Graph-Based Model-Agnostic Data Subsampling for Recommendation Systems

Xiaohui Chen*
Tufts University
Medford, Massachusetts, USA
xiaohui.chen@tufts.edu

Jiankai Sun
ByteDance Inc.
Bellvue, Washington, USA
jiankai.sun@bytedance.com

Taiqing Wang
ByteDance Inc.
Bellvue, Washington, USA
taiqing.wang@bytedance.com

Ruocheng Guo
ByteDance Inc.
London, UK
ruocheng.guo@bytedance.com

Li-Ping Liu
Tufts University
Medford, Massachusetts, USA
liping.liu@tufts.edu

Aonan Zhang*
Apple Inc.
Bellvue, Washington, USA
aonan_zhang@apple.com

ABSTRACT

Data subsampling is widely used to speed up the training of large-scale recommendation systems. Most subsampling methods are model-based and often require a pre-trained pilot model to measure data importance via e.g. sample hardness. However, when the pilot model is misspecified, model-based subsampling methods deteriorate. Since model misspecification is persistent in real recommendation systems, we instead propose model-agnostic data subsampling methods by only exploring input data structure represented by graphs. Specifically, we study the topology of the user-item graph to estimate the importance of each user-item interaction (an edge in the user-item graph) via graph conductance, followed by a propagation step on the network to smooth out the estimated importance value. Since our proposed method is model-agnostic, we can marry the merits of both model-agnostic and model-based subsampling methods. Empirically, we show that combing the two consistently improves over any single method on the used datasets. Experimental results on KuaiRec and MIND datasets demonstrate that our proposed methods achieve superior results compared to baseline approaches.

ACM Reference Format:
Xiaohui Chen, Jiankai Sun, Taiqing Wang, Ruocheng Guo, Li-Ping Liu, and Aonan Zhang. 2023. Graph-Based Model-Agnostic Data Subsampling for Recommendation Systems. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599834

1 INTRODUCTION

Recommendation systems learn user preferences through user-item interactions such as user clicks. Usually, a user click is considered as a positive sample that indicates the user's interest in the clicked item while a no-click between a user-item pair is considered as a negative sample. A click-through rate (CTR) prediction model outputs click probabilities of user-item pairs, and such probabilities can be used to rank items upon a user's request. A CTR model is trained using data collected from online platforms, where user-item pairs with no-clicks dominate. The imbalance of the dataset [4] justifies *negative sampling* (NS), which down-samples negative samples and significantly reduces model training costs.

In contrast to treating all data points as equally important, non-uniform subsampling aims to obtain more informative samples. Previous model-based methods [8, 9, 14, 32, 34, 35, 40] utilize a pilot model to assess the importance of data. When the pilot model is correctly pre-trained, one can achieve the optimal sampling rate: the method is measuring the importance of data using pilot prediction scores together with first and second-order derivatives of the loss function [35]. Since optimal negative sampling rates are proportional to the pilot model's prediction score, a high sampling rate indicates an inaccurate model prediction. Thus, one can interpret the sampling strategy as using hard negative samples (HNS) [8, 40].

We find that model-based sampling algorithms may not be applicable in real industrial scenarios. Figure 1 demonstrates how a real recommendation system deploys data subsampling. A user initiates a request to the online serving model and receives recommendations returned by the server. If the user clicks the recommended item, then a positive instance is collected. Otherwise, if the user does not click the item within a period of time, a negative instance is collected. Note that the pilot prediction score and other statistics are recorded in the instance to calculate the sampling rate. All instances are filtered by the data subsampling module before I/O to reduce the I/O and network bandwidth bottleneck. Offline models are trained with historical data before being deployed online.

In these scenarios, there are two unavoidable obstacles to the application of model-based sampling. First, offline model training is vulnerable to model misspecification, which leads to inferior results. Unfortunately, model misspecification is persistent due to online-offline discrepancy, especially in continuous integration and deployment (CI/CD) [28] in real systems. Second, coupling data subsampling and model training introduces extra dependencies across system modules, which increases system maintenance cost and brings about extra technical debt [27].

To maintain a scalable and sustainable data subsampling service, we propose model-agnostic data subsampling methods based

*The majority of this work was done while these authors were at Bytedance.

