PrivNet: Safeguarding Private Attributes in Transfer Learning for Recommendation

Guangneng Hu
HKUST / Hong Kong, China
njuhgn@gmail.com

Qiang Yang
HKUST / Hong Kong, China
qyang@cse.ust.hk

Abstract

Transfer learning is an effective technique to improve a target recommender system with the knowledge from a source domain. Existing research focuses on the recommendation performance of the target domain while ignores the privacy leakage of the source domain. The transferred knowledge, however, may unintendedly leak private information of the source domain. For example, an attacker can accurately infer user demographics from their historical purchase provided by a source domain data owner. This paper addresses the above privacy-preserving issue by learning a privacy-aware neural representation by improving target performance while protecting source privacy. The key idea is to simulate the attacks during the training for protecting unseen users' privacy in the future, modeled by an adversarial game, so that the transfer learning model becomes robust to attacks. Experiments show that the proposed PrivNet model can successfully disentangle the knowledge benefitting the transfer from leaking the privacy.

1 Introduction

Recommender systems (RSs) are widely used in everyday life ranging from Amazon products (Zhou et al., 2018; Wan et al., 2020) and YouTube videos (Gao et al., 2010; Cheng et al., 2016) to Twitter microblogs (Huang et al., 2016) and news feeds (Wang et al., 2018a; Ma et al., 2019b). RSs estimate user preferences on items from their historical interactions. RSs, however, cannot learn a reliable preference model if there are too few interactions in the case of new users and items, i.e., suffering from the data sparsity issues.

Transfer learning is an effective technique for alleviating the issues of data sparsity by exploiting the knowledge from related domains (Pan et al., 2010; Liu et al., 2018). We may infer user preferences on videos from their Tweet texts (Huang and Lin, 2016), from movies to books (Li et al., 2009), and from news to apps (Hu et al., 2018, 2019). These behaviors across domains are different views of the same user and may be driven by some inherent user interests (Elkahky et al., 2015).

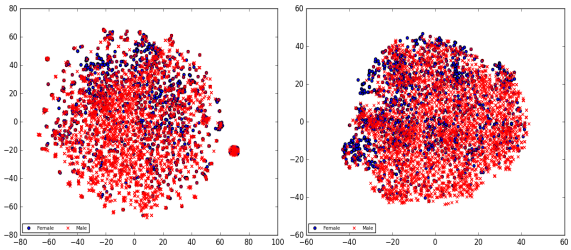


Figure 1: t-SNE projection of transferred representations of users with (left) and without (right) training of PrivNet on the MovieLens-Gender dataset. (see Section 5.5.1 for details)

There is a privacy concern when the source domain shares their data to the target domain due to the ever-increasing user data abuse and privacy regulations (Ramakrishnan et al., 2001; Yang et al., 2019b). Private information contains those attributes that users do not want to disclose, such as gender and age (Jia and Gong, 2018). They can be used to train better recommendation models by alleviating the data sparsity issues to build better user profiles (Zhao et al., 2014; Cheng et al., 2016). Previous work (Weinsberg et al., 2012; Beigi et al., 2020) shows that an attacker can accurately infer a user's gender, age, and occupation from their ratings, recommendation results, and a small amount of users who reveal their demographics.

A technical challenge for protecting user privacy in transfer learning is that the transferred knowledge has dual roles: usefulness to improve target recommendation and uselessness to infer source user privacy. In this work, we propose a novel model (PrivNet) to achieve the two goals by learning privacy-aware transferable knowledge such that it is useful for improving recommendation performance in the target domain while it is useless to

infer private information of the source domain. The key idea is to simulate the attack during the training for protecting unseen users' privacy in the future. The privacy attacker and the recommender are naturally modeled by an adversarial learning game. The main contributions are two-fold:

- PrivNet is the first to address the privacy protection issues, i.e., protecting source user private attributes while improving the target performance, during the knowledge transfer in neural recommendation.

- PrivNet achieves a good tradeoff between the utility and privacy of the source information through evaluation on real-world datasets by comparing with strategies of adding noise (i.e., differential privacy) and perturbing ratings.

2 Related Work

2.1 Transfer learning in recommendation

Transfer learning in recommendation (Cantador et al., 2015) is an effective technique to alleviate the data sparsity issue in one domain by exploiting the knowledge from other domains. Typical methods apply matrix factorization (Singh and Gordon, 2008; Pan et al., 2010; Yang et al., 2017b) and representation learning (Zhang et al., 2016; Man et al., 2017; Yang et al., 2017a; Gao et al., 2019a; Ma et al., 2019a) on each domain and share the user (item) factors, or learn a cluster level rating pattern (Li et al., 2009; Yuan et al., 2019). Transfer learning is to improve the target performance by exploiting knowledge from auxiliary domains (Pan and Yang, 2009; Elkahky et al., 2015; Zhang and Yang, 2017; Chen et al., 2019; Gao et al., 2019b). One transfer strategy (two-stage) is to initialize a target network with transferred representations from a pre-trained source network (Oquab et al., 2014; Yosinski et al., 2014). Another transfer strategy (end-to-end) is to transfer knowledge in a mutual way such that the source and target networks benefit from each other during the training, with examples including the cross-stitch networks (Misra et al., 2016) and collaborative cross networks (Hu et al., 2018). These transfer learning methods have access to the input or representations from source domain. Therefore, it raises a concern on privacy leaks and provides an attack possibility during knowledge transfer.

