Influential Exemplar Replay for Incremental Learning in Recommender Systems

Xinni Zhang¹, Yankai Chen¹, Chenhao Ma², Yixiang Fang², Irwin King¹

¹The Chinese University of Hong Kong
²The Chinese University of Hong Kong, Shenzhen
{xnzhang23, ykchen, king}@cse.cuhk.edu.hk, {machenhao, fangyixiang}@cuhk.edu.cn

Abstract

Personalized recommender systems have found widespread applications for effective information filtering. Conventional models engage in knowledge mining within the static setting to reconstruct singular historical data. Nonetheless, the dynamics of real-world environments are in a constant state of flux, rendering acquired model knowledge inadequate for accommodating emergent trends and thus leading to notable recommendation performance decline. Given the typically prohibitive cost of exhaustive model retraining, it has emerged to study incremental learning for recommender systems with ever-growing data. In this paper, we propose an effective model-agnostic framework, namely INFER. INFER facilitates recommender models in retaining the earlier assimilated knowledge, e.g., users’ enduring preferences, while concurrently accommodating evolving trends manifested in users’ new interaction behaviors. We commence with a vanilla implementation that centers on identifying the most representative data samples for effective consolidation of early knowledge. Subsequently, we propose an advanced solution, namely INFEROnce, to optimize the computational overhead associated with the vanilla implementation. Extensive experiments on four prototypical backbone models, two classic recommendation tasks, and four widely used benchmarks consistently demonstrate the effectiveness of our method as well as its compatibility for extending to several incremental recommender models.

Introduction

To tackle overwhelming information overload, recommender systems play a crucial role in online services, e.g., Web search, news recommendation, and E-commerce advertising. Modern algorithms exploit knowledge mining from similar user correlations, e.g., collaborative filtering (Rendle et al. 2012; He et al. 2017; Chen et al. 2023c) and social regularization (Ma et al. 2011), to content-based analysis, e.g., multimodal understanding (Zheng et al. 2018; Gupta et al. 2020) and auxiliary knowledge supplement (Wang et al. 2019a,b; Chen et al. 2022c,b; Zhang et al. 2022a). With the advancements of deep learning techniques, these recommender models exhibit strong capability in learning complex and latent features, demonstrating the performance superior-

Figure 1: Illustration of INFER motivations.
incremental interaction data flow is for unveiling the shifts of users’ preferences and hidden intentions. (2) Moreover, different from images that are essentially isolated objects, incremental interaction data encompass pairs of users and items. Incrementally learning such interactive data thereby affects the feature updating of bipartite parties via the mutual propagation of collaborative filtering signals for recommendation. Therefore, it has emerged to specifically develop incremental learning for recommender systems with a few recent attempts at graph neural methodologies (Xu et al. 2020; Wang, Zhang, and Coates 2021; Ahrabian et al. 2021).

While the aforementioned pioneer works rely on graph-based modeling, in this paper, we propose an effective model-agnostic incremental learning framework, namely **INfluential Exemplar Replay (INFER)**, that is compatible with more recommender backbones. Generally, our methodology is rooted in the experience replay strategy (Rebuffi et al. 2017; Zenke, Poole, and Ganguli 2017), which interleaves a small proportion of early data samples with new data for incremental model training. By doing so, the updated model gets to revisit and learn from its past experiences, preventing the loss of earlier learned knowledge, e.g., users’ long-term interests or persistent preferences. Intuitively, the core of experience replay lies in finding high-quality data samples. To provide effective and compact replay samples, we propose to retrieve influential exemplars from historical interaction data. Different from straightforward strategies (Vitter 1985; Welling 2009; Rebuffi et al. 2017; Prabhu, Torr, and Dokania 2020; Ahrabian et al. 2021) that infer important samples based on data-level properties, e.g., frequencies or diversities, as shown in Figure 1b, our distinctive influential exemplars embody the elements that directly impact the model-level performance across various recommendation tasks. Our vanilla implementation, denoted by **INFERVanilla**, is underpinned by the influence function of Robust Statistics (Koh and Liang 2017). We further propose an advanced solution, i.e., **One-step INfluential Estimation** denoted by **INFERONE**, to optimize the efficiency and instability bottlenecks of vanilla influence functions (Basu, Pope, and Feizi 2021; Epifano et al. 2023) for large recommender models. In empirical evaluation, we implement our methodology into four renowned recommender backbones. Experiments on four real-world benchmarks and two recommendation tasks consistently showcase the performance superiority of **INFERONE** with desirable generalization and efficiency. Our principal contributions are summarized as follows:

- To the best of our knowledge, we are the first to incorporate analytical influence methodology for constructing incremental recommender models, via retrieving replay exemplars directly influential to downstream tasks.
- To deal with the efficiency and instability issues of our vanilla implementation, we further propose a refined approach namely **INFERONE** with computation approximation and acceleration.
- Extensive experiments demonstrate the high adaptability of **INFERONE** across several backbone architectures. It consistently outperforms recent end-to-end incremental recommender solutions on four public benchmarks.

