the observed item attributes, namely $\sum_{j\in\mathcal{V}_i^k} \mathbf{s}_{v,j}^k \in \mathbb{R}^{d_{s,v}^k}$, where $\mathcal{V}_i^k$ represents the set of observed items for user u_i^k. When considering side information, we sum up the codes encoded from ratings and side information.

We cannot train AAE directly because, in our privacy-preserving setting, an isolated domain does not have access to the ratings from other domains. To address this issue, we apply the assisted learning framework and develop a multi-target algorithm for MTMDR.

We propose Multi-Target Assisted Learning (MTAL) as demonstrated in Algorithm 1, so that *each domain can operate on its own local data, model, and objective function.* Our algorithm considers both user- and item-aligned multi-domain recommender systems. The item-aligned case can be regarded as a transposed version of the user-aligned case. Figure 3 gives a flow chart of our algorithm. Next, we describe the learning and prediction procedures in detail.

4.2 Algorithm

At the beginning, each organization, or domain k, coordinates with others to construct the set of common users $\mathcal{U}_c^k$ or items $\mathcal{V}_c^k$, depending on whether the system is user-aligned or item-aligned. During the Learning stage, each domain initializes with an unbiased base model $F_0^k(\mathcal{R}^k) = \mathbb{E}_n(\mathcal{R}^k) \overset{\Delta}{=} B^{-1}\sum \mathbf{r}_i^k \in \mathbb{R}^{n_k}$. For the explicit feedback, the base model is the average ratings, where B is the number of ratings of each item at domain k. For the implicit feedback, the base model is the popularity estimates, where B is the number of users at domain k. At every assistance round t, each domain computes its own 'pseudo residuals' $r_t^{k,k}$ and broadcast its common residuals $r_t^{k,l}$ to another domain l, where

$$r_t^{k,k} = -\left[\frac{\partial L_k\left(F_{t-1}^k(\mathcal{R}^k), \mathcal{R}^k\right)}{\partial F_{t-1}^k(\mathcal{R}^k)}\right], \; r_t^{k,l} = \{r_{t,i}^{k,k}, i \in \mathcal{U}^k \cap \mathcal{U}^l\}, \quad (5)$$

$L^k(\cdot)$ is the overarching loss function used by domain k, and $F_{t-1}^k(\mathcal{R}^k)$ is the output from the previous assistance round. Here, the superscript of $r^{i,j}$ means the domain i transmits the residuals to domain j, or the domain j receives from domain i. In Figure 3, we let $r^{1:K,k}$ denote all the received residuals of domain k from domains 1 to K.

Then, each domain aggregates its own residuals together with the received common residuals from other domains into 'pseudo targets' $\hat{\mathcal{R}}_t^k = \{r_t^{1,k}, \ldots, r_t^{K,k}\} \in \mathbb{R}^{m_k \times n}$. Note that $\hat{\mathcal{R}}_t^k$ is a sparse matrix of size $m_k \times n$. The dimension of items increases from n_k to n, because the received common residuals from other domains introduce unobserved targets of items. Next, each domain will fit a local AAE f_t^k with the pseudo targets $\hat{\mathcal{R}}_t^k$ and the local loss function $\ell^k(\cdot)$. Each domain will then broadcast the common predicted outputs $\hat{r}_t^{k,1:K}$ to domains 1 to K.

$$f_t^k = \underset{f^k}{\operatorname{argmin}}\, \ell^k\left(f^k(\mathcal{R}^k), \hat{\mathcal{R}}^k\right), \quad (6)$$

$$\hat{r}_t^{k,k} = f_t^k(\mathcal{R}^k), \; \hat{r}_t^{k,l} = \{\hat{r}_{t,i}^{k,k}, i \in \mathcal{U}^k \cap \mathcal{U}^l\}. \quad (7)$$

Subsequently, each domain can train suitable gradient assistance weights w_k to aggregate received outputs, and gradient assisted learning rate η_k to minimize the overarching loss.

$$w_t^k = \underset{w\in P_M}{\operatorname{argmin}}\, \ell_k\left(\sum_{j=1}^K w_j \hat{r}_t^{j,k}, \hat{\mathcal{R}}^k\right) \quad (8)$$

$$\eta_t^k = \underset{\eta^k\in\mathbb{R}}{\operatorname{argmin}}\, \mathcal{L}_k\left(F_{t-1}^k(\mathcal{R}^k) + \eta^k \sum_{j=1}^K w_t^j \hat{r}_t^{j,k}, \mathcal{R}^k\right) \quad (9)$$

where $P_K = \{w \in \mathbb{R}^K : \sum_{k=1}^K w_k = 1, w_k \geq 0\}$ denotes the probability simplex. We note that optimizing w_k and η_k is optional. We have conducted ablation studies of w_k and η_k in Section 5. Finally, the output at round t is

$$F_t^k(\mathcal{R}^k) = F_{t-1}^k(\mathcal{R}^k) + \eta^k \sum_{j=1}^K w_t^j \hat{r}_t^{j,k}. \quad (10)$$

The above procedure can be iterated for T assistance rounds until we obtain a satisfactory performance (e.g., on validation data).

In the Prediction stage, each domain will predict outputs from their local models f_t^k for all assistance rounds from 1 to T. The

Algorithm 1: MTAL: Multi-Target Assisted Learning for Recommender Systems.

Input: K decentralized data domains, domain k holding rating matrix $\mathcal{R} \in \mathbb{R}^{m_k \times n_k}$, Assisted AutoEncoders (AAE) $f^k(\cdot)$, gradient assistance weights w_k, gradient assisted learning rate η_k, overarching loss function L_k, local loss function ℓ_k, and the number of assistance rounds T.

