Novel Collaborative Filtering Recommender Friendly to Privacy Protection

Jun Wang[1*], **Qiang Tang**[2], **Afonso Arriaga**[3] and **Peter Y. A. Ryan**[1]
[1]Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg
[2]Luxembourg Institute of Science and Technology
[3]INCERT GIE, Luxembourg
junwang.lu@gmail.com, qiang.tang@list.lu, afonso.arriaga@gmail.com, peter.ryan@uni.lu

Abstract

Nowadays, recommender system is an indispensable tool in many information services, and a large number of algorithms have been designed and implemented. However, fed with very large datasets, state-of-the-art recommendation algorithms often face an efficiency bottleneck, i.e., it takes huge amount of computing resources to train a recommendation model. In order to satisfy the needs of privacy-savvy users who do not want to disclose their information to the service provider, the complexity of most existing solutions becomes prohibitive. As such, it is an interesting research question to design simple and efficient recommendation algorithms that achieve reasonable accuracy and facilitate privacy protection at the same time.

In this paper, we propose an efficient recommendation algorithm, named CryptoRec, which has two nice properties: (1) can estimate a new user's preferences by directly using a model pre-learned from an expert dataset, and the new user's data is not required to train the model; (2) can compute recommendations with only addition and multiplication operations. As to the evaluation, we first test the recommendation accuracy on three real-world datasets and show that CryptoRec is competitive with state-of-the-art recommenders. Then, we evaluate the performance of the privacy-preserving variants of CryptoRec and show that predictions can be computed in seconds on a PC. In contrast, existing solutions will need tens or hundreds of hours on more powerful computers.

1 Introduction

Recommender system is one of the most frequently used machine learning technologies in many different applications. Both academia and industry have spent a lot of efforts in designing and implementing new recommendation algorithms. These efforts have largely focused on improving the recommendation accuracy, while some efforts have also been dedicated to other useful properties such as explanability. Today, many state-of-the-art recommendation algorithms are very sophisticated, due to the aim for higher accuracy. Fed with a very large dataset, most of them face an efficiency bottleneck, i.e., it takes huge amount of computing resources to train a recommender model. We argue that, although there is a proliferation of recommendation algorithms including deep learning based ones [Zhang *et al.*, 2017], designing simple and efficient recommendation algorithms with good accuracy performance is still an interesting research question.

In parallel, with the ever-increasing personal data abuse and new privacy regulations, privacy issues in recommender systems has become a hot topic, even though the issues have been mentioned many years ago, e.g., [Ramakrishnan *et al.*, 2001]. So far, a number of privacy-preserving recommender solutions have been proposed. Differential-privacy based solutions (e.g., [McSherry and Ilya, 2009; Berlioz *et al.*, 2015]) often require the users to trust the service provider and downgrade the accuracy. Cryptographic solutions can allow users to protect their data without affecting accuracy, but they are often too complex to be practical. Representative cryptographic solutions, such as Nikolaenko et al. [Nikolaenko *et al.*, 2013] and Kim et al. [Kim *et al.*, 2016], introduce third-party crypto service providers to improve efficiency, nevertheless their performance is still far from practical and moreover the third party is difficult to be instantiated in practice. From the privacy perspective, it is also an interesting research question to design simple and efficient recommendation algorithms that facilitate privacy protection.

1.1 Our Contribution

In this paper, we propose a new recommender, named CryptoRec, which aims to facilitate privacy protection while still achieving state-of-the-art accuracy. CryptoRec decouples user features from the model parameter space, allowing a service provider to estimate a new user's preferences by directly using a model pre-learned from an expert dataset (e.g., the massive user data collected from the internet or a company's business can be used to construct such a dataset). It only relies on additions and multiplications, simplifying the execution on encrypted data. Inspired by [Han *et al.*, 2015], we propose a new model compression strategy, named Sparse-Quantization-Reuse, to reduce response latency without affecting accuracy. We evaluate the accuracy of CryptoRec with three real-world datasets and show that it is competitive

*Contact Author

with state-of-the-art recommenders. In addition, we demonstrate CryptoRec also has a nice transferability property.

Different from existing model-based algorithms, in CryptoRec, the new user's data is not required in the recommendation model training process. This is a crucial property for efficient privacy protection, because the complexity of existing solutions such as [Nikolaenko *et al.*, 2013; Kim *et al.*, 2016] mainly comes from the requirement that the user's data need to be used to train the model! We introduce two privacy-preserving protocols to compute predictions for privacy-savvy users with CryptoRec: the first one uses a pre-learned CryptoRec model to compute recommendations directly; the second one allows us to fine-tune the model parameters based on privacy-savvy user's data to further improve accuracy. Our results show that our first protocol (i.e., without fine-tuning) allows securely computing predictions of thousands of items within a few seconds on a single PC, significantly outperforming existing solutions which are also far more inefficient than the second protocol.

2 Preliminaries

In this section, we introduce collaborative filtering based recommenders and our major privacy protection building block, namely homomorphic encryption.

2.1 Collaborative Filtering (CF)

Collaborative filtering (CF) is a state-of-the-art technique for building recommender systems [Zhang *et al.*, 2017]. Below, we briefly review three representative types of CF algorithms.