## Related Work

### Collaborative Filtering

Collaborative filtering (CF), as one classic paradigm of recommender algorithms, parameterizes users and items as embeddings to reconstruct historical interactions (Ying et al. 2018; Chen et al. 2022a; Zhang et al. 2023b; Chen et al. 2023b). Early CF models such as matrix factorization (Koren, Bell, and Volinsky 2009; Chen et al. 2020) usually decompose the user-item interaction matrix into low-dimensional embeddings. More recent neural models like DeepFM (Guo et al. 2017), TwoTower (Huang et al. 2013) and NCF (He et al. 2017) further leverage neural networks to improve modeling capability. Since user-item interactions can be naturally represented by a bipartite graph (Chen et al. 2023a), Graph Neural Networks (Hamilton, Ying, and Leskovec 2017; Velickovic et al. 2017; Zhang et al. 2022b, 2023c; Song, Zhang, and King 2023; Ma et al. 2023), as the neural architectures specifically for graph data (Li et al. 2023; Zhang et al. 2023a), have gained impressive development to be a favored formulation for CF algorithms, such as NGCF (Wang et al. 2019c), LightGCN (He et al. 2020), and SimGCL (Yu et al. 2022). All these approaches work in a static setting, where models are designed to be trained and deployed for the entire snapshot of interaction data. Our research deviates from this assumption to focus on the evolving data setting that imitates the real-life environment. Specifically, our goal is to develop models with the ability to continuously learn knowledge from new interaction data as it becomes available.

### Incremental Learning

Incremental learning (IL), a.k.a., continual learning, enables intelligence models with continual knowledge acquisition ability for new environment adaptation (De Lange et al. 2021; Zenke, Poole, and Ganguli 2017; Prabhu, Torr, and Dokania 2020; Rannen et al. 2017; Aljundi, Chakravarty, and Tuytelaars 2017). IL-based methods initially garner attention in Computer Vision. The main challenge is avoiding catastrophic forgetting (McCloskey and Cohen 1989), which is the tendency for models to substantially forget previously learned information when updated with new data.

There are three major types of IL methodologies. (1) **Weight regularization methods** (Rannen et al. 2017; Kirkpatrick et al. 2017; Wang, Zhang, and Coates 2021) aim to restrict the overall amount of parameter changes by designing additional regularization of loss terms. (2) **Parameter isolation methods** (Mallya and Lazebnik 2018; Hung et al. 2019; Yuan et al. 2021) allocate model capacity with specific focuses on enabling parameter sharing and isolation while also preventing knowledge leakage during the incremental data learning. (3) **Experience replay methods** (Rebuffi et al. 2017; Prabhu, Torr, and Dokania 2020; Zenke, Poole, and Ganguli 2017; Bonicelli et al. 2022; Zhou and Cao 2021) usually sample and reuse a subset of previously encountered data in model updating to avert forgetting issues. However, most of these methods are designed for image-processing tasks, while only a few recent attempts have studied the problem for recommender systems (Xu et al. 2020; Wang, Zhang,
Figure 2: Training workflow of incremental recommender models with influential exemplar replay of INFER.

and Coates 2021; Ahriban et al. 2021; He et al. 2023b; Wang et al. 2023). In this work, we focus on experience replay strategy and are motivated to propose a fundamental method that is compatible with various recommender backbone models.

**INFER Methodology**

**Preliminaries**

**Problem Formulation.** Given a user-item interaction stream list $D$ with consecutive data segments: $D = \{D_0, D_1, \cdots, D_T\}$. At each timestamp $t$, the incremental data learning for recommendation accesses segment $D_t$ for model updating, whilst maintaining previous knowledge from $\{D_0, D_1, \cdots, D_{t-1}\}$ to prevent catastrophic forgetting problem.