Learning Stage:

Alignment:

Construct the set of common users $\mathcal{U}_c^k$ or items $\mathcal{V}_c^k$

Intialization:

Let $t = 0$, and initialize $F_0^k(x) = \mathbb{E}_n(\mathcal{R}^k)$ (where $\mathbb{E}_n$ denotes the sample average)

for assistance round t from 1 to T **do**

Compute pseudo-residuals $r_t^{k,k}$ and broadcast common pseudo-residuals $r_t^{k,l}$ to other organizations

$r_t^{k,k} = -\left[\frac{\partial L_k(F_{t-1}^k(\mathcal{R}^k),\mathcal{R}^k)}{\partial F_{t-1}^k(\mathcal{R}^k)}\right]$, $r_t^{k,l} = \{r_{t,i}^{k,k}, i \in \mathcal{U}^k \cap \mathcal{U}^l\}$

for organization m from 1 to M *in parallel* **do**

Aggregates pseudo-residuals and construct pseudo-targets $\hat{\mathcal{R}}_t^k = \{r_t^{1,k}, \ldots, r_t^{K,k}\} \in \mathbb{R}^{m_k \times n}$

Fit local AAE and broadcast the common predicted outputs $\hat{r}_t^{k,l}$ to other domains

$f_t^k = \text{argmin}_{f^k} \ell^k\left(f^k(\mathcal{R}^k), \hat{\mathcal{R}}^k\right)$, $\hat{r}_t^{k,k} = f_t^k(\mathcal{R}^k)$, $\hat{r}_t^{k,l} = \{\hat{r}_{t,i}^{k,k}, i \in \mathcal{U}^k \cap \mathcal{U}^l\}$

end

Optimize the gradient assistance weights $w_t^k = \text{argmin}_{w \in P_M} \ell_k\left(\sum_{j=1}^K w_j \hat{r}_t^{j,k}, \hat{\mathcal{R}}^k\right)$

Line search for the gradient assisted learning rate $\eta_t^k = \text{argmin}_{\eta^k \in \mathbb{R}} \mathcal{L}_k\left(F_{t-1}^k(\mathcal{R}^k) + \eta^k \sum_{j=1}^K w_t^j \hat{r}_t^{j,k}, \mathcal{R}^k\right)$

$F_t^k(\mathcal{R}^k) = F_{t-1}^k(\mathcal{R}^k) + \eta^k \sum_{j=1}^K w_t^j \hat{r}_t^{j,k}$

end

Prediction Stage:

Gather predictions $\hat{r}_t^{j,k} = f_t^k(\mathcal{R}^k)$, $t = 1, \ldots, T$ from each domain j, $j = 1, \ldots, K$

Predict with $F^T(\mathcal{R}^k) \triangleq F^0(\mathcal{R}^k) + \sum_{t=1}^T \eta^k \sum_{j=1}^K w_t^j \hat{r}_t^{j,k}$

Return $F^T(\mathcal{R}^k)$

predicted results will be broadcast to other domains, which will aggregate them with gradient assistance weights $w_{1:T}$ and gradient assisted learning rate $\eta_{1:T}$, to eventually generate a final prediction $F^T(x)$ that is implicitly operated on $\mathcal{R}$. Compared with the Learning stage, the Prediction stage does not need synchronization of each assistance round because we can operate all local AAEs across T rounds before broadcasting the outputs.

Our framework can help participants to form a shared community of interest. In particular, every data domain can provide its own task and seek assistance from others. The end-to-end assistance provided for another organization does not require the sharing of anyone's proprietary data, models, and objective functions. In practice, the participating organizations may receive financial rewards from the assisting ones. Consequently, all participants can become mutually beneficial to each other.

4.3 Discussion

Why AutoEncoders? We choose autoencoders as our local backbone model because AE is naturally compatible with our proposed MTAL algorithm. In particular, autoencoders can treat the rating matrix as tabular data. The rows and columns of the rating matrix are considered as data sample and feature space, respectively. A multi-domain system can be viewed as each domain holding a subset of the feature space. We can adopt user-based autoencoders for user-aligned multi-domain recommender systems, and item-based for item-aligned systems. On the contrary, collaborative filtering takes user-item pairs as the input, which do not contain a feature space. Thus, it is not suitable to integrate collaborative filtering with our MTAL algorithm. It is worth mentioning that the proposed MTAL algorithm is not restricted to autoencoders. A possible future work is to discover better local models which can treat the rating matrix as tabular data for MTAL.

Loss function Our algorithm involves two kinds of loss functions, namely a local loss function $\ell_k(\cdot)$ for fitting the pseudo-targets $\hat{\mathcal{R}}^k$, and an overarching loss function $L_k(\cdot)$ for fitting the ratings $\mathcal{R}^k$ in hindsight. Since fitting the pseudo-targets is a regression problem, it is standard to let $\ell_k(\cdot)$ be ℓ_1- or ℓ_2-norm. Depending on the ratings are explicit or implicit feedback, we choose different overarching loss functions. We use regression loss functions such as the ℓ_2-norm for explicit feedback and classification loss functions such as binary cross-entropy for implicit feedback.

Technical novelties Our proposed AAE absorbs many merits of previous autoencoder-based recommender systems, such as tanh$(\cdot)$ activation function [5, 12], encoders for side information [23], and Dropout [5, 12]. AAE extends the scope of standard autoencoders for multi-domain recommender systems by increasing the output dimension. Moreover, the proposed MTAL algorithm

develops AL in two ways. First, we generalize AL from a single-target to a multi-target learning framework. In particular, GAL [4] assumes multiple participants assist one organization. Our MTAL method fits multiple targets of all domains with a single local model. Therefore, MTAL can simultaneously improve the performance of all participants. Meanwhile, we also perform ablation studies of gradient assistance weights and gradient assisted learning rate. Our results show that 1) a constant gradient assisted learning rate suits the explicit feedback, while an optimized gradient assisted learning rate suits the implicit feedback, and 2) optimized gradient assistance weights tend to improve the result if the domain partition is Non-IID.

Privacy Our proposed algorithm allows different domains to improve their recommendation performance without sharing their local data, models, and objective functions. We consider this requirement as a bottom line for privacy-preserving MTMDR. Nevertheless, we are aware that it is possible to apply further privacy enhancement techniques such as Differential Privacy (DP) by adding noises to the transmitted residuals [4].