Neighborhood-Based Method (NBM). The neighborhood based method estimates a user's rating on a targeted item by taking the weighted average of a certain number of ratings of the user or of the item. A typical item-based NBM (I-NBM) is defined as $\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in \mathcal{N}_u(i)} s_{ij}(r_{uj} - \bar{r}_j)}{\sum_{j \in \mathcal{N}_u(i)} |s_{ij}|}$, where $\bar{r}_i$ is the mean rating of item i, r_{uj} is the rating user u gave to item j. $s_{ij} \in \mathbf{S}^{m \times m}$ is the similarity between item i and j, m is the number of items, $\mathcal{N}_u(i)$ denotes a set of items rated by user u that are the most similar to item i. Cosine and Pearson correlation are widely used similarity metrics. User-based NBM is the symmetric counterpart of I-NBM, while I-NBM is more accurate [Su and Khoshgoftaar, 2009].

Matrix Factorization (MF). Let $\mathbf{R}^{n \times m}$ be a sparse rating matrix formed by n users and m items, in which each user rated a small number of the m items, and the missing values are marked with zero. Matrix factorization (MF) decomposes the rating matrix $\mathbf{R}$ into two low-rank and dense feature matrices [Koren *et al.*, 2009]: $\mathbf{R} \approx \mathbf{P}\mathbf{Q}^T$, where $\mathbf{P} \in \mathbb{R}^{n \times d}$ is the user feature space, $\mathbf{Q} \in \mathbb{R}^{m \times d}$ is the item feature space and $d \in \mathbb{N}^+$ is the dimension of user and item features. To predict how user u would rate item i, we compute $\hat{r}_{ui} = \mathbf{p}_u \mathbf{q}_i^T$, where $\mathbf{p}_u^{1 \times d} \subset \mathbf{P}$ and $\mathbf{q}_i^{1 \times d} \subset \mathbf{Q}$ denote the learned features vectors of user u and item i, respectively. A standard way of optimizing $\mathbf{P}$ and $\mathbf{Q}$ is to minimize the regularized squared objective function using only observed ratings ($\{r_{ui} > 0\}_{(u,i) \in \mathbf{R}}$)

$$\min_{\mathbf{P}, \mathbf{Q}} \sum_{(u,i) \in \mathbf{R}} (\mathbf{p}_u \mathbf{q}_i^T - r_{ui})^2 + \lambda(||\mathbf{p}_u||^2 + ||\mathbf{q}_i||^2) \quad (1)$$

by using the stochastic gradient descent method (SGD) [Koren *et al.*, 2009].The constant λ is a regularization factor.

Neural Network Approach. AutoRec [Sedhain *et al.*, 2015] is a notable example of neural network based recommenders, built on top of Autoencoders. Item-based AutoRec (I-AutoRec) reconstructs the inputs $\mathbf{r}_i$ by computing $\hat{\mathbf{r}}_i = f(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r}_i + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)})$, where $g(\cdot)$ and $f(\cdot)$ are activation functions, e.g., the Sigmoid function ($\frac{1}{1+e^{-x}}$). Nonlinear activation functions are crucial to the success of neural networks. Model parameters $\mathbf{W} \in \mathbb{R}^{n \times d}$, $\mathbf{V} \in \mathbb{R}^{d \times n}$, $\mathbf{b}^{(1)} \in \mathbb{R}^{d \times 1}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^{n \times 1}$ are learned by minimizing the regularized square objective function

$$\min_{\mathbf{W}, \mathbf{V}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}} \sum_{i \in R} ||\hat{\mathbf{r}}_i - \mathbf{r}_i||^2 + \lambda(||\mathbf{W}||^2 + ||\mathbf{V}||^2) \quad (2)$$

where using only observed ratings. The user-based AutoRec (U-AutoRec) is defined symmetrically in the obvious way, which is not as accurate as I-AutoRec [Sedhain *et al.*, 2015].

2.2 Homomorphic Encryption (HE)

Homomorphic encryption (HE) is a form of encryption that allows computations to be carried over ciphertexts without decryption. The result, after decryption, is the same as if the operations had been performed on the plaintexts [Gentry, 2009]. Consider two plaintexts x_1 and x_2 and their corresponding ciphertexts $[\![x_1]\!] \leftarrow \mathsf{HE.Enc}(x_1, \mathsf{pk})$ and $[\![x_2]\!] \leftarrow \mathsf{HE.Enc}(x_2, \mathsf{pk})$. An encryption scheme is additively homomorphic if it satisfies $x_1 + x_2 = \mathsf{HE.Dec}([\![x_1]\!] \oplus [\![x_2]\!], \mathsf{sk})$ or multiplicatively homomorphic if we have $x_1 \times x_2 = \mathsf{HE.Dec}([\![x_1]\!] \otimes [\![x_2]\!], \mathsf{sk})$, where $\oplus$ and $\otimes$ represent the homomorphic additive and multiplicative operators, respectively. In addition to the homomorphic properties of ciphertexts, HE schemes also allow additions ($\oplus$) and multiplications ($\odot$) between a ciphertext and a plaintext, i.e., $x_1 + x_2 = \mathsf{HE.Dec}([\![x_1]\!] \oplus x_2, \mathsf{sk})$ and $x_1 \times x_2 = \mathsf{HE.Dec}([\![x_1]\!] \odot x_2, \mathsf{sk})$.

3 Encryption-aware CryptoRec Recommender

In this section, we first present the new CryptoRec recommender which is HE-friendly; then we introduce a Sparse-Quantization-Reuse method for further improving efficiency; lastly, we describe the transferability property of CryptoRec.