2.2 Privacy-preserving techniques

Existing privacy-preserving techniques mainly belong to three research threads. One thread adds noise (e.g., differential privacy (Dwork et al., 2006)) to the released data or the output of recommender systems (McSherry and Mironov, 2009; Jia and Gong, 2018; Meng et al., 2018; Wang et al., 2018b; Wang and Zhou, 2020). One thread perturbs user profiles such as adding (or deleting/changing) dummy items to the user history so that it hides the user's actual ratings (Polat and Du, 2003; Weinsberg et al., 2012). Adding noise and perturbing ratings may still suffer from privacy inference attacks when the attacker can successfully distinguish the true profiles from the noisy/perturbed ones. Furthermore, they may degrade performance since data is corrupted. Another thread uses adversary loss (Resheff et al., 2019; Beigi et al., 2020) to formulate the privacy attacker and the recommender system as an adversarial learning problem. However, they face the data sparsity issues. A recent work (Ravfogel et al., 2020) trains linear classifiers to predict a protected attribute and then remove it by projecting the representation on its null-space. Some other work uses encryption and federated learning so as to protect the personal data without affecting performance (Nikolaenko et al., 2013; Chen et al., 2018; Wang et al., 2019). They suffer from efficiency and scalability due to high cost of computation and communication.

3 Problem Statement

We have two domains, a source domain S and a target domain T. User sets in two domains are shared, denoted by $\mathcal{U}$ (of size $m = |\mathcal{U}|$). Denote item sets in two domains by $\mathcal{I}_S$ and $\mathcal{I}_T$ (of size $n_S = |\mathcal{I}_S|$ and $n_T = |\mathcal{I}_T|$), respectively. For the target domain, a binary matrix $\boldsymbol{R}_T \in \mathbb{R}^{m \times n_T}$ describes the user-item interactions, where the entry $r_{ui} \in \{0, 1\}$ equals 1 if user u has an interaction with item i and 0 otherwise. Similarly, for the source domain, we have $\boldsymbol{R}_S \in \mathbb{R}^{m \times n_S}$ and the entry $r_{uj} \in \{0, 1\}$. We reserve i and j to index the target and source items, respectively. Let $\boldsymbol{Y}^p \in \mathbb{R}^{m \times c_p}$ denote the p-th user private attribute (e.g., p='Gender') matrix where each entry $y_{u,p}$ is the value of the p-th private information for user u (e.g., $y_{u,p}$='Male') and there are c_p choices. Denote all n private attributes data by $\boldsymbol{Y} = \{\boldsymbol{Y}^p\}_{p=1}^{n}$ (e.g., Gender, Age). We can define the problem as follows:

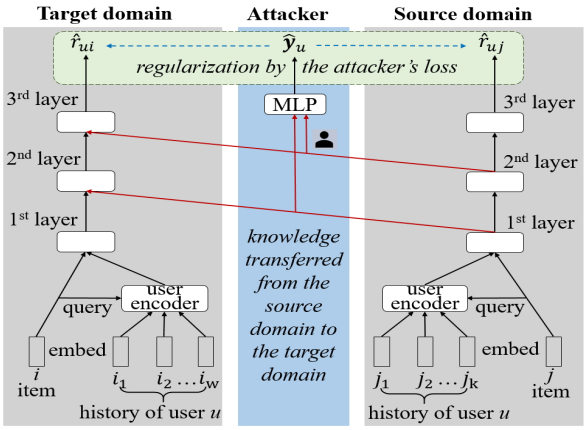


Figure 2: Architecture of PrivNet (a version of three layers). It has two components: the recommender and privacy attacker. The recommender (the left & right parts, see Section 4.1) is a representation-based transfer learning model where the red arrows indicate the representations transferred from the source domain to the target domain in a multilayer way. The privacy attacker (the middle part, see Section 4.2) marked by an avatar infers user privacy from the transferred representations. PrivNet (see Section 4.3) exploits the knowledge from the source domain with regularization from the adversary loss of the attacker indicated by the dotted box.

PROBLEM: Privacy-aware transfer learning in recommendation.

INPUT: $\boldsymbol{R}_T, \boldsymbol{R}_S, \boldsymbol{Y}$.

OUTPUT: Generate a ranked list of items for users in the target domain.

REQUIRE: An attacker is difficult to infer the source user private attributes from the knowledge transferred to the target domain.

ASSUMPTION: Some users $\mathcal{U}^{pub} \subset \mathcal{U}$ share their private information with the public profile.

4 The Proposed Framework

The architecture of PrivNet is shown in Figure 2. It has two components, a recommender and an attacker. We introduce the recommender (Section 4.1) and present an attack against it (Section 4.2). We propose PrivNet to protect source user privacy during the knowledge transfer (Section 4.3).

4.1 Recommender

In this section, we introduce a novel transfer-learning recommender which has three parts, a source network for the source domain, a target network for the target domain, and a knowledge transfer unit between the two domains.

Target network The input is a pair of (user, item) and the output is their matching degree. The user is represented by their w-sized historical items $[i_1, ..., i_w]$. First, an item embedding matrix $\boldsymbol{A}_T$ projects the discrete item indices to the d-dimensional continuous representations: $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i_*}$ where $* \in [1, 2, ..., w]$. Second, the user representation $\boldsymbol{x}_u$ is computed by the user encoder module based on an attention mechanism by querying their historical items with the predicted item: $\boldsymbol{x}_u = \sum_{i_*} \alpha_{i_*} \boldsymbol{x}_{i_*}$, where $\alpha_{i_*} = \boldsymbol{x}_i^T \boldsymbol{x}_{i_*}$ (normalized: $\sum \alpha_{i_*} = 1$). Third, a multilayer perceptron (MLP) f_T parameterized by ϕ_T is used to compute target preference score (the notation $[\cdot, \cdot]$ denotes concatenation):

$$\hat{r}_{ui} = P(r_{ui}|u, i; \theta_T) = f_T([\boldsymbol{x}_u, \boldsymbol{x}_i]),$$

where $\theta_T = \{\boldsymbol{A}_T, \phi_T\}$ is the model parameter.