5 EXPERIMENTS

5.1 Experimental Setup

Data We conduct experiments with MovieLens100K (ML100K), MovieLens1M (ML1M), and MovieLens10M (ML10M) datasets [7] under various circumstances. We have following control settings.
1) Explicit vs. implicit feedback. The explicit feedback is the default rating $(1-5)$, while the implicit feedback is the binarization of default ratings (positive if greater than 3.5).
2) User- vs. item-alignment. We introduce two types of data partition for multi-domain recommender systems. The 'Uniform' data partition splits items (respectively users) for user-aligned (respectively item-aligned) multi-domain recommender systems into $K = 8$ domains. Each domain has roughly the same number of items or users. The 'Genre' data partition splits items according to $K = 18$ of movies for user-aligned multi-domain recommender systems.
3) With vs. without side information. The user profiles like gender and occupation are available for ML100K and ML1M datasets. The item attributes are one-hot indicators of genres.
4) Ablation studies. We conduct ablation studies for gradient assisted learning rate, gradient assistance weights, and partial alignment. In addition, the summary statistics of each dataset are elaborated in Table 7 in the Appendix. For all datasets, we train on 90% of the available data and test on the remaining. We conduct four random experiments for all datasets with different seeds. The standard errors of all our results are smaller than $1E-3$.

Model We compare the results of AAE with the unbiased Base model described in Section 4.2, and collaborative filtering methods such as Matrix Factorization (MF) [17], Multi-Layer Perceptron (MLP) [8], and Neural Collaborative Filtering (NCF) [8]. We also use a subscript s to represent models incorporating side information, namely MF_s and AAE_s.

Our proposed method is a novel integration of Assisted Autoencoders (AAE) and Multi-Target Assisted Learning (MTAL) algorithm. Although we have used the same model architecture for every domain in the experiments, our algorithm does not require every domain to use the same local models. An organization can choose the size of autoencoders based on its local computational capabilities. We use the ℓ_2-norm as the local loss function ℓ_m throughout experiments. We use the Adam optimizer with a learning rate of 10^{-3} [11]. We set the number of assistance rounds $T = 10$ throughout our experiments. The number of local training epochs at each round is 20. To optimize the gradient assistance weights and gradient assisted learning rate, we use the Limited-Memory BFGS (L-BFGS) optimizer with a learning rate of 1. Details of model architecture and learning hyper-parameters are included in Tables 5 and 6 in the Appendix.

Baseline We compare the proposed method with two baselines, 'Joint' and 'Alone'. The 'Joint' denotes the centralized case where all the data are held by one organization. The 'Alone' denotes the case where every domain trains and tests on its local data. As mentioned in Equations 3 and 4, the goal of our method is to significantly outperform the 'Alone' case and perform competitively with the 'Joint' case. We note that in 'Alone' and 'Joint' cases when AAE is not equipped with MTAL, the input and output dimensions of AAE are the same. In that case, the AAE reduces to a conventional autoencoder-based recommender system.

5.2 Experimental Results

We tabulate experimental results in Tables 1, 2, and 4. We illustrate evaluations across all assistance rounds in Figures 4, 5, and 6. We also provide ablation studies of gradient assisted learning rate, gradient assistance weights, and partial alignment in Tables 3, 8, and 9, respectively. We provide detailed discussions below.

Explicit vs. Implicit feedback As shown in Tables 1, 2, and 4, AAE is able to effectively handle explicit and implicit feedback by using different objective functions. We consider explicit feedback as a regression task. For explicit feedback, the objective function is ℓ_2-norm, and the metric is Root Mean Squared Error (RMSE). For implicit feedback, it was shown that we could treat the problem as a binary classification task [17] and use a ranking criterion Mean Average Precision (MAP) to evaluate the performance. Therefore, we use binary cross-entropy as our objective function for implicit feedback.

As shown in Tables 1, 2, and 4, AAE performs competitively with the baseline recommender systems in 'Joint' scenario with explicit feedback. However, compared with baseline models, AAE does not perform well in the 'Alone' scenario with explicit feedback. Moreover, as shown in the results of ML10M, AAE sometimes performs worse in 'Joint' than it does in 'Alone'. It shows that the performance of autoencoder-based recommender systems may be sensitive to the input dimension, namely the number of local items n_k or the number of local users m_k in a user- or item-based recommender. Specifically, when an organization has explicit feedback for a small number of items or users, it should use collaborative filter methods instead of autoencoder-based methods. Moreover, when the organization has explicit feedback for a very large number of items or users, the width and depth of hidden layers of AAE may also increase to alleviate the curse of dimensionality. On the other hand, AAE outperforms baseline models in both 'Joint' and 'Alone' cases with implicit feedback.

Our proposed method AAE equipped with MTAL significantly outperforms all 'Alone' cases with different backbone models for

both explicit and implicit feedback. Our proposed method also performs competitively with the 'Joint' cases with explicit feedback. Furthermore, our method can moderately outperform the 'Joint' case with implicit feedback and 'Genre' data partition. It was demonstrated in [4] that local models could specialize on the Non-IID partitioned data and outperforms the 'Joint' case.

User- vs. Item-alignment It needs to point out that that user- and item-alignment have no impact on baseline models because the input of the base model and collaboratively filter models can be a pair of user and item. However, we use different training procedures for user- and item-aligned recommender systems to compare with autoencoder-based systems. Recall that user- and item-based autoencoder-based recommender systems handle user- and item-alignment in a tabular fashion, respectively. Therefore, each data sample of a user- or item-aligned task corresponds to the ratings of one user giving to associated items or one item received from associated users.

The results of baseline models for explicit feedback are similar for both user- and item-alignment. As for implicit feedback, the results of baseline models are different because of the ranking criterion MAP. We point out that the item-based AAE outperforms baseline models in the 'Joint' case with both types of feedback, and in the 'Alone' case with implicit feedback. AAE equipped with MATL significantly outperforms all 'Alone' cases for both types of alignment, and also performs close to the 'Joint case' for user-alignment. Furthermore, our method also moderately outperforms the 'Joint' case for item-alignment.