3.1 Description of CryptoRec Recommender

In CryptoRec, a user is profiled by items that the user has consumed and, likewise, an item is essentially identified by the ratings received from users. Therefore, it is possible to learn item features from massive user data collected by the server. Correspondingly, we can also construct user features by aggregating the pre-learned item features according to a user's ratings history. Suppose the server has already learned the item features $\mathbf{Q} = \{\mathbf{q}_i\}_{i=1}^m$ from its database, we define the user features as follows,

$$\mathbf{p}_u = ((\mathbf{r}_u - \bar{r}_u) \cdot \boldsymbol{\phi}_u)\mathbf{Q} \quad (3)$$

where $\boldsymbol{\phi}_u = \{\phi_{ui}\}_{i=1}^m$ and $(\mathbf{r}_u - \bar{r}_u) \cdot \boldsymbol{\phi}_u$ denotes $\{(r_{ui} - \bar{r}_u)\phi_{ui}\}_{i=1}^m$. If user u rated item i, then $\phi_{ui} = 1$, otherwise,

we let $\phi_{ui} = 0$ and $r_{ui} = 0$. To approximate an observed rating r_{ui}, we follow matrix factorization [Koren *et al.*, 2009], performing a dot production between user and item features,

$$r_{ui} \approx \hat{r}_{ui} = \underbrace{(((\mathbf{r}_u - \bar{r}_u) \cdot \boldsymbol{\phi}_u)\mathbf{Q})}_{\mathbf{p}_u} \mathbf{q}_i^T \tag{4}$$

Real-world datasets often exhibit large systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others. This fact may prevent the model (Eq. (4)) from precisely modeling user preferences. A common solution is to model the biases of individual users or items, subjecting only the true user-item interactions to the factor machine [Koren *et al.*, 2009]. Follow this, we separate the user preference estimation into the biases approximator and user-item interaction approximator,

$$r_{ui} \approx \hat{r}_{ui} = \underbrace{\mu + b_u + b_i}_{biases} + \underbrace{(((\mathbf{r}_u - \bar{r}_u) \cdot \boldsymbol{\phi}_u)\mathbf{Q})\mathbf{q}_i^T}_{interaction} \tag{5}$$

where μ is the global rating average. $b_u = \bar{r}_u - \mu$ and $b_i = \bar{r}_i - \mu$ approximate user and item individual biases, respectively. As such, only the true user-item interaction is modeled by the factor machine (i.e., Eq.(4)).

The accuracy of user-item interaction modeling only relies on the item feature $\mathbf{Q}$, which may limit the model expressiveness. So that we relax the item features which were used to construct user features $\mathbf{P}$, and redefine Eq. (5) as,

$$r_{ui} \approx \hat{r}_{ui} = \underbrace{\mu + b_u + b_i}_{biases} + \underbrace{(((\mathbf{r}_u - \bar{r}_u) \cdot \boldsymbol{\phi}_u)\mathbf{A})\mathbf{q}_i^T}_{interaction} \tag{6}$$

Note that $\mathbf{A} \in \mathbb{R}^{m \times d}$ is a new item feature space, which will be optimized independently.

Model Training. The model $\Theta = \{\mathbf{A}, \mathbf{Q}\}$ can be learned by minimizing the regularized square objective function,

$$\mathcal{L} = \sum_{u=1}^{n} ||(\hat{\mathbf{r}}_u - \mathbf{r}_u) \cdot \boldsymbol{\phi}_u||^2 + \lambda \cdot (\|\mathbf{A}\|^2 + \|\mathbf{Q}\|^2) \tag{7}$$

Using the back-propagation method, we have the gradient of each model parameter of CryptoRec as follows,

$$\begin{aligned} \Delta\mathbf{A} &= \frac{\partial \mathcal{L}}{\partial \mathbf{p}_u} \cdot \frac{\partial \mathbf{p}_u}{\partial \mathbf{A}} = ((\mathbf{e}_u \cdot \boldsymbol{\phi}_u)\mathbf{Q}) \circledast \mathbf{r}_u^T + \lambda \cdot \mathbf{A} \\ \Delta\mathbf{q}_i &= \frac{\partial \mathcal{L}}{\partial \mathbf{q}_i} = \boldsymbol{\phi}_{ui} \cdot (e_{ui} \cdot ((\mathbf{r}_u - \bar{r}_u)\mathbf{A}) + \lambda \cdot \mathbf{q}_i \end{aligned} \tag{8}$$

where $e_{ui} = \hat{r}_{ui} - r_{ui}$, $\mathbf{e}_u = \{e_{ui}\}_{i=1}^m$, $\mathbf{e}_i = \{e_{ui}\}_{u=1}^n$, and $\mathbf{e}_u \cdot \boldsymbol{\phi}_u = \{e_{ui} \cdot \phi_{ui}\}_{i=1}^m$. $\circledast$ denotes outer product. In the training phase, we compute the gradient for each randomly-divided batch. Algorithm 1 outlines the training procedure.

Prediction Computation. Suppose a server has trained the parameters $\{\mathbf{A}, \mathbf{Q}\}$ on its dataset, then it can predict a new client ratings for the unrated items by Eq. (6). We stress that the client's rating vector is not necessarily involved in training the model. This is different from existing collaborative filtering approaches, e.g., those from Section 2.1. We refer this to be the fast-mode prediction, denoted as **CryptoRec-f**.

Even **CryptoRec-f** can provide very good accuracy, we can still fine-tune the pre-learned parameters $\Theta = \{\mathbf{A}, \mathbf{Q}\}$ with a new client v's rating vector to achieve better recommendation accuracy for this client. The fine-tuning process is identical to Algorithm 1, where the inputs become $\mathbf{r}_v$, $\boldsymbol{\phi}_v$,$\bar{\mathbf{r}}_v$ and $\Theta = \{\mathbf{A}, \mathbf{Q}\}$. Note that $\mathbf{r}_v$, $\boldsymbol{\phi}_v$,$\bar{\mathbf{r}}_v$ are possessed by the client v and Θ is possessed by the server. We refer this to be the accurate-mode prediction, denoted as **CryptoRec-a**.