**Experience replay Strategy.** To memorize the early knowledge, experience replay explicitly maintains a reservoir to buffer a small subset of historical data. During incremental training, the buffered data are retrieved and integrated with the new data; then the underlying model is iteratively updated to mitigate its tendency to forget previously learned knowledge, e.g., users’ long-term preferences.

**Framework Procedure**

For each timestamp $t$, with the original data segment $D_t$, we obtain the incremental train data $D_t^*$ as $D_t^* := \mathcal{P}_{t-1} \cup D_t$ for model updating, where $\mathcal{P}_{t-1}$ denotes the reservoir constructed at previous timestamp $t-1$. As for the construction of reservoir $\mathcal{P}_t$ at time $t$, $K$ samples are filtered out from $D_t^*$, which is prepared for the next incremental training. Notably, the quality of the target data reservoir directly determines the performance of incremental recommender models, which motivates us to buffer most influential data samples, i.e., namely influential exemplars. In this work, we interpret such a notion of influence as an alteration of data occurrence that perturbs recommendation performance. Specifically, our proposed framework can be abstracted by the function $f$ as follows:

$$\mathcal{P}_t := f(\hat{\theta}_t|K, D_t^*).$$

(1)

$\hat{\theta}_t$ refers to certain optimal parameter settings of the underlying recommender system at each timestamp $t$:

$$\hat{\theta}_t := \arg \min_{\theta_t \in \Theta} \sum_{z_i \in D_t^*} L(z_i, \theta_t),$$

(2)

where $\Theta$ is the search space of model parameters, $z_i$ is a user-item interaction from the current data segment, and $L(z_i, \theta_t)$ can be any recommendation loss function at timestamp $t$ to measure the disparity against the ground-truth. Our objective of Eqn. (1) thus is to consistently find the influential exemplars that further boost the recommendation performance for the future timestamp. A high-level workflow is depicted in Figure 2.

**Vanilla Solution: Influence-based Replay**

Specifically, to implement $f(\cdot)$ in Eqn. (1) for sampling the most influential replay data, we aim to understand the dependence of recommender model predictions on a single user-item interaction. While a straightforward manner would be leave-one-out retraining (Basu, You, and Feizi 2020), which is however computationally infeasible, we incorporate influence function (Koh and Liang 2017) to avoid this inadequacy. Generally, the influence function is an important concept in Robust Statistics, which measures the effect of a change in one observation on an estimator. To fill in the reservoir at this timestamp $t$, i.e., $\mathcal{P}_t$, for any interaction candidate $z_m \in D_t^*$, our initial idea is to explicitly observe the impact of upweighting $z_m$ by some small perturbation $\epsilon$. This produces the following new parameter settings:

$$\hat{\theta}_{t,\epsilon,z_m} := \arg \min_{\delta_t \in \Theta} \left( \epsilon L(z_m, \theta_t) + \sum_{z_i \in D_t^*} L(z_i, \theta_t) \right).$$

(3)

Under the strict convexity and second-order differentiability of the loss function $L$, the influence of upweighting $z_m$ on parameters $\theta_t$, i.e., achieving $\hat{\theta}_{t,\epsilon,z_m}$ rather than $\hat{\theta}_t$, denoted by $I_{\text{params}}(z_m)$, can be quantified with the first-order Taylor’s approximation (Scanlon 1996; Koh and Liang 2017):

$$I_{\text{params}}(z_m) := \frac{d\hat{\theta}_{t,\epsilon,z_m}}{d\epsilon}
|_{\epsilon=0} = -H^{-1}_{\hat{\theta}_t} \nabla_{\hat{\theta}_t} L(z_m, \hat{\theta}_t),$$

(4)

where $\nabla$ is the first-order differential operator and $H_{\hat{\theta}_t} = \sum_{z_i \in D_t^*} \nabla_{\hat{\theta}_t}^2 L(z_i, \hat{\theta}_t)$ denotes the Hessian matrix and is positive definite by assumption.