With vs. Without side information Recall that we use a subscript s to denote models incorporating side information. The side information contains user profiles such as occupation and gender, and item attributes such as genre. The results show that all models can benefit from the side information. However, autoencoder-based recommender systems do not leverage the side information as much as collaborative filter methods. Some potential future works include incorporating raw side information such as text, image, and video, better modeling techniques of side information, and using side information from domains without ratings.

Gradient assisted learning rate As shown in Tables 3, 8, and 9, we perform an ablation study of the gradient assisted learning rate η_k. We have three control groups of η_k, including '$\eta_k = 0.1$', '$\eta_k = 0.1$', and 'Optimize η_k'. Here, '$\eta_k = 0.1$' and '$\eta_k = 0.3$' represent a constant gradient assisted learning rate for all domains. In all three cases, the optimization of gradient assistance weights is disabled. In particular, the weighted average becomes a direct average of outputs. As demonstrated in Figures 4, 5, and 6, the optimization η_k for explicit feedback may result in gradient explosion due to a very large η_k. However, compared with $\eta_k = 0.1$, a moderately large $\eta_k = 0.3$ can improve the convergence rate and result. On the other hand, the optimization of η_k for implicit feedback greatly improves the performance, because the binary cross-entropy function may require a large η_k at the early assistance rounds. The results show that η_k has a large impact on the success of our method. A more reliable solution to choose η_k is to use a validation set.

Gradient assistance weights We set $\eta_k = 0.1$ for the ablation study of gradient assistance weights w_k. As shown in (b,d) of Figures 4, 5, and 6, the optimization of w_k labeled as 'AAE (Optimize w_k)' moderately improves the performance for 'Genre' data partition. Although the impact of w_k is not obvious in other scenarios, previous works show that using w_k is robust against noisy residuals when we apply privacy enhancement techniques. A potential future work is to develop better techniques to aggregate outputs from each domain.

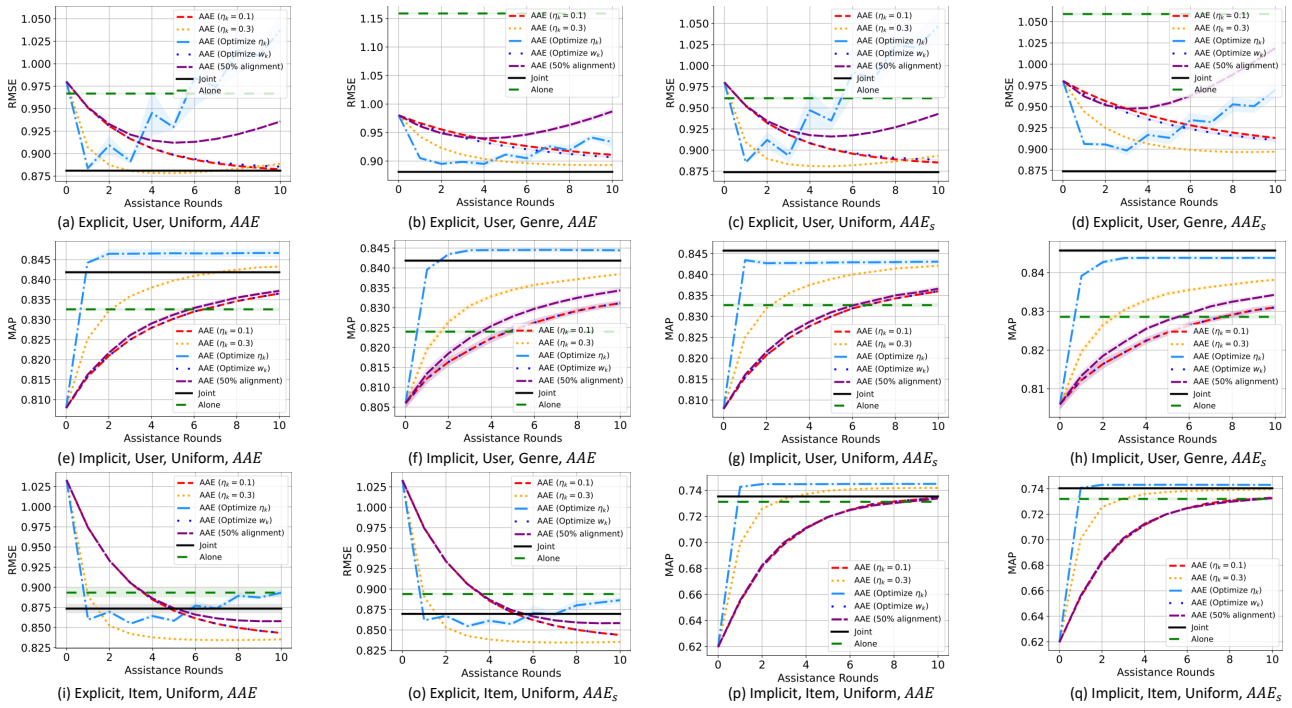
Partial Alignment To compare with baseline models, we assume all users or items are common for user-aligned or item-aligned recommender systems. As mentioned in Equations 1 and 2, the common users or items are shared between a pair of domains. Specifically, two domains can share a subset of their users or items. Therefore, we conduct an ablation study of partial alignment. We assume 50% of users or items are shared among all domains, labeled as '50% Alignment', while the other half is unique for each domain. We set $\eta_k = 0.1$ and disable the optimization of w_k. As demonstrated in Figures 4, 5, and 6, 50% alignment performs worse than full alignment with explicit feedback and close to the full alignment with implicit feedback. It indicates that our proposed solution is robust enough to improve the performance of partially aligned domains.

Table 1: Results of the ML1M dataset for explicit and implicit feedback measured with RMSE and MAP, user- and item-alignment, with and without side information, and Uniform ($K = 8$) and Genre ($K = 18$) data partition.