Algorithm 1 CryptoRec training procedure $\mathcal{T}$

Input: $\mathbf{R}$, $\boldsymbol{\Phi} = \{\boldsymbol{\phi}_u\}$,$\bar{\mathbf{r}}_u = \{\bar{r}_u\}$, $\Theta = \{\mathbf{A}^{(0)}, \mathbf{Q}^{(0)}\}$
Output: Optimized $\Theta = \{\mathbf{A}^{(K)}, \mathbf{Q}^{(K)}\}$

procedure $\mathcal{T}(\{\mathbf{R}, \boldsymbol{\Phi}, \bar{\mathbf{r}}_u\}, \Theta)$
  for $k \leftarrow \{1, 2, \cdots, K\}$ **do**
    $\mathbf{A}^{(k)} \leftarrow \mathbf{A}^{(k-1)} - \eta \cdot \Delta\mathbf{A}^{(k-1)}$ ▷ η: learning rate
    $\mathbf{Q}^{(k)} \leftarrow \mathbf{Q}^{(k-1)} - \eta \cdot \Delta\mathbf{Q}^{(k-1)}$
  return $\Theta = \{\mathbf{A}^{(K)}, \mathbf{Q}^{(K)}\}$

3.2 Sparse-Quantization-Reuse Method

When we apply homomorphic encryption (HE) to protect privacy in computing predictions, shown in Section 5, the multiplications in the user-item interaction estimation (i.e., $((\mathbf{r}_u - \bar{r}_u)\mathbf{A})\mathbf{Q}^T$) will dominate the runtime.

To minimize the number of multiplications on encrypted data, we introduce a sparse-quantization-reuse method. We first sparsify a pre-learned CryptoRec model $\Theta = \{\mathbf{A}, \mathbf{Q}\}$ by removing parameters which don't contribute to final predictions, without losing accuracy. Specifically, we remove all weights whose absolute values ($\theta \in \Theta$) fall below a threshold (e.g., $|\theta| < 1.0 \times 10^{-3}$). Next, we quantify the weights to enforce more weights to share the same values. Following the method in [Han *et al.*, 2015], we classify each feature matrix $\mathbf{Q}$ and $\mathbf{A}$ into 512 clusters (i.e., each feature contains 512 shared weights), without affecting the accuracy. Lastly, we reuse the shared multiplicative results if possible, illustrated in Figure 1.

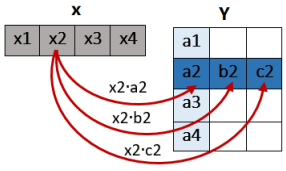

Figure 1: Reuse: for a vector-matrix multiplication between $\mathbf{x}$ and $\mathbf{Y}$, if $c_2 = a_2$, reusing the result of x_2a_2 when computing x_2c_2.

In Section 4.2, we will show this method can greatly improve the efficiency in privacy-preserving protocols. Besides facilitating privacy protection, together with data compression [Van Leeuwen, 1976], the sparse-quantization step can reduce the model size significantly. This will facilitate the deployment of recommendation services on mobile devices subjecting to limited network bandwidth and storage resources.

3.3 Transferability Property of CryptoRec

In recommender systems, transferring the knowledge learned from a source domain to a target domain to improve accuracy

and mitigate the notorious cold start problem is a challenging task [Cantador *et al.*, 2015]. CryptoRec naturally achieves transferability if two datasets share the same item entries, improving both efficiency and accuracy (Section 4.3).

4 Evaluation of CryptoRec Recommender

We test CryptoRec on three movie rating datasets (Table 1): netflix [Netflix, 2010], ml1m [Grouplens, 2010], yahoo [Yahoo!, 2010]. The testbed is a PC with 8 Intel Xeon(R) processors (3.5 GHz, 16 GB RAM), using Ubuntu 16.04.

	user #	item #	density	scale
netflix	11,000	4,768	2.17%	[1,5]
ml1m	6,040	3,952	4.2%	[1,5]
yahoo	7,637	3,791	0.72%	[1,5]

Table 1: Datasets used for benchmarking

The baseline models include item-based NBM (I-NBM) [Desrosiers and Karypis, 2011], matrix factorization (BiasedMF) [Koren *et al.*, 2009], user-based AutoRec (U-AutoRec) and item-based AutoRec (I-AutoRec) [Sedhain *et al.*, 2015]. To train the CryptoRec model, we choose $\eta = 0.0002$, $\lambda = 0.00002$ and the dimension $d = 500$. To train the baseline models, we perform a grid search around the suggested settings given in their original papers. For each dataset, we randomly split all the users into a training set (80%) and a validation set (20%), then we randomly divide each user vector of the validation set into a feeding set (90%, simulating the client's data) and a testing set (10%, denote as $\mathcal{D}$). For all the models, we repeat the accuracy experiments five times on each dataset. The root mean square error (RMSE) is adopted as the accuracy metric, $RMSE = \sqrt{\frac{\sum_{(u,i)\in\mathcal{D}}(\hat{r}_{ui}-r_{ui})^2}{|\mathcal{D}|}}$.

4.1 Accuracy Evaluation

With respect to the accurate-mode algorithm **CryptoRec-a**, we observed that the first several iterations contribute the most to the accuracy increase. Particularly, the accuracy is asymptotically close to the best performance within 3 to 5 iterations. This means that we can adopt an early-stop strategy to halt the fine-tuning, when applying privacy protection mechanism (i.e., HE) in which case the outputs cannot be monitored by the server to determine convergence. In the following discussions, **CryptoRec-a**(1) means fine-tuning in one iteration and **CryptoRec-a**(n) represents fine-tuning until convergence. Figure 2 shows the accuracy loss (i.e., RMSE increase), where I-AutoRec serves as the benchmark. Compared to the best accuracy (i.e., I-AutoRec), CryptoRec loses accuracy at most 4.6% and 1.8% on the fast-mode and accurate-mode, respectively. All baseline models are trained with all the user data from scratch, while **CryptoRec-f** is only based on a pre-trained model (the client's data is excluded from the training process), and **CryptoRec-a** is fine-tuned with only the client's data.