Source network Similar to the three-step computing process in the target network, we compute the source preference score by: $\hat{r}_{uj} = P(r_{uj}|u, j; \theta_S) = f_S([\boldsymbol{x}_u, \boldsymbol{x}_j])$ where $\theta_S = \{\boldsymbol{A}_S, \phi_S\}$ is the model parameter with item embedding matrix $\boldsymbol{A}_S$ and multilayer perceptron ϕ_S.

Transfer unit The transfer unit implements the knowledge transfer from the source to the target domain. Since typical neural networks have more than one layer, say L, the representations are transferred in a multilayer way. Let $\boldsymbol{x}_{u|\#}^{\ell}$ where $\# \in \{S, T\}$ be user u's source/target representation in the ℓ-th layer ($\ell = 1, 2, ..., L-1$) where $\boldsymbol{x}_{u|S}^1 = [\boldsymbol{x}_u, \boldsymbol{x}_j]$ and $\boldsymbol{x}_{u|T}^1 = [\boldsymbol{x}_u, \boldsymbol{x}_i]$. The transferred representation is computed by projecting the source representation to the space of target representations with a translation matrix $\boldsymbol{H}^{\ell}$:

$$\boldsymbol{x}_{u|trans}^{\ell} = \boldsymbol{H}^{\ell} \boldsymbol{x}_{u|S}^{\ell}, \tag{1}$$

With the knowledge from the source domain, the target network learns a linear combination of the two input activations from both networks and then feeds these combinations as input to the successive layer's filter. In detail, the $(\ell+1)$-th layer's input of the target network is computed by: $\boldsymbol{W}_T^{\ell} \boldsymbol{x}_{u|T}^{\ell} + \boldsymbol{x}_{u|trans}^{\ell}$ where $\boldsymbol{W}_T^{\ell}$ is the connection weight matrix in the ℓ-th layer of the target network. The total transferred knowledge is concatenated by all layers's representations:

$$\boldsymbol{x}_{u|trans} = [\boldsymbol{x}_{u|trans}^{\ell}]_{\ell=1}^{L-1}. \tag{2}$$

Objective The recommender minimizes the neg-

ative logarithm likelihood:

$$\mathcal{L}(\theta) = -\sum_{D_T} \log P(r_{ui}|u,i;\theta_T) - \sum_{D_S} \log P(r_{uj}|u,j;\theta_S), \quad (3)$$

where $\theta = \{\theta_T, \theta_S, \{\boldsymbol{H}^\ell\}_{\ell=1}^{L-1}\}$, D_T and D_S are target and source training examples, respectively. We introduce how to generate them in Section 4.4.

4.2 Attacker

The recommender can fulfil the Problem 1 (see Section 3) if there is no attacker existing. A challenge for the recommender is that it does not know the attacker models in advance. To address this challenge, we add an attacker component during the training to simulate the attacks for the test. By integrating a simulated attacker into the recommender, it can deal with the unseen attacks in the future. In this section, we introduce an attacker to infer the user private information from the transferred knowledge. In the next Section 4.3, we will introduce an adversarial recommender by exploiting the simulated attacker to regularize the recommendation process in order to fool the adversary so that it can protect the privacy of unseen users in the future.

The attacker model predicts the private user attribute from their source representation sent to the target domain:

$$\hat{y}_{u,p} = P(y_{u,p}|\boldsymbol{x}_{u|trans};\theta_p) = f_p(\boldsymbol{x}_{u|trans};\theta_p), \quad (4)$$

where $\hat{y}_{u,p}$ is the predicted value of user u's p-th private attribute and $p = 1, ..., n$. f_p is the prediction model parameterized by θ_p. Note that, an attacker can use any prediction model and here we use an MLP due to its nonlinearity and generality.

For all n private user attributes, the attacker model minimizes the multitask loss:

$$\mathcal{L}(\Theta) = -\frac{1}{n}\sum_p \sum_{D_p} \log P(y_{u,p}|\boldsymbol{x}_{u|trans};\theta_p), \quad (5)$$

where $\Theta = \{\theta_p\}_{p=1}^n$ and D_p is training examples for the p-th attribute. We introduce how to generate them in Section 4.4.

4.3 PrivNet

So far, we have introduced a recommender to exploit the knowledge from a source domain and a privacy attacker to infer user private information from the transferred knowledge. To fulfill the Problem 1 in Section 3, we need to achieve two goals: improving the target recommendation and protecting the source privacy. In this section, we propose a novel model (PrivNet) by exploiting the attacker component to regularize the recommender.

Since we have two rival objectives (i.e., target quality and source privacy), we adopt the adversarial learning technique (Goodfellow et al., 2014) to learn a privacy-aware transfer model. The generator is a privacy attacker which tries to accurately infer the user privacy, while the discriminator is an recommender which learns user preferences and deceives the adversary. The recommender of PrivNet minimizes:

$$\tilde{\mathcal{L}}(\theta) = \mathcal{L}(\theta) - \lambda\mathcal{L}(\Theta), \quad (6)$$

where the hyperparameter λ controls the influence from the attacker component. PrivNet seeks to improve the recommendation quality (the first term on the right-hand side) and fools the adversary by maximizing the loss of the adversary (the second term,). The adversary has no control over the transferred knowledge, i.e., $\boldsymbol{x}_{u|trans}$. Losses of the two components are interdependent but they optimize their own parameters. PrivNet is a general framework since both the recommender and the attacker can be easily replaced by their variants. PrivNet reduces to privacy-agnostic transfer model when $\lambda = 0$.