Feedback		Explicit			Implicit		
Alignment		User		Item	User		Item
Partition		Uniform	Genre	Uniform	Uniform	Genre	Uniform
Joint	Base	1.020		1.040	0.744		0.623
	MF	0.939		0.944	0.768		0.681
	MF_s	0.912		0.911	0.784		0.696
	MLP	0.937		0.921	0.775		0.694
	MLP_s	0.922		0.919	0.771		0.698
	NCF	0.937		0.913	0.777		0.693
	NCF_s	0.912		0.913	0.774		0.693
	AAE	0.927		0.899	0.788		0.702
	AAE_s	0.926		0.898	0.792		0.705
Alone	Base	1.020	1.011	1.040	0.744	0.748	0.624
	MF	1.007	1.184	1.037	0.750	0.755	0.651
	MF_s	0.986	0.971	0.975	0.756	0.770	0.668
	MLP	0.980	0.970	0.980	0.744	0.768	0.669
	MLP_s	0.974	0.960	0.978	0.747	0.768	0.670
	NCF	0.981	0.974	0.980	0.747	0.769	0.670
	NCF_s	0.972	0.960	0.977	0.751	0.768	0.669
	AAE	1.076	1.353	1.094	0.767	0.777	0.667
	AAE_s	1.041	1.174	1.057	0.767	0.780	0.679
MTAL	AAE	**0.938**	**0.945**	**0.883**	0.789	**0.802**	0.706
	AAE_s	0.940	0.950	0.884	**0.789**	0.801	**0.706**

6 CONCLUSION

In this work, we present a new framework of privacy-preserving Multi-Target Multi-Modal Recommender system (MTMDR) based on Assisted AutoEncoders (AAE) and Multi-Target Assisted Learning (MTAL). Our proposed solution allows different organizations to improve their recommendation performance simultaneously while preserving their local data, models, and task labels. Our method covers broad application scenarios, including user- or item-based



(a) Explicit, User, Uniform, AAE (b) Explicit, User, Genre, AAE (c) Explicit, User, Uniform, AAE_s (d) Explicit, User, Genre, AAE_s

(e) Implicit, User, Uniform, AAE (f) Implicit, User, Genre, AAE (g) Implicit, User, Uniform, AAE_s (h) Implicit, User, Genre, AAE_s

(i) Explicit, Item, Uniform, AAE (o) Explicit, Item, Uniform, AAE_s (p) Implicit, Item, Uniform, AAE (q) Implicit, Item, Uniform, AAE_s

Figure 4: Results of the ML1M dataset across all assistance rounds. MTAL significantly outperforms 'Alone' and performs close to 'Joint.' (a-d) Explicit feedback and User-Alignment. (e-h) Implicit feedback and User-Alignment. (i-q) Item Alignment.

Table 2: Results of the ML10M dataset for explicit and implicit feedback measured with RMSE and MAP, user- and item alignment, with and without side information, and Uniform ($K = 8$) and Genre ($K = 18$) data partition.

Feedback		Explicit			Implicit		
Alignment		User		Item	User		Item
Partition		Uniform	Genre	Uniform	Uniform	Genre	Uniform
Joint	Base	0.943		0.977	0.771		0.613
	MF	0.883		0.881	0.793		0.680
	MF_s	0.868		0.875	0.805		0.694
	MLP	0.846		0.843	0.806		0.792
	MLP_s	0.857		0.861	0.806		0.793
	NCF	0.839		0.842	0.806		0.793
	NCF_s	0.849		0.861	0.806		0.793
	AAE	0.949		0.941	0.814		0.786
	AAE_s	0.846		0.939	0.821		0.789
Alone	Base	0.943	0.943	0.977	0.771	0.770	0.613
	MF	0.916	1.025	0.877	0.776	0.774	0.682
	MF_s	0.885	0.883	0.880	0.790	0.789	0.731
	MLP	0.886	0.886	0.855	0.787	0.786	0.786
	MLP_s	0.883	0.882	0.848	0.786	0.786	0.781
	NCF	0.886	0.884	0.855	0.786	0.787	0.785
	NCF_s	0.883	0.892	0.844	0.785	0.786	0.781
	AAE	0.933	1.102	0.834	0.806	0.797	0.797
	AAE_s	0.928	1.074	0.831	0.806	0.800	0.804
MTAL	AAE	**0.847**	**0.863**	0.787	0.816	0.816	**0.806**
	AAE_s	0.850	0.864	**0.787**	**0.816**	**0.816**	0.806

alignment, explicit or implicit feedback, and with or without side information. Extensive experiments demonstrate that our method

Table 3: Ablation study of the ML1M dataset for gradient assistance weights, gradient assisted learning, and partial alignment.

Feedback	Explicit			Implicit		
Alignment	User		Item	User		Item
Partition	Uniform	Genre	Uniform	Uniform	Genre	Uniform
AAE ($\eta_k = 0.1$)	0.883	0.911	0.843	0.837	0.831	0.734
AAE_s ($\eta_k = 0.1$)	0.885	0.913	0.844	0.836	0.831	0.733
AAE ($\eta_k = 0.3$)	**0.878**	0.893	**0.834**	0.843	0.838	0.742
AAE_s ($\eta_k = 0.3$)	0.881	0.897	0.835	**0.842**	0.838	**0.740**
AAE (Optimize η_k)	0.884	**0.893**	0.855	**0.847**	**0.845**	**0.745**
AAE_s (Optimize η_k)	0.884	0.897	0.855	0.844	0.844	0.743
AAE (Optimize w_k)	0.886	**0.907**	0.843	0.837	0.831	0.734
AAE_s (Optimize w_k)	0.889	0.911	0.844	0.836	0.831	0.733
AAE (50% alignment)	0.912	0.940	0.858	0.837	0.834	0.734
AAE_s (50% alignment)	0.916	0.947	0.858	0.837	0.834	0.733

significantly outperforms the case where each domain is locally trained, and it performs competitively with the centralized training where all data are shared. Consequently, our approach can effectively integrate organizations from different domains to form a community of shared interest.

The **Appendix** includes further experimental details and results.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17, 6 (2005), 734–749.

[2] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. 2007. Cross-domain mediation in collaborative filtering. In *International Conference on User Modeling*. Springer, 355–359.