4.2 Sparse-Quantization-Reuse Evaluation

We first prune the parameters (i.e., $\mathbf{A}$, $\mathbf{Q}$) of which the weights are in the range of $(-5.0\times10^{-4}, 5.0\times10^{-4})$, without affecting the accuracy. Then, we use k-means clustering (k=512, in this paper) to identify the shared values of each feature, so that all the parameters that fall into the same cluster share the same value. To reduce the latency, the server should perform the two steps offline. Lastly, we reuse the shared multiplication results when computing $\mathbf{p}_u \leftarrow \mathbf{r}_u\mathbf{A}$ and $\hat{\mathbf{r}}_u \leftarrow \mathbf{p}_u\mathbf{Q}^T$ in prediction computation.

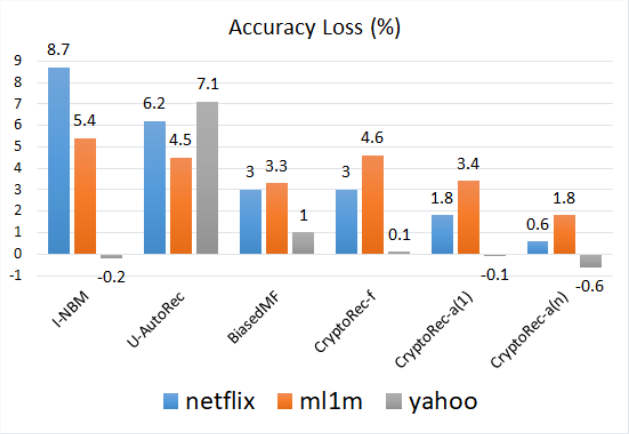

Figure 2: Accuracy loss (%). Benchmark: I-AutoRec

	netflix	ml1m	yahoo
pruning ratio	7.1%	9.2%	29.4%
reuse ratio	90.7%	90.5%	91.5%

Table 2: Pruning ratio and reuse ratio of CryptoRec

As shown in Table 2, the server can remove more than 90% of computations, resulting in a $10\times$ better efficiency. The pruning ratio is defined as $\frac{\#\ of\ pruned\ parameter}{\#\ of\ all\ the\ parameter}$. The reuse ratio is defined as $1 - \frac{\sum_i shared_i}{\sum_i all_i}$, where $shared_i$ (resp. all_i) indicates the number of shared values (resp. all the non-zero values) at i-th row of a feature matrix.

4.3 Transferability Evaluation

To evaluate the transferability of CryptoRec, we constructed two datasets pairs (ml_1 (from ml1m) and yah (from yahoo); ml_2 (from ml1m) and net (from netflix)), and each pair share the same item entries. Details are summarized in Table 3.

item intersection	data set	user#	item#	density
ml1m ⋂ yahoo	ml_1	6040	2715	5.6%
	yah	5507	2715	0.5%
ml1m ⋂ netflix	ml_2	6040	2718	5.1%
	net	10000	2718	9.2%

Table 3: New datasets resulting from the intersections other datasets

For knowledge transferring, we use the model trained on the source (src) dataset to initialize the training on the target (tgt) dataset. As shown in Table 4, the transferred knowledge improves both the accuracy and efficiency (reducing training iterations (iter#)). Especially, when transferring the knowledge learned from a relatively dense dataset to a sparse dataset, the improvement is more significant.

$src : tgt$	no transfer		transfer	
	RMSE	iter#	RMSE	iter#
$ml_1 : yah$	0.867±0.024	122	0.831±0.016	22
$yah : ml_1$	0.846±0.008	120	0.841±0.011	31
$net : ml_2$	0.853±0.003	118	0.842±0.004	15
$ml_2 : net$	0.801±0.004	95	0.791±0.005	18

Table 4: No transfer (resp. transfer): the accuracy on tgt w/o (resp. w/) knowledge transferred from src. iter#: training iteration#

5 Privacy-preserving CryptoRec Protocols

In this section, for convenience, we simplify the definition of CryptoRec (Eq. (6)) as $\hat{r}_{ui} = \mu + b_u + b_i + (\mathbf{r}_u\mathbf{A})\mathbf{q}_i^T$, and denote the client as v. In all protocols, the client generates a public/private homomorphic encryption (HE) key pair and shares the public key with the server.

5.1 Fast-Mode Prediction Protocol

As shown in Figure 3, the protocol runs in three stages. First, the client v sends her encrypted rating vector $[\![\mathbf{r}_v]\!]$ and $[\![\bar{r}_v]\!]$ to the server; Then, the server executes the prediction process $\mathcal{P}$ (Algorithm 2 which converts Eq. (6) into an HE domain) and returns the encrypted results $\hat{\mathbf{r}}_v$. Lastly, the client post-processes the predictions after decryption, such as clamping the predictions into a proper range or selecting favorite items.