4.4 Generating Training Examples

We generate D_T and D_S as follows and take the target domain as an example since the procedure is the same for the source domain. Suppose we have a whole item interaction history for some user u, say $[i_1, i_2, ..., i_l]$. Then we generate the positive training examples by sliding over the sequence of the history: $D_T^+ = \{([i_w]_{w=1}^{c-1}, i_c) : c = 2, ..., l\}$. We adopt the random negative sampling technique (Pan et al., 2008) to generate the corresponding negative training examples $D_T^- = \{([i_w]_{w=1}^{c-1}, i'_c) : i'_c \notin [i_1, i_2, ..., i_l]\}$. As the same with (Weinsberg et al., 2012; Beigi et al., 2020), we assume that some users $\mathcal{U}^{pub} \subset \mathcal{U}$ share their private attributes with the public profile. Then we have the labelled privacy data $D_{priv} = \{D_p\}_{p=1}^n$ where $D_p = \{(u, y_{u,p}) : u \in \mathcal{U}^{pub}\}$.

4.5 Model Learning

The training process of PrivNet is illustrated in Algorithm 1. Lines 1-3 are to optimize the privacy

Algorithm 1: Training PrivNet.

Input: Target data D_T, source data D_S, privacy data D_{priv}, hyperparameter λ
Output: PrivNet
for *number of training iterations* **do**
1. Accumulate (user, attributes) with a mini-batch $(\mathcal{U}_b, \mathcal{Y}_b)$ from D_{priv}
2. Feed users $\mathcal{U}_b$ and their history in D_S into the source network (see Sec. 4.1) so as to generate the transferred knowledge $\mathcal{X}_{b|S}$
3. Update Θ using examples $(\mathcal{X}_{b|S}, \mathcal{Y}_b)$ via gradient descent over $\mathcal{L}(\Theta)$.
4. Update θ using mini-batch examples from D_S and D_T with adversary loss via gradient descent over $\tilde{\mathcal{L}}(\theta)$.

end
The gradient-based updates can use any standard gradient-based learning rule. Deep learning library (e.g., TensorFlow) can automatically calculate gradients.

part related parameter, i.e., Θ in $\mathcal{L}(\Theta)$. On line 1, it creates a mini-batch size examples from data D_{priv}. Each example contains a user and their corresponding private attributes $(u, \{y_{u,p}\}_{p=1}^{n})$. On line 2, it feeds users and their historical items in the source domain to the source network so as to generate transferred knowledge $\boldsymbol{x}_{u|trans}$. On line 3, the transferred knowledge and their corresponding private attributes $(\boldsymbol{x}_{u|trans}, \{y_{u,p}\}_{p=1}^{n})$ are used to train the privacy attacker component by descending its stochastic gradient using the mini-batch examples: $\nabla_{\Theta}\mathcal{L}(\Theta)$. Line 4 is to optimize the recommender part related parameter, i.e., θ by descending its stochastic gradient with adversary loss using mini-batch examples:$\nabla_{\theta}\tilde{\mathcal{L}}(\theta)$.

4.6 Complexity Analysis

The parameter complexity of PrivNet is the addition of its recommender component and the privacy component. The embedding matrices of the recommender dominate the number of parameters as they vary with the input. As a result, the parameter complexity of PrivNet is $\mathcal{O}(d \cdot (n_S + n_T))$ where d is the embedding dimension, and n_S and n_T are the number of items in the source and target domains respectively.

The learning complexity of PrivNet divides into two parts: the forward prediction and backward parameter update. The forward prediction of PrivNet is the addition of its recommender component and two times of the privacy component since the recommender component needs the loss from the privacy component. The complexity of backward parameter update is the addition of its recommender component and the privacy component since they optimize their own parameters.

Data	#user	Target domain		Source domain		Private attribute
		#item	#rating	#item	#rating	
FS	29,515	28,199	357,553	28,407	467,810	G
ML	5,967	2,049	274,115	1,484	299,830	G, A

Table 1: Statistics of datasets. (G=Gender, A=Age)

5 Experiments

In this section, we conduct experiments to evaluate both recommendation performance and privacy protection of PrivNet.

5.1 Dataset

We evaluate on the following real-world datasets.

Foursquare (FS) It is a public available data on user-venue checkins (Yang et al., 2019a). The source and target domains are divided by the checkin's time, i.e., dealing with the covariate shift issues where the distribution of the input variables change between the old data and the newly collected one. The private user attribute is Gender.

MovieLens (ML) It is a public available data on user-movie ratings (Harper and Konstan, 2016). We reserve those ratings over three stars as positive feedbacks. The source and target domains are divided by the movie's release year, i.e., transferring from old movies to the new ones. The private user attributes are Gender and Age. Following (Beigi et al., 2020), we categorize Age into three groups: over-45, under-35, and between 35 and 45.

The statistics are summarized in Table 1 and we can see that all of the datasets have more than 99% sparsity. It is expected that the transfer learning technique is helpful to alleviate the data sparsity issues in these real-world recommendation services.