[3] Mohamed Reda Bouadjenek, Esther Pacitti, Maximilien Servajean, Florent Masseglia, and Amr El Abbadi. 2018. A distributed collaborative filtering algorithm using multiple data sources. *arXiv preprint arXiv:1807.05853* (2018).

[4] Enmao Diao, Jie Ding, and Vahid Tarokh. 2021. Gradient Assisted Learning. *arXiv preprint arXiv:2106.01425* (2021).

[5] Diana Ferreira, Sofia Silva, António Abelha, and José Machado. 2020. Recommendation system using autoencoders. *Applied Sciences* 10, 16 (2020), 5510.

[6] Chen Gao, Xiangning Chen, Fuli Feng, Kai Zhao, Xiangnan He, Yong Li, and Depeng Jin. 2019. Cross-domain recommendation without sharing user-relevant data. In *The world wide web conference*. 491–502.

[7] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[9] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.

[10] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. 2010. *Recommender systems: an introduction*. Cambridge University Press.

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[12] Oleksii Kuchaiev and Boris Ginsburg. 2017. Training deep autoencoders for collaborative filtering. *arXiv preprint arXiv:1708.01715* (2017).

[13] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 811–820.

[14] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 305–314.

[15] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.

[16] Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. 2017. Cross-Domain Recommendation: An Embedding and Mapping Approach.. In *IJCAI*, Vol. 17. 2464–2470.

[17] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Fourteenth ACM Conference on Recommender Systems*. 240–248.

[18] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.

[19] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender systems: introduction and challenges. In *Recommender systems handbook*. Springer, 1–34.

[20] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*. 111–112.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[22] Florian Strub and Jeremie Mary. 2015. Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *NIPS workshop on machine learning for eCommerce*.

[23] Florian Strub, Jérémie Mary, and Romaric Gaudel. 2016. Hybrid collaborative filtering with autoencoders. *arXiv preprint arXiv:1603.00806* (2016).

[24] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications* 37 (2019), 100879.

[25] Xinran Wang, Yu Xiang, Jun Gao, and Jie Ding. 2021. Information laundering for model privacy. *International Conference on Learning Representations* (2021).

[26] Xun Xian, Xinran Wang, Jie Ding, and Reza Ghanadan. 2020. Assisted Learning: A Framework for Multi-Organization Learning. *Advances in Neural Information Processing Systems* 33 (2020).

[27] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[28] Yu Zhang, Bin Cao, and Dit-Yan Yeung. 2012. Multi-domain collaborative filtering. *arXiv preprint arXiv:1203.3535* (2012).

[29] Zihan Zhang, Xiaoming Jin, Lianghao Li, Guiguang Ding, and Qiang Yang. 2016. Multi-domain active learning for recommendation. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[30] Lili Zhao, Sinno Jialin Pan, Evan Wei Xiang, Erheng Zhong, Zhongqi Lu, and Qiang Yang. 2013. Active transfer learning for cross-system recommendation. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

[31] Feng Zhu, Chaochao Chen, Yan Wang, Guanfeng Liu, and Xiaolin Zheng. 2019. Dtcdr: A framework for dual-target cross-domain recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1533–1542.

[32] Feng Zhu, Yan Wang, Chaochao Chen, Guanfeng Liu, Mehmet Orgun, and Jia Wu. 2020. A deep framework for cross-domain and cross-system recommendations. *arXiv preprint arXiv:2009.06215* (2020).

[33] Feng Zhu, Yan Wang, Chaochao Chen, Jun Zhou, Longfei Li, and Guanfeng Liu. 2021. Cross-domain recommendation: challenges, progress, and prospects. *arXiv preprint arXiv:2103.01696* (2021).

[34] Feng Zhu, Yan Wang, Jun Zhou, Chaochao Chen, Longfei Li, and Guanfeng Liu. 2021. A Unified Framework for Cross-Domain and Cross-System Recommendations. *IEEE Transactions on Knowledge and Data Engineering* (2021).

Appendix

The Appendix includes further experimental details and results.

Table 6: Hyperparameters used in our experiments for training local models, gradient assistance weights w_m, and gradient assisted learning rates η_m.

Dataset		ML100K	ML1M	ML10M
Local	Batch size	100	500	5000
	Epoch		20	
	Optimizer		Adam	
	Learning rate		1.0E-03	
	Weight decay		5.0E-04	
Optimize w_k, η_k	Epoch		10	
	Batch size		Full	
	Optimizer		L-BFGS	
	Learning rate		1	
Assistance rounds			10	

Table 7: Summary of statistics of ML100K, ML1M, and ML10M datasets. Each dataset contains m users and n items. The dimension of the feature of user profile and item attribute are $d^k_{s,u}$ and $d^k_{s,v}$. Here, n_k (Genre data partition) represents the number of items of each domain partitioned by genres.

	m	n	$d^k_{s,u}$	$d^k_{s,v}$	n_k (Genre)
ML100K	943	1682	30	18	[105, 57, 17, 59, 336, 51, 48, 539, 10, 8, 63, 22, 26, 135, 42, 113, 33, 18]
ML1M	6040	3706	30	18	[232, 117, 43, 105, 799, 109, 105, 1065, 30, 21, 228, 59, 46, 235, 130, 261, 68, 53]
ML10M	69878	10677	N/A	18	[583, 399, 121, 1, 2193, 388, 428, 3291, 183, 62, 555, 173, 184, 693, 316, 702, 244, 161]

Table 8: Ablation study of ML100K dataset for gradient assistance weights, gradient assisted learning, and partial alignment.