While the original definition, i.e., Eqn. (4), measures $z_m$’s influence on the model parameters, in incremental learning for recommendation, we are however more interested in measuring $z_m$’s influence on recommendation predictions. Ideally, such influence on recommendation results should be measured as completely as possible, which can be formulated as follows:

$$I_{\text{rec-loss}}(z_m) := \sum_{z_i \in D_{0:t}} \frac{dL(z_i, \hat{\theta}_{t,\epsilon,z_m})}{d\epsilon}
|_{\epsilon=0}.$$ 

(5)

We use notation $D_{0:t}$ to denote all historical interactions. Based on the computed values $I_{\text{rec-loss}}$ from $D_{0:t}$, we hope to retrieve the most influential samples such that they make substantial impacts on all previous data. This actually implies that the retrieved samples are representative to maintain the earlier learned knowledge such as users’ long-term preferences. Noteworthily, in the incremental learning setting, the complete historical data before timestamp $t$, i.e., $D_{0:t-1}$, is actually unavailable. We thus use the empirical
risk on \( P_{t-1} \) to approximate the terms derived by the original \( D_{0:t-1} \), by presuming a close correlation between them. Hence, Eqn. (5) is rewritten as follows:

\[
I_{\text{rec-\text{loss}}}(z_m) \approx \sum_{z_i \in P_{t-1} \cup D_t} \frac{dL(z_i, \hat{\theta}_{t,e,z_m})}{de} \bigg|_{e=0} \\
= \sum_{z_i \in D^*_t} \nabla_{\hat{\theta}_t} L(z_i, \hat{\theta}_t)^\top \cdot I_{\text{params}}(z_m). \tag{6}
\]

To keep reservoir \( P \) informative, we thus pick out those interactions with the most \( K \) negative numerical values, which conversely indicates the most representative data for recommender models to capture users’ long-term preferences.

**INFERONCE: One-step Influence Estimation**

**Limitations of Vanilla Implementation.** However, the vanilla implementation may exhibit practical flaws. Our discussions are twofold: (1) Intuitively, inverting the Hessian matrix in \( I_{\text{params}} \) term of Eqn. (6) is computationally intensive, i.e., quadratic to the size of parameters \( \theta_t \) of the downstream recommender backbone. This is unacceptable as modern recommender systems typically have hundreds of millions of parameters, particularly for scenarios with limited computation resources. We detail the time complexity analysis in the later section. (2) Moreover, as we will empirically elaborate in experiments, such vanilla version may underperform for large recommender models, which is also observed by recent research (Basu, Pope, and Feizi 2021; Epifano et al. 2023).

**Fast Influence Estimation.** We now formally introduce our advanced method **One-step INFLUENCE Estimation (INFERONCE).** Recall in Eqn. (3), to algorithmically compute \( \hat{\theta}_{t,e,z_m} \), one may adopt gradient-based methods, e.g., Stochastic Gradient Descent (SGD), to conduct iterative optimization until convergence:

\[
\hat{\theta}_{t}^{(h+1)} \leftarrow \hat{\theta}_{t}^{(h)} - \eta \nabla_{\hat{\theta}_t}(\epsilon L(z_m, \hat{\theta}_t^{(h)}) + \sum_{z_i \in D^*_t} L(z_i, \hat{\theta}_t^{(h)})).
\tag{7}
\]

With a slight notation abuse, we use \( \hat{\theta}_t^{(h)} \) to specifically refer to the intermediate status of \( \theta_t \) at iteration \( h \). Here \( \eta \) denotes the learning rate. Then \( \hat{\theta}_{t,e,z_m} \) is defined as the converged value of \( \theta_t \) from Eqn. (7). Since the efficiency bottleneck of vanilla implementation lies in the exhaustive iterations, i.e., to implement Eqn. (7), we thus propose to approximate \( \hat{\theta}_{t,e,z_m} \) with only one-step estimation as follows:

\[
\hat{\theta}_{t,e,z_m} := \theta_t - \lambda \nabla_{\hat{\theta}_t}(\epsilon L(z_m, \theta_t) + \sum_{z_i \in D^*_t} L(z_i, \theta_t)). \tag{8}
\]

Which significantly reduces the computation cost compared to Eqn. (7) whilst providing flexibility of tuning approximation rate with hyper-parameter \( \lambda \). Then we derive our one-step influence estimation, denoted by \( \hat{I}_{\text{rec-\text{loss}}}(\cdot) \), to similarly approximate the vanilla one in Eqn. (6) as follows:

\[
\hat{I}_{\text{rec-\text{loss}}}(z_m) := \sum_{z_i \in D^*_t} \frac{dL(z_i, \hat{\theta}_{t,e,z_m})}{de} \bigg|_{e=0} \\
= \sum_{z_i \in D^*_t} \nabla_{\hat{\theta}_t} L(z_i, \hat{\theta}_t)^\top \cdot \nabla_{\hat{\theta}_t} L(z_m, \hat{\theta}_t). \tag{9}
\]