5.2 Experimental Setting

5.2.1 Evaluation Metric

For privacy evaluation, we follow the protocol in (Jia and Gong, 2018) to randomly sample 80% of users as the training set and treat the remaining users as the test set. The users in the training set has publicly shown their private information while

Hyperparameter	Setting
train:valid:test	7:1:2
user representation size	80
item representation size	80
history length cutoff (#items)	10
neural collaborative filtering layers	[80, 64]
attention unit layers	[80, 64]
number of transfer layers	1
negative sampling ratio for training	1
test positive:negative	1:99
clip norm	5
batch size	128
bias init	0
weight init	Glorot uniform
embedding init	Glorot uniform
learning rate	5e-4
optimizer	Adam
activation function	sigmoid
total epochs (with early stopping)	50

Table 2: Setting of hyperparameters.

the users in the test set keep it private. We split a small data from the training set as the validation set where the ratio is train:valid:test=7:1:2. For privacy metrics, we compute Precision, Recall, and F1-score in a weighted[1] way which are suitable for imbalanced data distribution (Fawcett, 2006). We report results for each private attribute. We first calculate metrics for each label, and then compute their average weighted by support (the number of true instances for each label). A lower value indicates better privacy protection.

For recommendation evaluation, we follow the leave-one-out strategy in (He et al., 2017), i.e., reserving the latest one interaction as the test item for each user, then randomly sampling a number of (e.g., 99) negative items that are not interacted by the user. We evaluate how well the recommender can rank the test item against these negative ones. We split a small data from the training set as the validation set where the ratio is train:valid:test=7:1:2. For recommendation metrics, we compute hit ratio (HR), normalized discounted cumulative gain (NDCG), mean reciprocal rank (MRR), and AUC for top-K (default $K = 10$) item recommendation (Gao et al., 2019a). A higher value indicates better recommendation.

5.2.2 Implementation

All methods are implemented using TensorFlow. Parameters are initialized by default. The optimizer is the adaptive moment estimation with learning rate 5e-4. The size of mini-batch is 128 with negative sampling ratio 1. The embedding size is 80 while the MLP has one hidden layer with size 64. The history size is 10. λ is 1 in Eq. (6). The noise level is 10%. The number of dummy items are 5. The privacy related metrics are computed by Python scikit-learn library. The setting of hyperparameters used to train our model and the baselines is summarized in Table 2.

[1]Note, the weighted F1 values are not necessarily equal to the harmonic mean of the corresponding Precision and Recall values.

Methods	Knowledge transfer	Privacy protection (+strategy)
BPRMF (Rendle et al., 2009)	✗	✗
MLP (He et al., 2017)	✗	✗
CSN (Misra et al., 2016)	✓	✗
CoNet (Hu et al., 2018)	✓	✗
BlurMe (Weinsberg et al., 2012)	✗	✓ (+perturbation)
LDP (Bassily and Smith, 2015)	✗	✓ (+noise)
PrivNet (ours)	✓	✓ (+adversary)

Table 3: Categorization of comparing methods.

5.3 Baseline

We compare PrivNet with various kinds of baselines as summarized in Table 3.

The following methods are privacy-agnostic. *BPRMF*: Bayesian personalized ranking (Rendle et al., 2009) is a latent factors approach which learns user and item factors via matrix factorization. *MLP*: Multilayer perceptron (He et al., 2017) is a neural CF approach which learns the user-item interaction function using neural networks. *CSN*: The cross-stitch network (Misra et al., 2016) is a deep transfer learning model which couples the two basic networks via a linear combination of activation maps using a translation scalar. *CoNet*: Collaborative cross network (Hu et al., 2018) is a deep transfer learning method for cross-domain recommendation which learns linear combination of activation maps using a translation matrix.

The following methods are privacy-aware. *BlurMe*: This method (Weinsberg et al., 2012) perturbs a user's profile by adding dummy items to their history. It is a representative of the perturbation-based technique to recommend items while protect private attributes. *LDP*: Local differential privacy (Bassily and Smith, 2015) modifies user-item ratings by adding noise to them based on the differential privacy. It is a representative of the noise-based technique to recommend items while protect private attributes. Note, the original LDP and BlurMe are single-domain models which are also used as comparing baselines in (Beigi et al.,

Dataset	Metric	BPRMF	MLP	CSN	CoNet	BlurMe	LDP	PrivNet
Foursquare	HR	36.5	47.0	52.7	53.4*	52.6	44.5	**54.3**
	NDCG	22.0	31.5	35.9	36.3*	35.4	29.9	**36.8**
	MRR	17.6	31.9	35.0*	**35.3**	32.1	27.1	33.4
MovieLens	HR	53.0	77.4	82.7	77.1	85.7	85.8*	**86.0**
	NDCG	37.0	50.5	55.7	50.7	69.7	**69.9**	**69.9**
	MRR	32.0	44.5	49.3	44.6	**65.9**	**65.9**	65.7*

Table 4: Comparison results of different methods on recommendation performance. The bold face indicates the best result while the star mark indicates the second best.

2020). To be fair and to investigate the influence of privacy-preserving strategies, we replace the adversary strategy of PrivNet with the strategy of LDP (adding noise) and BlurMe (perturbing ratings), and keep the other components the same.

5.4 Result on Recommendation Performance

The results of different methods on recommendation are summarized in Table 4. A higher value indicates better recommendation performance.

Comparing with the privacy-agnostic methods (BPRMF, MLP, CSN, and CoNet), PrivNet is superior than them with a large margin on the MovieLens dataset. This shows that PrivNet is effective in recommendation while it protects the source private attributes. Since these four methods represent a wide range of typical recommendation methods (matrix factorization, neural CF, transfer learning), we can see that the architecture of PrivNet is a reasonable design for recommender systems.

Comparing with the privacy-aware methods (LDP and BlurMe), we can see that LDP significantly degrades recommendation performance with a reduction about six to ten percentage points on the Foursquare dataset. This shows that LDP suffers from the noisy source information since it harms the usefulness of the transferred knowledge to the target task. For BlurMe, we can see that BlurMe still degrades recommendation performance on the Foursquare dataset, for example with relative 4.0% performance reduction in terms of MRR. This shows that BlurMe suffers from the perturbed source information since it harms the usefulness of the transferred knowledge to the target task.