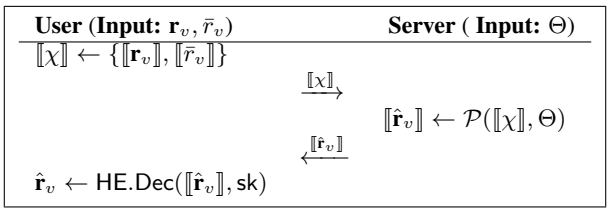

Figure 3: Privacy-preserving Fast-Mode Protocol

Algorithm 2 CryptoRec prediction procedure $\mathcal{P}$

Input: ratings $[\![\mathbf{r}_v]\!], [\![\bar{r}_v]\!]$; $\Theta = \{\mathbf{A}, \mathbf{Q}, \mu, \bar{\mathbf{r}}_i\}$
Output: predictions $[\![\hat{\mathbf{r}}_v]\!]$

1: **procedure** $\mathcal{P}(\{[\![\mathbf{r}_v]\!], [\![\bar{r}_v]\!]\}, \Theta)$
2: $\quad[\![\mathbf{p}_v]\!] \leftarrow [\![\mathbf{r}_v]\!]\mathbf{A}$ ▷ HE dot-product using $\odot$ and $\oplus$
3: $\quad$**for** $i \leftarrow [1, 2, \cdots, m]$ **do**
4: $\quad\quad[\![\hat{r}_{vi}]\!] \leftarrow b_i \oplus [\![\bar{r}_v]\!] \oplus [\![\mathbf{p}_v]\!]\mathbf{q}_i^T$ ▷ $b_i = \bar{r}_i - \mu$
5: $\quad\quad[\![\hat{\mathbf{r}}_v]\!][i] \leftarrow [\![\hat{r}_{vi}]\!]$
6: $\quad$**return** $[\![\hat{\mathbf{r}}_v]\!]$

For the sake of succinctness, we do not explicitly describe the sparse-quantization-reuse in Algorithm 2 (at line 2, 4).

5.2 Accurate-Mode Prediction Protocol (full)

In this protocol, shown in Figure 4, the server needs to fine-tune the pre-learned model (using the client's encrypted data) before computing predictions. The fine-tuning process is identical to the regular training process ($\mathcal{T}$, Algorithm 1). Since the training process only contains additions and multiplications, we can also convert it into an HE space easily. Different from the fast-mode protocol, the client v also has to send an encrypted indication vector $[\![\boldsymbol{\phi}_v]\!]$ to the server, completing the objective function computation. In contrast to the fast-mode protocol in Algorithm 2, after the fine-tuning process, the model Θ becomes encrypted. So the related operations in $\mathcal{P}$ (Algorithm 2) should be updated to their corresponding homomorphic operations.

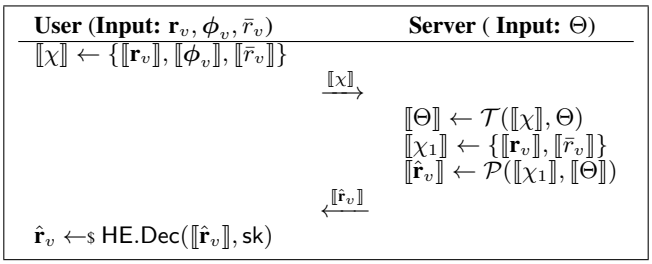

Figure 4: Privacy-preserving Accurate-Mode Protocol

5.3 Accurate-Mode Prediction Protocol (simp)

Performing the fine-tuning process on encrypted data is challenging, due to the prohibitive cost in ciphertext-only computation and corresponding storage, see the Section 5.4. Though it is not infeasible for a commercial server, it is too expensive to respond to a single query while the accuracy improvement is limited. Fortunately, as shown in Figure 2, the first fine-tuning iteration moves a big step towards the convergence. Combining together this fact and software engineering, we can achieve better accuracy with a modest increase of computation cost. Briefly, in the forward pass of the first iteration, we can apply the sparse-quantization-reuse method from Section 3.2 to accelerate the computation. In the backward pass, we immediately release the model parameters which will not be used in the future. Algorithm 3 details the first-iteration fine-tuning and prediction computation process.

Algorithm 3 CryptoRec-a(1): one-iteration fine-tuning

1: **procedure** FINE-TUNING($[\![\mathbf{r}_u]\!], [\![\boldsymbol{\phi}_u]\!], \mathbf{A}^{(0)}, \mathbf{Q}^{(0)}, \lambda, \eta$)
2: $\quad[\![\mathbf{y}_u]\!] \leftarrow [\![\mathbf{r}_u]\!]\mathbf{A}^{(0)}$
3: $\quad[\![\mathbf{e}_u]\!] \leftarrow [\![\mathbf{y}_u]\!]\mathbf{Q}^{(0)} \ominus [\![\mathbf{r}_u]\!]$ ▷ $\ominus = -\oplus$
4: $\quad[\![\mathbf{x}_u]\!] \leftarrow ([\![\mathbf{e}_u]\!] \otimes [\![\boldsymbol{\phi}_u]\!])\mathbf{Q}^{(0)}$
5: $\quad$**for** $j \leftarrow \{1, 2, \cdots, d\}$ **do**
6: $\quad\quad[\![\Delta\mathbf{A}_{:j}]\!] \leftarrow ([\![\mathbf{x}_u]\!][j] \otimes [\![\mathbf{r}_u^T]\!]) \oplus \lambda \cdot \mathbf{A}_{:j}^{(0)}$ ▷ gradient
7: $\quad\quad[\![\mathbf{A}_{:j}]\!] \leftarrow \mathbf{A}_{:j}^{(0)} \ominus (\eta \odot [\![\Delta\mathbf{A}_{:j}]\!])$ ▷ updates $\mathbf{A}_{:j}$
8: $\quad\quad[\![\mathbf{p}_u]\!][j] \leftarrow [\![\mathbf{r}_u]\!][\![\mathbf{A}_{:j}]\!]$ ▷ computes user features
9: $\quad\quad$release $[\![\mathbf{A}_{:j}]\!]$, $[\![\mathbf{x}_u]\!][j]$
10: $\quad$**for** $i \leftarrow \{1, 2, \cdots, m\}$ **do**
11: $\quad\quad[\![\Delta\mathbf{q}_i]\!] \leftarrow [\![\boldsymbol{\phi}_u]\!][i] \otimes (([\![\mathbf{e}_u]\!][i] \otimes [\![\mathbf{y}_u]\!]) \oplus \lambda \cdot \mathbf{q}_i^{(0)})$
12: $\quad\quad[\![\mathbf{q}_i]\!] \leftarrow \mathbf{q}_i^{(0)} \ominus (\eta \odot [\![\Delta\mathbf{q}_i]\!])$ ▷ updates $\mathbf{q}_i$
13: $\quad\quad[\![\hat{\mathbf{r}}_u]\!][i] \leftarrow [\![\mathbf{p}_u]\!][\![\mathbf{q}_i]\!]$ ▷ prediction $\hat{r}_{ui}$
14: $\quad\quad$release $[\![\mathbf{q}_i]\!]$, $[\![\mathbf{e}_u]\!][i]$, $[\![\boldsymbol{\phi}_u]\!][i]$
15: $\quad$**return** $[\![\hat{\mathbf{r}}_u]\!]$