Feedback	Explicit			Implicit		
Alignment	User		Item	User		Item
Partition	Uniform	Genre	Uniform	Uniform	Genre	Uniform
AAE ($\eta_k = 0.1$)	0.939	0.954	0.885	0.783	0.791	0.703
AAE$_s$ ($\eta_k = 0.1$)	0.941	0.956	0.886	0.781	0.791	0.703
AAE ($\eta_k = 0.3$)	**0.938**	0.949	**0.883**	0.787	0.801	0.706
AAE$_s$ ($\eta_k = 0.3$)	0.940	0.951	0.884	**0.789**	0.800	**0.706**
AAE (Optimize η_k)	0.951	0.955	0.894	0.783	**0.802**	0.702
AAE$_s$ (Optimize η_k)	0.955	0.960	0.895	0.786	0.801	0.701
AAE (Optimize w_k)	0.942	**0.945**	0.902	0.775	0.791	0.704
AAE$_s$ (Optimize w_k)	0.946	0.950	0.905	0.782	0.791	0.703
AAE (50% alignment)	0.969	0.981	0.945	0.789	0.790	0.697
AAE$_s$ (50% alignment)	0.973	0.987	0.949	0.775	0.790	0.698

Table 9: Ablation study of ML10M dataset for gradient assistance weights, gradient assisted learning, and partial alignment.

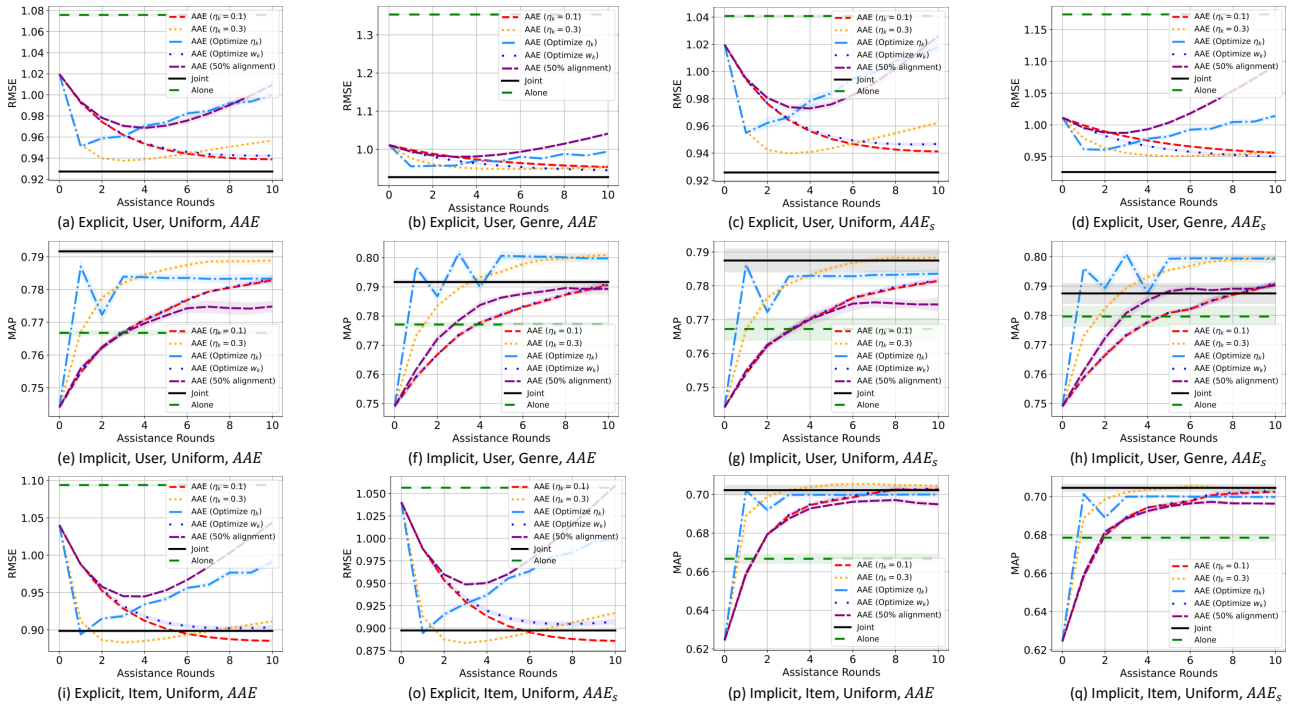
Feedback	Explicit			Implicit		
Alignment	User		Item	User		Item
Partition	Uniform	Genre	Uniform	Uniform	Genre	Uniform
AAE ($\eta_k = 0.1$)	0.857	0.875	0.811	0.809	0.804	0.786
AAE$_s$ ($\eta_k = 0.1$)	0.858	0.876	0.811	0.809	0.804	0.792
AAE ($\eta_k = 0.3$)	0.853	0.865	0.787	0.814	0.810	0.801
AAE$_s$ ($\eta_k = 0.3$)	0.855	0.867	**0.787**	0.814	0.810	0.803
AAE (Optimize η_k)	**0.847**	**0.863**	0.791	0.816	0.816	**0.806**
AAE$_s$ (Optimize η_k)	0.850	0.864	0.791	**0.816**	**0.816**	0.806
AAE (Optimize w_k)	0.865	0.879	0.811	0.809	0.804	0.786
AAE$_s$ (Optimize w_k)	0.868	0.883	0.811	0.809	0.804	0.792
AAE (50% alignment)	0.892	0.920	0.813	0.809	0.806	0.785
AAE$_s$ (50% alignment)	0.898	0.923	0.813	0.808	0.806	0.792

Table 4: Results of ML100K dataset for explicit and implicit feedback measured with RMSE and MAP, user- and item alignment, with and. without side information, and Uniform ($K = 8$) and Genre ($K = 18$) data partition.