Among the privacy-aware methods, PrivNet achieves the best recommendation performance in terms of all HR, NDCG, and MRR on the Foursquare dataset, and the best in terms of HR on the MovieLens dataset. It shows that PrivNet is better for improving the usefulness of the transferred knowledge by comparing with LDP and BlurMe.

Dataset	Metric	LDP	BlurMe	PrivNet
Foursquare	Precision	**64.7**	73.2	66.8
	Recall	75.2	75.3	**71.1**
	F1	**66.0**	66.7	68.1
MovieLens-G	Precision	73.4	**69.4**	70.9
	Recall	75.4	**71.7**	72.5
	F1	73.6	70.1	**62.0**
MovieLens-A	Precision	63.8	**54.6**	55.4
	Recall	65.5	58.1	**57.9**
	F1	61.4	54.2	**46.3**

Table 5: Comparison results on privacy protection. The bold face indicates the best result (the lower the better).

In summary, PrivNet is effective in transferring the knowledge, showing that the adversary strategy of PrivNet achieves state-of-the-art performance by comparing with the strategies of adding noise (LDP) and perturbing ratings (BlurMe).

5.5 Result on Privacy Protection

The results of different methods on privacy inference are summarized in Table 5 (Note, there are no results for the four privacy-agnostic methods). A lower value indicates better privacy protection.

Comparing PrivNet and BlurMe, we can see that the perturbation method by adding dummy items still suffers from privacy inference attacks in terms of Precision and Recall on the Foursquare dataset, and in terms of F1 on the MovieLens dataset. The reason may be that the attacker can effectively distinguish the true profiles from the dummy items. That is, it can accurately learn from the true profiles while ignore the dummy items. Comparing PrivNet and LDP, we can see that adding noise to ratings still suffers from privacy inference attacks in terms of Recall on the Foursquare dataset, and in terms of all three metrics on the MovieLens dataset. It implies that the occurrence of a rating, regardless of its numeric value (true or noisy), leaks the user

privacy. That is, the binary event of excluding or including an item in a user's profile is a signal for user privacy inference nearly as strong as numerical ratings. In particular, there are 50 movies rated by Female only (e.g., Country Life (1994)) while 350 by Male only (e.g., Time Masters (1982)). Adding noise to these ratings may not influence the inference of Gender for these users very much.

PrivNet achieves nearly half the best results on privacy protection in terms of three evaluation metrics on the two datasets. It has significantly lower F1 scores in comparison to all baselines on the MovieLens dataset. It is effective to hide private information during the knowledge transfer. By simulating the attacks during the training, PrivNet is prepared against the malicious attacks for unseen users in the future. In summary, PrivNet is an effective source privacy-aware transfer model such that it makes the malicious attackers more difficult to infer the source user privacy during the knowledge transfer, compared with the strategies of adding noise (LDP) and perturbing ratings (BlurMe).

5.5.1 Clustering

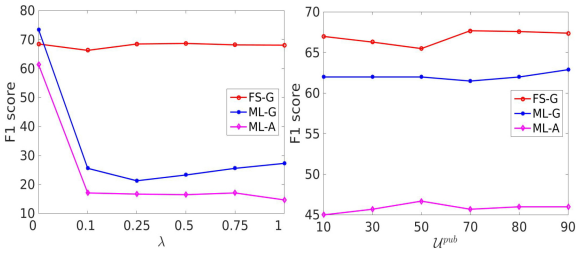
Figure 1 shows t-SNE projections of 4,726 users' transferred representations on the MovieLens-Gender dataset. These user vectors are computed from the user encoder as shown in Figure 2. We can see that the vectors are more mixed distributed among male and female users with the training of PrivNet. In contract, the vectors for female users are clustered on the top-left corner while male users are on the bottom-right without the training of PrivNet ($\lambda = 0$, see Section 5.6.1). To quantify the difference, we perform K-means clustering on the user vectors where K=2, and calculate the V-measure (Rosenberg and Hirschberg, 2007) which assesses the degree of overlap between the 2 clusters and the Gender groups. The measure is 0.0119 and 0.0027 respectively for without and with training of PrivNet. Note that a lower measure is better since we do not want to the two classes to be easily separable.

5.6 Parameter Sensitivity

In this section, we analyse the model ablation, impact of privacy inference component, and impact of public users who share their profiles.

5.6.1 Model Ablation

The key component of PrivNet is the adversary loss used to regularize the recommender. We remove this component to show its necessity to protect the private attributes by setting the $\lambda = 0$ in Eq. (6). The results are summarized in Table 6. As we expect, PrivNet without adversary loss is most vulnerable to privacy attacks since it has no privacy defense. There is a significant drop in terms of all three privacy-related metrics without this model component.



(a) Privacy attacker. (b) Public users.