We observe from Eqn. (9) that, for any candidate \( z_m \) throughout the whole training data \( D^*_t \), its estimated influence \( \hat{I}_{\text{rec-\text{loss}}}(z_m) \) is determined as significant by two manners:

- either, the gradient norm of \( z_m \) at parameters \( \theta_t \), i.e., \( ||\nabla_{\theta_t} L(z_m, \theta_t)||_2 \), is as large as possible that reflects a substantial marginal impact to the downstream recommendation loss;
- or, the gradient direction of \( \nabla_{\theta_t} L(z_m, \theta_t) \) is as consistent as possible to the direction of term \( \sum_{z_i \in D^*_t} \nabla_{\theta_t} L(z_i, \theta_t) \) that regards to the recommendation losses capturing early model knowledge for all users in \( D^*_t \).

Therefore, these influential exemplars produced by our INFERONCE are also in line with our intuition. We provide a visual illustration in Figure 3.

**Approximation Discussion of INFERONCE.** During model training at each timestamp \( t \), we encourage \( \theta_t \) to approach \( \hat{\theta}_t \). Then the estimated influence in Eqn. (9) is similar to the vanilla one in Eqn. (6), only excluding the inverse Hessian matrix \( H_{\hat{\theta}_t}^{-1} \). Notice that this covariant-weighted matrix measures the inter-dependence between the training samples, as it gauges their “resistance” to other samples’ perturbations. In recommendation scenarios, such inter-dependence in underlying interaction data tends to be complex and noisy, due to the subjective nature of user behaviors and the sensitivity to external factors. As for our INFERONCE methodology, we thus omit those high-order gradients that are extremely expensive to analytically calculate or empirically observe. This provides a straightforward yet effective way to measure the influence of a single interaction on model optimization, thus expediting the convergence with less complexity. We demonstrate this later in runtime experiments of Section 11. The pseudocodes of INFERONCE are attached in Algorithm 1.

**Complexity Analysis.** The time complexity of our vanilla method INFERVANILLA in Eqn. (6) is \( O(|D^*_t|^2 + q^4) \), where \( q \) is the total number of model parameters and variables. On the contrary, the time complexity of our advanced solution INFERONCE in Eqn. (9) is \( O(|D^*_t| q) \) with \( q > |D^*_t| \).
Algorithm 1: Working Procedure of INFERONCE

\begin{algorithm}
\caption{Working Procedure of INFERONCE}
\begin{algorithmic}[1]
\State \textbf{Input:} the sequence of user-item interaction segments $D = \{D_0, D_1, \ldots, D_T\}$; influential exemplar number $K$; model parameters $\theta_{(0, \ldots, T)}$ of the underlying recommender backbone $RecSys$.
\State $\theta_0 \leftarrow$ Initialization;
\For {$t = 0$ to $T$}
\State $P_t \leftarrow \emptyset$;
\If {$t = 0$}
\State $D_0^t \leftarrow D_0$;
\Else
\State $D_t^t \leftarrow D_t \cup P_{t-1}$;
\EndIf
\State Training $RecSys$ with $D_t^t$ starting from setting $\theta_t$;
\State $\theta_t \leftarrow$ Optimal parameters of well-trained $RecSys$;
\State $P_t \leftarrow f(\theta_t, K, D_t^t)$ with INFERONCE;
\State $\theta_{t+1} \leftarrow \hat{\theta}_t$;
\EndFor
\end{algorithmic}
\end{algorithm}

Experiments

In this section, we present a series of empirical evaluations with the aim of addressing the following research questions:

• **RQ1:** How does INFERONCE plug-and-play for different backbone models and perform compared to other incremental methodologies?

• **RQ2:** How close is INFERONCE to the “ideally-optimal” case of model retraining with complete historical data?

• **RQ3:** Can INFERONCE possibly further enhance end-to-end incremental recommender models?

• **RQ4:** How does INFERONCE optimize over the vanilla implementation in the recommendation scenario?

• **RQ5:** How does the reservoir size of INFERONCE influence the model performance?