Feedback		Explicit			Implicit		
Alignment		User		Item	User		Item
Partition		Uniform	Genre	Uniform	Uniform	Genre	Uniform
Joint	Base	0.980		1.033	0.808		0.620
	MF	0.915		0.914	0.831		0.653
	MF$_s$	0.910		0.909	0.832		0.720
	MLP	0.899		0.908	0.835		0.730
	MLP$_s$	0.884		0.885	0.832		0.725
	NCF	0.893		0.909	0.832		0.727
	NCF$_s$	0.883		0.881	0.832		0.725
	AAE	0.881		0.873	0.842		0.735
	AAE$_s$	0.874		0.870	0.846		0.740
Alone	Base	0.980	0.980	1.033	0.807	0.805	0.619
	MF	0.949	1.060	0.935	0.809	0.809	0.657
	MF$_s$	0.936	0.934	0.908	0.819	0.821	0.711
	MLP	0.932	0.928	0.918	0.811	0.811	0.704
	MLP$_s$	0.930	0.924	0.909	0.812	0.812	0.704
	NCF	0.931	0.929	0.921	0.812	0.811	0.703
	NCF$_s$	0.929	0.923	0.913	0.812	0.817	0.702
	AAE	0.967	1.159	0.893	0.833	0.824	0.731
	AAE$_s$	0.961	1.059	0.894	0.833	0.829	0.732
MTAL	AAE	**0.878**	**0.893**	**0.834**	**0.847**	**0.845**	**0.745**
	AAE$_s$	0.881	0.897	0.835	0.844	0.844	0.743

Table 5: The model architecture of models used in our experiments. The embedding size of MF and NCF is 128. The fully connected networks used in MLP and NCF have four layers of size $[128, 64, 32, 16]$. Our proposed AAE has a two-layer encoder of size $[n_k$ or $m_k, 256, 128]$ and a two-layer decoder of size $[128, 256, n$ or $m]$.

Model	Architecture
MF	128
MLP	$[128, 64, 32, 16]$
NCF	$[128, 64, 32, 16]$
AAE	$[n_k$ or $m_k, 256, 128]$, $[128, 256, n$ or $m]$

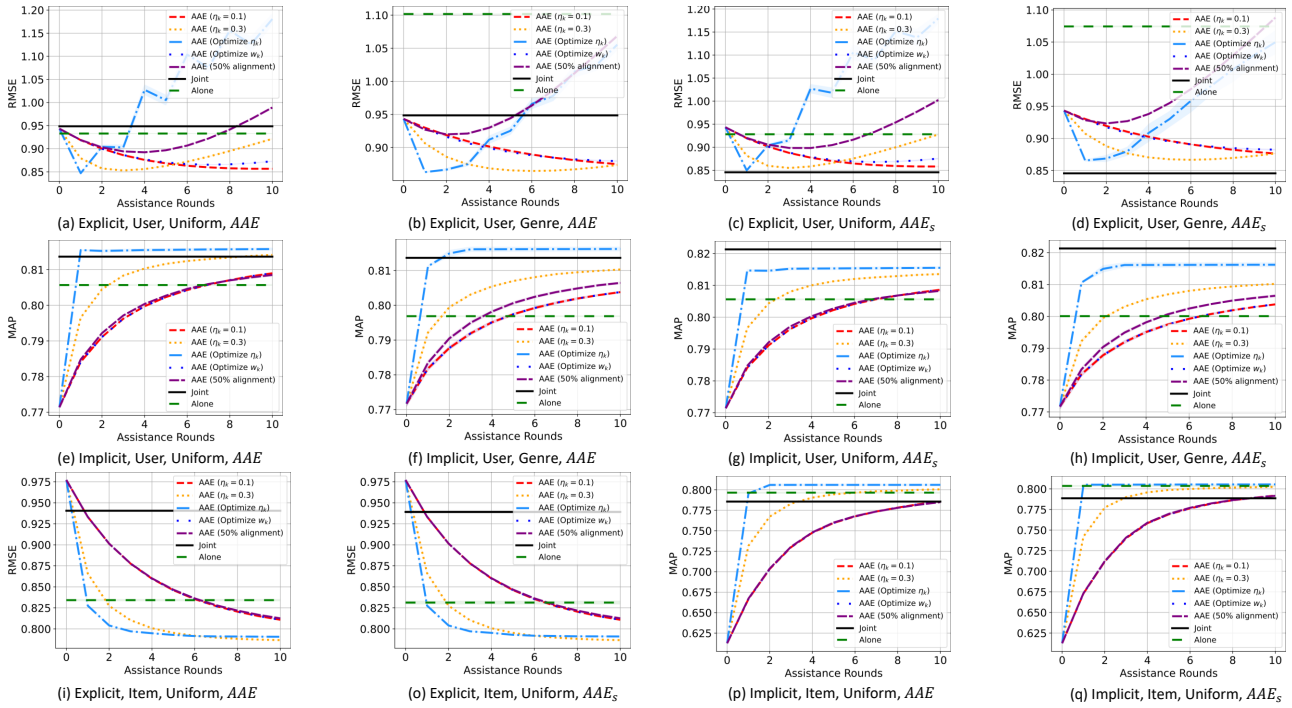


(a) Explicit, User, Uniform, AAE (b) Explicit, User, Genre, AAE (c) Explicit, User, Uniform, AAE_s (d) Explicit, User, Genre, AAE_s

(e) Implicit, User, Uniform, AAE (f) Implicit, User, Genre, AAE (g) Implicit, User, Uniform, AAE_s (h) Implicit, User, Genre, AAE_s

(i) Explicit, Item, Uniform, AAE (o) Explicit, Item, Uniform, AAE_s (p) Implicit, Item, Uniform, AAE (q) Implicit, Item, Uniform, AAE_s

Figure 5: Results of the ML100K dataset across all assistance rounds. MTAL significantly outperforms 'Alone' and performs close to 'Joint.' (a-d) Explicit feedback and User-Alignment. (e-h) Implicit feedback and User-Alignment. (i-q) Item Alignment.



(a) Explicit, User, Uniform, AAE (b) Explicit, User, Genre, AAE (c) Explicit, User, Uniform, AAE_s (d) Explicit, User, Genre, AAE_s

(e) Implicit, User, Uniform, AAE (f) Implicit, User, Genre, AAE (g) Implicit, User, Uniform, AAE_s (h) Implicit, User, Genre, AAE_s

(i) Explicit, Item, Uniform, AAE (o) Explicit, Item, Uniform, AAE_s (p) Implicit, Item, Uniform, AAE (q) Implicit, Item, Uniform, AAE_s

Figure 6: Results of the ML10M dataset across all assistance rounds. MTAL significantly outperforms 'Alone' and performs close to 'Joint.' (a-d) Explicit feedback and User-Alignment. (e-h) Implicit feedback and User-Alignment. (i-q) Item Alignment.