	$\oplus$	$\otimes$	$\odot$	div	Sigmoid	sqrt
I-NBM	$\mathcal{O}(nm^2)$	$\mathcal{O}(nm^2)$	$\mathcal{O}(nm^2)$	$\mathcal{O}(nm^2)$	$\varnothing$	$\mathcal{O}(nm)$
U-AutoRec	$\mathcal{O}(Kmd)$	$\mathcal{O}(Kmd)$	$\mathcal{O}(Kzd+md)$	$\varnothing$	$\mathcal{O}(Kmd)$	$\varnothing$
I-AutoRec	$\mathcal{O}(K(mn+N)d)$	$\mathcal{O}(K(m+N)d)$	$\mathcal{O}(Kmzd)$	$\varnothing$	$\mathcal{O}(Knd)$	$\varnothing$
BiasedMF	$\mathcal{O}(Kmnd)$	$\mathcal{O}(K(m+N)d)$	$\mathcal{O}(md)$	$\varnothing$	$\varnothing$	$\varnothing$
CryptoRec-f	$\mathcal{O}(md)$	$\varnothing$	$\mathcal{O}(md)$	$\varnothing$	$\varnothing$	$\varnothing$
CryptoRec-a	$\mathcal{O}(Kmd)$	$\mathcal{O}(Kmd)$	$\mathcal{O}(Kzd+md)$	$\varnothing$	$\varnothing$	$\varnothing$

Table 5: Fine-tuning complexity. K: Training iterations. N: # of observed ratings. z: Rating scale

5.4 Asymptotic Performance Analysis

The complexity of each HE operation type is different. Generally, $\otimes$ is higher than $\odot$, which in turn higher than $\oplus$. Non-linear operations are not compatible with HE schemes, and incur much higher complexity. I-NBM, I-AutoRec and U-AutoRec contain non-linear operations. BiasedMF has to be trained on the whole dataset. In contrast, **CryptoRec-f** needs only $\odot, \oplus$ and **CryptoRec-a** only relies on the client's data, resulting in a significant advantage in efficiency. Detailed complexity information appear in Table 5.

5.5 Experimental Performance Analysis

We implement the fast-mode prediction protocol (which needs $\oplus, \odot$) with an efficient additive HE scheme, Paillier cryptosystem [Paillier, 1999] supported by the library python-paillier [CSIRO, 2018]. The secret key size is 2048. In this setting, the size of each encrypted message is 0.5KB.

For accurate-mode prediction protocol (which needs $\oplus, \odot, \otimes$), we adopt a ring-based somewhat (fully) HE scheme, the Fan-Vercauteren scheme [Fan and Vercauteren, 2012] supported by the SEAL library [Chen *et al.*, 2017]. We let the polynomial degree be 4096, the plaintext module be 65537. To encode real numbers, we reserve 1024 coefficients of the polynomial for the integral part and expand the fractional part to 16 digits of precision. For more detail on this configuration, we refer interested readers to [Chen *et al.*, 2017]. In this setting, the size of a ciphertext is 96 KB. However, the item features often contain millions of parameters. Take the **CryptoRec-a**($n > 1$) on dataset ml1m as an example, the features $[\![\mathbf{A}^{3952\times 500}]\!]$ and $[\![\mathbf{Q}^{3952\times 500}]\!]$ has around 4 million parameters, which requires 360GB RAM. In the experiment, we implemented the one-iteration fine-tuning (Algorithm 3), which only needs less than 2GB RAM.

Table 6 describes the running time and communication cost on each dataset. Compared to the fast-mode implementation, though the time cost of the accurate-mode implementation is significantly increased, the server can amortize the cost by packing multiple user data into one message similar to [Gilad-Bachrach *et al.*, 2016].

Protocol		netflix	ml1m	yahoo
Fast	Message size (MB)	4.8	3.86	3.72
	Server time cost (s)	14.2	10.9	7.3
	Client time cost (s)	7.1	5.8	5.6
Accurate	Message size (GB)	1.31	1.08	1.04
	Server time cost (h)	9.4	7.8	7.5
	Client time cost (s)	14.3	11.8	11.4

Table 6: CryptoRec: Efficiency Evaluation

As far as we know, CryptoRec is the first solution allowing online recommendation services in real-time, while preserving the privacy. Using existing recommenders (e.g., MF), before computing recommendations, the server often has to train the model with the client's encrypted data and its database. Recent advances in devising secure frameworks to execute existing recommenders come from GraphSC [Nayak *et al.*, 2015], in which a single training iteration of MF took 13 hours to run on 128 processors. On the ml1m dataset, about 20 iterations are necessary for convergence.