Experimental Setups

Evaluation Benchmarks. We incorporate four real-world benchmarks that vary in size, domain, sparsity, and duration, as reported in Table 1. Each dataset is partitioned chronologically into a 50% base segment and five consecutive 10% incremental segments. The base segment is randomly divided into training, validation, and testing sets in a 6:2:2 ratio. For incremental segments, to imitate real production scenarios, we retrieve recommender models trained in the previous step and assess their performance with the most recent consecutive data. The data is evenly divided into two halves, with one half used for validation and the other for testing.

Evaluation Metrics. We consider two evaluation tasks: Click-through rate (CTR) prediction and Top-$N$ recommendation. CTR prediction is evaluated using $AUC$, while Top-$N$ recommendation uses $Recall@N$. For stable reproducibility, we conduct five-fold cross validation. We report testing results averaged on all incremental updates.

Backbone Models. To demonstrate the generalization of INFERONCE, we implement it into four renowned neural recommender models: DeepFM (Guo et al. 2017) and TwoTower (Huang et al. 2013) for CTR prediction, MF-BPR (Rendle et al. 2012) and LightGCN (He et al. 2020) for Top-$N$ recommendation.

Competing Algorithms. Apart from our basic implementation, INFER Vanilla, we additionally include eight competing methods that are broadly subsumed into three groups.

• **Conventional Implementations:** (1) Fine-Tune can be considered as the lower-bound of incremental models which just fine-tunes the previously trained model solely on the incremental segment. (2) FD-Retrain entails exhaustive full-data retraining at each time of data update. It can be considered the upper-bound performance based on all the accumulated data access.

• **Experience Replay Methods:** (3) Uniform (Vitter 1985; Ahriabian et al. 2021) is a classic experience replay strategy that uniformly samples a subset of historical data as the incremental segment. (4) ER-MIR (Rahaf and Lucas 2019) predicts parameter updates to retrieve the pivotal samples. (5) InvDeg (Ahriabian et al. 2021), taking inspiration from Gdumb (Prabhu, Torr, and Dokania 2020), leverages the graph properties for sampling. We further extend it to the non-graph-based scenario. For a fair comparison, we keep the same reservoir size for replay.

• **Incremental Recommender Models:** (6) GraphSAIL (Xu et al. 2020), (7) SGCT (Wang, Zhang, and Coates 2021), and (8) LWC-KD (Wang, Zhang, and Coates 2021) are three state-of-the-art methods that based on graph neural networks. They employ miscellaneous designs such as knowledge distillation mechanisms to preserve key model knowledge that was learned from historical data.

Overall Performance (RQ1)

In this section, we present an overall performance analysis between INFERONCE and all baselines for tasks of CTR prediction and Top-$N$ recommendation. The results are shown in Table 2, and our empirical observations are twofold:

• **CTR prediction:** (1) Our method outperforms Fine-Tune implementation across all backbone models and datasets, yielding performance improvements from 0.22% to 5.12% in terms of $AUC$. This highlights the efficacy of INFERONCE in mitigating the forgetting tendency that may occur when simply fine-tuning a previously trained model. (2) In comparison to other IL methods, INFERONCE consistently exhibits superior performance on CTR prediction. The only exception is the experiment on Foursquare using DeepFM as the backbone model.

Table 1: Dataset statistics. # IR (M) represents the number of interactions in millions ($10^6$). # S (m) refers to the sparsity in thousandths ($10^{-3}$).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Lastfm-2k</th>
<th>TB2014</th>
<th>Gowalla</th>
<th>Foursquare</th>
</tr>
</thead>
<tbody>
<tr>
<td># Users</td>
<td>1,090</td>
<td>8,844</td>
<td>29,858</td>
<td>51,919</td>
</tr>
<tr>
<td># Items</td>
<td>3,646</td>
<td>39,103</td>
<td>40,988</td>
<td>37,320</td>
</tr>
<tr>
<td># IR (M)</td>
<td>0.05</td>
<td>0.75</td>
<td>1.03</td>
<td>2.35</td>
</tr>
<tr>
<td># S (m)</td>
<td>13.22</td>
<td>21.7</td>
<td>0.84</td>
<td>1.21</td>
</tr>
<tr>
<td># Time</td>
<td>36yrs</td>
<td>31yds</td>
<td>19mths</td>
<td>22mths</td>
</tr>
</tbody>
</table>
Table 2: (1) Overall performance on tasks of CTR prediction ($AUC$) and Top-$N$ recommendation ($Recall@20$) based on five-fold cross validation. (2) The best results are bold and the second-best values are underlined.