Figure 3: Impact of privacy component and public users. (FS-G: Foursquare-Gender, ML-G: MovieLens-Gender, ML-A: MovieLens-Age)

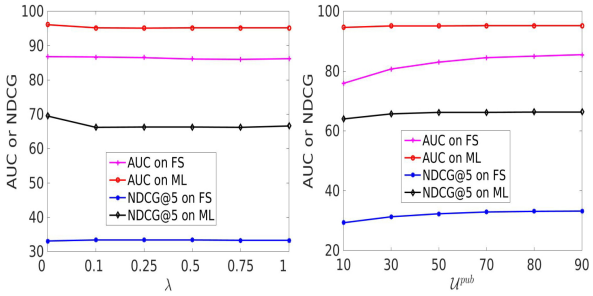
5.6.2 Impact of Privacy Component

We vary the λ (see Eq. (6)) of privacy component with $\{0, 0.1, 0.25, 0.5, 0.75, 1.0\}$ to show the its impact on privacy protection and recommendation (where $\lambda = 0$ corresponds to without privacy attack component, see also Table 6). Figure 3a shows the impact on privacy protection. The privacy inference generally becomes more difficult with the increase of λ, showing that the privacy inference component of PrivNet is a key factor for protecting the user privacy in the source domain. In particular, all results of $\lambda \neq 0$ are better than that of $\lambda = 0$ in hiding the private information. Privacy inference results, however, are subtle among different private attributes and evaluation metrics. On the Foursquare dataset, F1 decreases at first (until λ to 0.1), then it increases. On the MovieLens-Gender dataset, the F1 score decreases at first (until λ to 0.25) and then it increases. It means that the private information is obscured more successfully in the beginning but less in the end. The reason may be that the model overfits by increasing the value of λ and leads to an inaccurate estimation of privacy inference. On the MovieLens-Age dataset, the F1 score consistently decreases with the increase of λ.

Figure 4a shows the impact on recommendation performance. The recommendation performance decreases with λ increasing from 0 to 0.1 on the MovieLens dataset, showing that increasing the impact of privacy inference component harms the

Adversary loss?	Foursquare			MovieLens-Gender			MovieLens-Age		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
No	73.2	75.4	68.5	73.6	75.6	73.5	60.8	65.3	61.4
Yes	**66.8**	**71.1**	**68.1**	**70.9**	**72.5**	**62.0**	**55.4**	**57.9**	**46.3**

Table 6: Necessity of adversary loss to regularize the recommender (lower value better privacy protection).



(a) Privacy component (b) Public users

Figure 4: Parameter sensitivity for recommendation.

No.	Movie	Genre	Attn weight
0	Chicken Run	Animation, Children, Comedy	*0.127*
1	X-Men	Action, Sci-Fi	0.069
2	Mission: Impossible	Action, Adventure, Mystery	0.001
3	Titan A.E.	Adventure, Animation, Sci-Fi	0.059
4	The Perfect Storm	Action, Adventure, Thriller	0.056
5	Gone in 60 Seconds	Action, Crime	0.053
6	Schindler's List	Drama, War	0.098
7	The Shawshank Redemption	Drama	**0.331**
8	The Matrix	Action, Sci-Fi, Thriller	0.062
9	Shakespeare in Love	Comedy, Romance	*0.140*
10	Howards End	Drama	N/A

Table 7: Example: Capturing short-/long-term user interests and high-level category relationship among items.

recommendation quality to some extent.

5.6.3 Impact of Public Users

We vary the percentage of public users $\mathcal{U}^{pub}$ (see Section 3) with $\{10, 30, 50, 70, 80, 90\}$. Figure 3b shows the impact on the privacy inference. It is surprising that the privacy inference does not become more easy with the increase of public users. On the Foursquare dataset, it infers inaccurately until the percentage increases to 50% and then accurately until to 80% in terms of F1. This shows that the adversary strategy of PrivNet is effective to protect unseen users' privacy when only a small number of users (e.g., 10%) reveal their profiles for the training. On the MovieLens dataset, it infers inaccurately after 50% until to 80% in terms of F1.

Figure 4b shows the impact on recommendation performance. Since the amount of public users controls how much knowledge is shared between the source and target domains, the recommendation performance improves with the increasing amount of public users. In summary, PrivNet is favourable in practice since it can achieve a good tradeoff on the utility and privacy when only a small amount of users reveal their profiles to the public.

5.7 Case Study

One advantage of PrivNet is that it can explain which item in a user's history matters the most for a candidate item by using the attention weights. Table 7 shows an example of interactions between a user's historical movies (No. 0∼9) and the candidate movie (No. 10). We can see that the latest movie matters a lot since the user interests may remain the same during a short period. The oldest movie, however, also has some impact on the candidate movie, reflecting that the user interests may mix with a long-term characteristic. PrivNet can capture these subtle short-/long-term user interests. Furthermore, the movie (No. 7) belonging to the same genre as the candidate movie matters the most. PrivNet can also capture this high-level category relationship.

6 Conclusion

We presented an attack scenario to infer the private user attributes from the transferred knowledge in recommendation, raising the issues of source privacy leakage beyond target performance. To protect user privacy in the source domain, a privacy-aware transfer model (PrivNet) is proposed beyond improving the performance in the target domain. It is effective in terms of recommendation performance and privacy protection, achieving a good trade-off between the utility and privacy of the transferred knowledge. In future works, we want to relax the assumption that the private user attributes need to provide in advance in order to train the privacy inference component for protecting unseen users.

Acknowledgement

We thank Dr. Yu Zhang for insightful discussion. We thank the new publication paradigm, i.e., "Findings of ACL: EMNLP 2020", which makes

this paper indexed in the ACL anthology. The work was supported by Hong Kong CERG projects 16209715/16244616, and Hong Kong PhD Fellowship Scheme.

References

R. Bassily and A. Smith. 2015. Local, private, efficient protocols for succinct histograms. In *ACM STOC*.

G. Beigi, A. Mosallanezhad, R. Guo, H. Alvari, et al. 2020. Privacy-aware recommendation with private-attribute protection using adversarial learning. In *ACM WSDM*.

Iván Cantador, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. 2015. Cross-domain recommender systems. In *Recommender systems handbook*, pages 919–959.

Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An efficient adaptive transfer neural network for social-aware recommendation. In *ACM SIGIR*.

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

H. Cheng, L. Koc, J. Harmsen, T. Shaked, et al. 2016. Wide & deep learning for recommender systems. In *ACM RecSys Workshop*.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*.

A. Elkahky, Y. Song, and X. He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*.

T. Fawcett. 2006. An introduction to roc analysis. *Pattern recognition letters*.

C. Gao, X. Chen, F. Feng, K. Zhao, et al. 2019a. Cross-domain recommendation without sharing user-relevant data. In *WWW*.

Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019b. Neural multi-task recommendation from multi-behavior data. In *IEEE ICDE*.

W. Gao, Y. Tian, T. Huang, and Q. Yang. 2010. Vlogging: A survey of videoblogging technology on the web. *ACM Computing Surveys*.

I. Goodfellow, J. Pouget, M. Mirza, B. Xu, et al. 2014. Generative adversarial nets. In *NIPS*.

F. Harper and J. Konstan. 2016. The movielens datasets: History and context. *ACM TIST*.

X. He, L. Liao, H. Zhang, L. Nie, et al. 2017. Neural collaborative filtering. In *WWW*.

G. Hu, Y. Zhang, and Q. Yang. 2018. Conet: Collaborative cross networks for cross-domain recommendation. In *ACM CIKM*.

Guangneng Hu, Yu Zhang, and Qiang Yang. 2019. Transfer meets hybrid: a synthetic approach for cross-domain collaborative filtering with text. In *The World Wide Web Conference*, pages 2822–2829.

H. Huang, Q. Zhang, Y. Gong, and X. Huang. 2016. Hashtag recommendation using end-to-end memory networks with hierarchical attention. In *COLING*.

Y. Huang and S. Lin. 2016. Transferring user interests across websites with unstructured text for cold-start recommendation. In *EMNLP*.

J. Jia and N. Gong. 2018. Attriguard: A practical defense against attribute inference attacks via adversarial machine learning. In *USENIX Security*.

B. Li, Q. Yang, and X. Xue. 2009. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *IJCAI*.

B. Liu, Y. Wei, Y. Zhang, Z. Yan, and Q. Yang. 2018. Transferable contextual bandit for cross-domain recommendation. In *AAAI*.

Muyang Ma, Pengjie Ren, Yujie Lin, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019a. Pi-net: A parallel information-sharing network for shared-account cross-domain sequential recommendations. In *ACM SIGIR*.

Y. Ma, L. Zong, Y. Yang, and J. Su. 2019b. News2vec: News network embedding with subnode information. In *EMNLP*.

Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. 2017. Cross-domain recommendation: an embedding and mapping approach. In *IJCAI*.

F. McSherry and I. Mironov. 2009. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *ACM SIGKDD*.

X. Meng, S. Wang, K. Shu, J. Li, et al. 2018. Personalized privacy-preserving social recommendation. In *AAAI*.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *IEEE CVPR*.

V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, et al. 2013. Privacy-preserving matrix factorization. In *ACM CCS*.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724.

R. Pan, Y. Zhou, B. Cao, N. Liu, et al. 2008. One-class collaborative filtering. In *IEEE ICDM*.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

W. Pan, E. Xiang, N. Liu, and Q. Yang. 2010. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*.

H. Polat and W. Du. 2003. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *IEEE ICDM*.

N. Ramakrishnan, B. Keller, B. Mirza, A. Grama, and G. Karypis. 2001. Privacy risks in recommender systems. *IEEE Internet Computing*.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. *ACL*.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*.

Yehezkel Resheff, Yanai Elazar, Moni Shahar, and Oren Shalom. 2019. Privacy and fairness in recommender systems via adversarial training of user representations. In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP*.

Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *SIGKDD*.

M. Wan, J. Ni, R. Misra, and J. McAuley. 2020. Addressing marketing bias in product recommendations. In *ACM WSDM*.

H. Wang, F. Zhang, X. Xie, and M. Guo. 2018a. Dkn: Deep knowledge-aware network for news recommendation. In *WWW*.

J. Wang, Q. Tang, A. Arriaga, and P. Ryan. 2019. Novel collaborative filtering recommender friendly to privacy protection. In *IJCAI*.

J. Wang and Z. Zhou. 2020. Differentially private learning with small public data. In *AAAI*.

Y. Wang, Q. Gu, and D. Brown. 2018b. Differentially private hypothesis transfer learning. In *ECML-PKDD*.

U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft. 2012. Blurme: Inferring and obfuscating user gender based on ratings. In *ACM RecSys*.

Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han. 2017a. Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In *ACM SIGKDD*.

Chunfeng Yang, Huan Yan, Donghan Yu, Yong Li, and Dah Ming Chiu. 2017b. Multi-site user behavior modeling and its application in video recommendation. In *ACM SIGIR*.

D. Yang, B. Qu, and P. Cudré. 2019a. Privacy-preserving social media data publishing for personalized ranking-based recommendation. *IEEE TKDE*.

Q. Yang, Y. Liu, T. Chen, and Y. Tong. 2019b. Federated machine learning: Concept and applications. *ACM TIST*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. Darec: Deep domain adaptation for cross-domain recommendation via transferring rating patterns. In *IJCAI*.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *ACM SIGKDD*.

Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.

X. Zhao, Y. Guo, Y. He, H. Jiang, et al. 2014. We know what you want to buy: a demographic-based system for product recommendation on microblogs. In *ACM SIGKDD*.

G. Zhou, X. Zhu, C. Song, Y. Fan, et al. 2018. Deep interest network for click-through rate prediction. In *ACM SIGKDD*.