6 Related Work

Canny et al. [Canny, 2002] introduced a distributed solutions to factor the Singular Value Decomposition securely, based on HE schemes. To further improve the efficiency, Nayak et al. [Nayak *et al.*, 2015] brought parallelism to the execution of the oblivious version of graph-based algorithms (e.g., MF). Shmueli and Tassa [Shmueli and Tassa, 2017] presented a solution for securely learning neighborhood-based models, by assuming a semi-honest mediator which computes and distributes intermediate values to each party. Technically, these solutions can be adjusted for providing online recommendation services. But the computation latency would prevent the use of the services in the real world.

Recent solutions for privacy-preserving Machine Learning as a Service often assume the server has a pre-learned machine learning model [Gilad-Bachrach *et al.*, 2016; Liu *et al.*, 2017; Zhang *et al.*, 2018], separating the training process from machine learning services. Most efforts have been focused on efficiently computing non-linear operations. Gilad-Bachrach et al. [Gilad-Bachrach *et al.*, 2016] substituted non-linear activation functions with a square function. However, this approach often leads to a significant accuracy loss [Liu *et al.*, 2017; Rouhani *et al.*, 2017; Zhang *et al.*, 2018]. To preserve accuracy, Liu et al. [Liu *et al.*, 2017] and Rouhani et al. [Rouhani *et al.*, 2017] proposed to evaluate neural networks using secure multiparty computation schemes, but requiring the server and client to be online constantly.

7 Conclusions

In this paper, we proposed an encryption-aware recommender, CryptoRec, which can provide privacy-preserving recommendation services with high throughput, while being competitive with state-of-the-art accuracy performance.

Acknowledgments

This work is supported by the Fonds National de la Recherche, Luxembourg (project code: C13/IS/5856658).

References

[Berlioz *et al.*, 2015] Arnaud Berlioz, Friedman Arik, Ali Kaafar Mohamed, Boreli Roksana, and Berkovsky Shlomo. Applying differential privacy to matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 107–114, 2015.

[Canny, 2002] John Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57. IEEE, 2002.

[Cantador *et al.*, 2015] Iván Cantador, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. Cross-domain recommender systems. In *Recommender Systems Handbook*, pages 919–959. Springer, 2015.

[Chen *et al.*, 2017] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library-seal v2. 1. In *International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 2017.

[CSIRO, 2018] CSIRO. *python-paillier*, 2018.

[Desrosiers and Karypis, 2011] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, pages 107–144, 2011.

[Fan and Vercauteren, 2012] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[Gentry, 2009] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

[Gilad-Bachrach *et al.*, 2016] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, 2016.

[Grouplens, 2010] Grouplens. *Grouplens Moivelens*, 2010.

[Han *et al.*, 2015] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[Kim *et al.*, 2016] Sungwook Kim, Kim Jinsu, Koo Dongyoung, Kim Yuna, Yoon Hyunsoo, and Shin Junbum. Efficient privacy-preserving matrix factorization via fully homomorphic encryption: Extended abstract. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 617–628, 2016.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[Liu *et al.*, 2017] Jian Liu, Mika Juuti, Yao Lu, and N Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–631. ACM, 2017.

[McSherry and Ilya, 2009] Frank McSherry and Mironov Ilya. Differentially private recommender systems: building privacy into the Netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 627–636, 2009.

[Nayak *et al.*, 2015] Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. Graphsc: Parallel secure computation made easy. In *Security and Privacy, IEEE Symposium on*. IEEE, 2015.

[Netflix, 2010] Netflix. *Netflix Prize Dataset*, 2010.

[Nikolaenko *et al.*, 2013] Valeria Nikolaenko, Ioannidis Stratis, Weinsberg Udi, Joye Marc, Taft Nina, and Boneh Dan. Privacy-preserving matrix factorization. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pages 801–812, 2013.

[Paillier, 1999] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology—EUROCRYPT*. Springer, 1999.

[Ramakrishnan *et al.*, 2001] Naren Ramakrishnan, Benjamin J. Keller, Batul J. Mirza, and Ananth Y. Grama. Privacy risks in recommender systems. *Internet Computing, IEEE*, 5:54–63, 2001.

[Rouhani *et al.*, 2017] Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. Deepsecure: Scalable provably-secure deep learning. *arXiv preprint arXiv:1705.08963*, 2017.

[Sedhain *et al.*, 2015] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *In International Conference on World Wide Web*. ACM, 2015.

[Shmueli and Tassa, 2017] Erez Shmueli and Tamir Tassa. Secure multi-party protocols for item-based collaborative filtering. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 89–97. ACM, 2017.

[Su and Khoshgoftaar, 2009] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[Van Leeuwen, 1976] Jan Van Leeuwen. On the construction of huffman trees. In *ICALP*, pages 382–410, 1976.

[Yahoo!, 2010] Yahoo! *R4-Yahoo! Movies*, 2010.

[Zhang *et al.*, 2017] Shuai Zhang, Lina Yao, Aixin Sun, and Tay Yi. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017.

[Zhang *et al.*, 2018] Qiao Zhang, Wang Cong, Wu Hongyi, Xin Chunsheng, and Tran V. Phuong. Gelu-net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning. In *IJCAI*, pages 3933–3939, 2018.