- **Top-$N$ recommendation**: (1) While INFER$_{Vanilla}$ performs well on smaller datasets, i.e., Lastfm-2k and Taobao2014, it however downgrades on larger ones, i.e., Gowalla and Foursquare. This can be attributed to the vanilla influence function tends to be sensitive to the scale of network structures or model parameters (most recommender models require explicit size allocation for both user- and item-embeddings), while the lightweight is more favorable to produce more accurate and stable performance (Basu, Pope, and Feizi 2021). On the contrary, our INFER$_{ONCE}$ performs the best for larger datasets, which reaffirms its effectiveness for Top-$N$ recommendation. (2) Moreover, compared to the latest graph-based models, i.e., GraphSAIL, SGCT, and LWC-KD, INFER$_{ONCE}$ is model-agnostic and can be adapted to the non-graph-based backbone models, e.g., BPR-MF. This demonstrates its generalization capability with consistent performance superiority. The RQ3 section empirically investigates how INFER$_{ONCE}$ further enhances these graph-based models.

INFER$_{ONCE}$ v.s. Full-data Retraining (RQ2)

We compare INFER$_{ONCE}$ with full-data retraining, which provides insights into how close our model is to the “optimal” case. We exhaustively repeat the experiments similar to the evaluation paradigm for RQ1 and plot the stacked-column charts in Figure 4. We notice that (1) for CTR prediction, INFER$_{ONCE}$ is competitive with even slightly better performance than retraining, e.g., plugged on DeepFM on Lastfm-2k, Gowalla, and Foursquare datasets. This showcases that INFER$_{ONCE}$ can well “memorize” users’ preferences to make correct click identification for recommendations. (2) While the CTR task can be interpreted as binary classification, Top-$N$ recommendation could be more difficult, as the relative item ranking personalized by each user is of the essence. Consequently, compared to model retraining with complete historical data, INFER$_{ONCE}$ is left with a discernible gap in terms of $Recall@20$ metric. However, one straightforward flaw is that full-data retraining is extremely expensive, we detail their runtime costs later.

Cross-model Compatibility Analysis (RQ3)

As we mentioned earlier, three state-of-the-art graph-based models, i.e., GraphSAIL, SGCT, and LWC-KD, incorporate the knowledge distillation mechanism to reduce the disparity between their previously-trained and newly-updated versions. We then apply INFER$_{ONCE}$ to enhance them by providing influential exemplar replay and show the resultant $Recall@20$ metric in Table 3. Notably, the substantial performance improvements are reasonable as our replay strategy is technically orthogonal to their methodologies. More importantly, this indicates that INFER$_{ONCE}$ can thus synergistically boost their performance across all datasets.

Empirical Study on INFER$_{ONCE}$ (RQ4)

**Runtime Efficiency.** The first motivation of INFER$_{Vanilla}$ is for computation acceleration. We showcase the holistic runtime costs of all models on the largest dataset Foursquare in Figure 5. We observe that (1)
INFER\textsubscript{ONCE} achieves much less runtime cost compared to INFER\textsubscript{Vanilla} whilst being on par with incremental learning methods, i.e., ER-MIR, InvDeg, and Uniform, across various backbone models. (2) Compared to recent incremental recommender models, e.g., GraphSAIL and LWC-KD, our method exhibits faster convergence speed, showing the efficiency of our one-step estimation design.

**Comparison with INFER\textsubscript{Vanilla}**. To understand how INFER\textsubscript{ONCE} approximates INFER\textsubscript{Vanilla} with different model parameter sizes, we visually compare their calculated results based on Lastfm-2k and Gowalla datasets. Specifically, for both of these implementations, we first respectively compute their influence values and rank them accordingly. Then we enlarge the recommender model size around 16 times, e.g., from 38k to 606k for Lastfm-2k dataset. For each dataset, we randomly sample 1,000 candidates and compare their ranking differences produced by INFER\textsubscript{ONCE} and INFER\textsubscript{Vanilla}. Intuitively, for each data candidate, the smaller the ranking difference is, the more similar these two implementations would be. We plot the ranking difference distributions with kernel density estimation in Figure 6. We observe the following trends: (1) Irrespective of the backbone models’ size, these two implementations exhibit substantial overlap in their distributions, indicating that INFER\textsubscript{ONCE} closely resembles INFER\textsubscript{Vanilla} with not many deviations. (2) With the model size increasing, we notice that their ranking disparities slightly rise in central distribution regions which corresponds to the case of minor differences in rankings. However, considering our early empirical findings, we believe that such small disagreements are acceptable as INFER\textsubscript{ONCE} essentially has more focus on efficiency consideration.

**Parameter Study on Reservoir Size (RQ5)**

Lastly, we vary the reservoir size by adjusting the reply ratio, i.e., $K/|D|$, to investigate the INFER\textsubscript{ONCE} performance. The reply ratio is altered from 0.05 to 0.35 at intervals of 0.05 and we implement INFER\textsubscript{ONCE} to two backbones, i.e., TwoTower and LightGCN, on Gowalla dataset. As shown in Figure 7, the model performance continually improves with the increasing ratio from 0.05 to 0.25 (TwoTower) or 0.30 (LightGCN). However, after further expanding the reservoir, apart from the increasing time costs, we also notice performance degradation, e.g., dropping from 0.8247 to 0.8127 of AUC on TwoTower. The explanation is straightforward as over-training the historical data may introduce bias that is not descriptive of new data, which is reasonable in practice to reflect the user’s preference drifting phenomenon.

**Conclusion**

In this work, we investigate incremental learning for recommender systems via influential exemplar replay. While our vanilla implementation finds the most analytically influential samples, our advanced solution INFER\textsubscript{ONCE} optimizes toward the efficiency bottleneck. Extensive empirical evaluation demonstrates not only the generalization of our methods across four backbone models but also the performance superiority over state-of-the-art incremental recommender models on four public benchmarks. For future work, we plan to explore influence functions that encapsulate graph structural information (Liu et al. 2022; Wu et al. 2022). Another promising direction is to devise generative models (Li et al. 2020; Ho, Jain, and Abbeel 2020) capable of synthesizing user interactions from historical archives, enhancing the practicality through generative replay.

### Table 3: Recall@20 of INFER\textsubscript{ONCE}-enhanced models.

<table>
<thead>
<tr>
<th>Method</th>
<th>Lastfm-2k</th>
<th>TB2014</th>
<th>Gowalla</th>
<th>Foursquare</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphSAIL</td>
<td>0.0553</td>
<td>0.0078</td>
<td>0.0725</td>
<td>0.1042</td>
</tr>
<tr>
<td>+ INFER\textsubscript{ONCE}</td>
<td><strong>0.0708</strong></td>
<td><strong>0.0082</strong></td>
<td><strong>0.0815</strong></td>
<td><strong>0.1195</strong></td>
</tr>
<tr>
<td>SGCT</td>
<td>0.0620</td>
<td>0.0080</td>
<td>0.0783</td>
<td>0.1157</td>
</tr>
<tr>
<td>+ INFER\textsubscript{ONCE}</td>
<td><strong>0.0722</strong></td>
<td><strong>0.0088</strong></td>
<td><strong>0.0881</strong></td>
<td><strong>0.1284</strong></td>
</tr>
<tr>
<td>LWC-KD</td>
<td>0.0674</td>
<td>0.0081</td>
<td>0.0792</td>
<td>0.1194</td>
</tr>
<tr>
<td>+ INFER\textsubscript{ONCE}</td>
<td><strong>0.0721</strong></td>
<td><strong>0.0088</strong></td>
<td><strong>0.0902</strong></td>
<td><strong>0.1292</strong></td>
</tr>
</tbody>
</table>

### Figures

- **Figure 5**: The holistic runtime cost (s) on the Foursquare dataset including reservoir construction, incremental model training with early-stop, and evaluation (best view in color).
- **Figure 6**: Distributions of ranking differences between our two implementations by differing model sizes.
- **Figure 7**: Results of varying replay ratios.
Acknowledgments

We thank anonymous reviewers for their valuable comments. The work described here was partially supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 1422922, RGC GRF No. 2151185 and (RGC Research Impact Fund RS034-18; CUHK 2410021). Chenhao Ma was supported in part by NSFC Grant (No. 62302421). Yixiang Fang was supported in part by NSFC Grant (No. 62102341).

References


Basu, S.; Pope, P.; and Feizi, S. 2021. Influence functions in deep learning are fragile. In ICLR.


Song, Z.; Zhang, Y.; and King, I. 2023. Optimal Block-wise Asymmetric Graph Construction for Graph-based Semi-supervised Learning. In NeurIPS.


Wang, Z.; Shen, Y.; Zhang, Z.; and Lin, K. 2023. Feature staleness aware incremental learning for CTR prediction. In IJCAI.