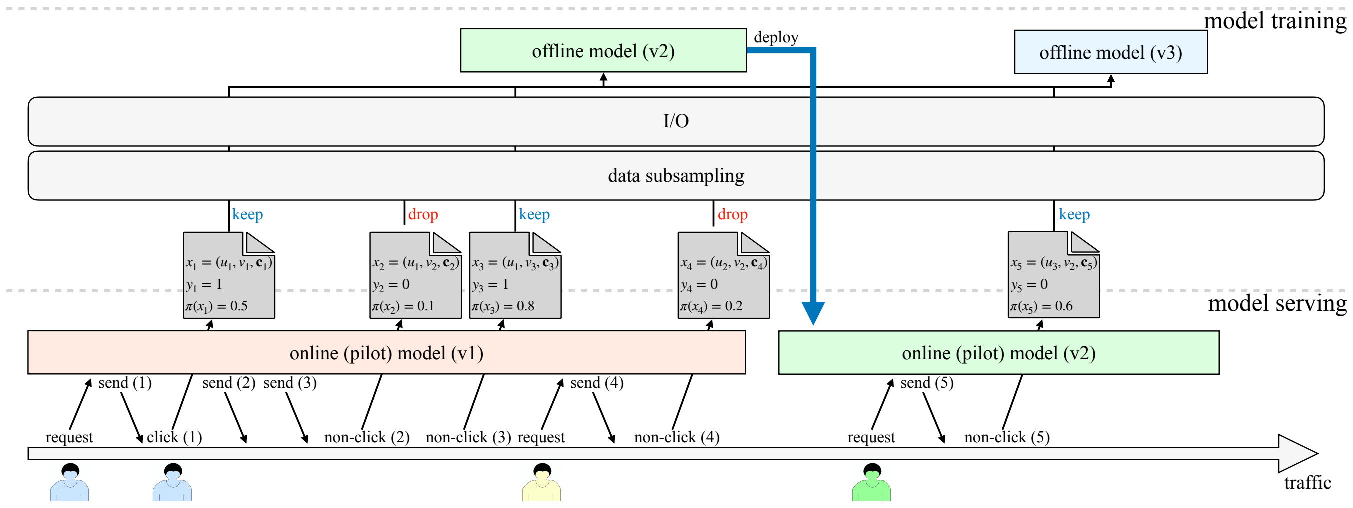

model training
offline model (v2)
deploy
offline model (v3)
I/O
data subsampling
keep
drop
keep
drop
keep
$x_1 = (u_1, v_1, \mathbf{c}_1)$ $y_1 = 1$ $\pi(x_1) = 0.5$
$x_2 = (u_1, v_2, \mathbf{c}_2)$ $y_2 = 0$ $\pi(x_2) = 0.1$
$x_3 = (u_1, v_3, \mathbf{c}_3)$ $y_3 = 1$ $\pi(x_3) = 0.8$
$x_4 = (u_2, v_2, \mathbf{c}_4)$ $y_4 = 0$ $\pi(x_4) = 0.2$
$x_5 = (u_3, v_2, \mathbf{c}_5)$ $y_5 = 0$ $\pi(x_5) = 0.6$
model serving
online (pilot) model (v1)
online (pilot) model (v2)
send (1)
send (2)
send (3)
send (4)
send (5)
request
click (1)
non-click (2)
non-click (3)
request
non-click (4)
request
non-click (5)
traffic

Figure 1: A simplified recommendation system. A user initiates a request to the online serving model and receives recommendations returned by the server. User clicks will trigger positive instances, while non-clicks will trigger negative instances. All instances are filtered by the data subsampling module. For model-based methods, subsampling rates are determined by online models (e.g. model v1), thus sub-optimal for offline training (e.g. model v2) purposes. When model v2 is deployed online, data subsampling is affected, producing inconsistent subsampling rates.

on user-item bipartite graphs. In the bipartite graph, two sets of nodes represent users and items separately, and each edge represent interactions between a user and an item. Then we treat edges as instances to consider subsampling. In the context of a bipartite graph, we reinterpret the idea of HNS using effective conductance [7]. An edge is considered a hard instance if the effective conductance over its two nodes is high. With this notion of hardness, we assign high sampling rates to edges with high effective conductance. Additionally, we exploit the bipartite graph to propagate and refine the sampling rates. Since our proposed method is model-agnostic, we can marry the merits of both model-agnostic and model-based subsampling methods. Empirically, we show that combing the two consistently improves over any single method on the widely used datasets. To summarize our contribution:

- we propose a model-agnostic method to measure data hardness scores via effective conductance on the user-item bipartite graph;
- we refine hardness scores via propagation on graphs;
- empirical results demonstrate that without requesting information from the pilot model, our model-agnostic methods achieve comparable or better results than model-based methods;
- we verify model-agnostic methods compliment model-based methods and combining them can achieve even better performance.

2 PRELIMINARIES

We consider a binary classification problem and let $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ be a training set of size N. Here $\mathbf{x}_n$ and y_n are, respectively, the feature vector and label of instance n. We study the generalized logistic regression (GLM) model, where the target model corresponding to

Algorithm 1 Hard Negative Sampling (HNS)

```
Given $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ and hardness score function $h(\cdot)$
Compute the counterfactual NSR $\pi(\mathbf{x}_n)$ via Eqn. (2).
for $n = 1, ..., N$ do
    Generate $\lambda_n \sim \mathbb{U}(0, 1)$.
    if $y_n = 1$ or $\lambda_n \le \pi(\mathbf{x}_n)$ then
        Include $\{\mathbf{x}_n, y_n, \pi(\mathbf{x}_n)\}$ in the training set.
    end if
end for
```

the offline model (e.g., model v2 before deployment in Figure 1) is represented as

$$f(\mathbf{x}; \theta) := p(y = 1|\mathbf{x}, \theta) = \frac{1}{1 + e^{-g(\mathbf{x};\theta)}}, \tag{1}$$

where the log-odd $g(\mathbf{x}; \theta)$ is implemented by a predictive model. Denote N_0 as the number of negative instances and $N_1 = N - N_0$ as the number of positive instances. We study the case where $\mathcal{D}$ is imbalanced, i.e., $N_0 \gg N_1$. Since information is sparsely distributed over many negative instances [34], we often use NS to reduce the dataset size and boost training efficiency.

An NS algorithm weighs each negative instance $\mathbf{x}$ with some measurement of its importance $\pi(\mathbf{x})$. The importance $\pi(\mathbf{x})$ is used as the negative sampling rate (NSR) of $\mathbf{x}$. The assignment of $\pi(\mathbf{x})$ follows a widely-used heuristic, which is exploiting "hard negative samples" [18, 25]. Sampling rates are proportional to non-negative hardness scores $h(\cdot)$:

$$\pi(\mathbf{x}_n) \propto h(\mathbf{x}_n), \quad \text{s.t.} \sum_{n=1}^{N_0} \pi(\mathbf{x}_n) = N_0\alpha, \tag{2}$$

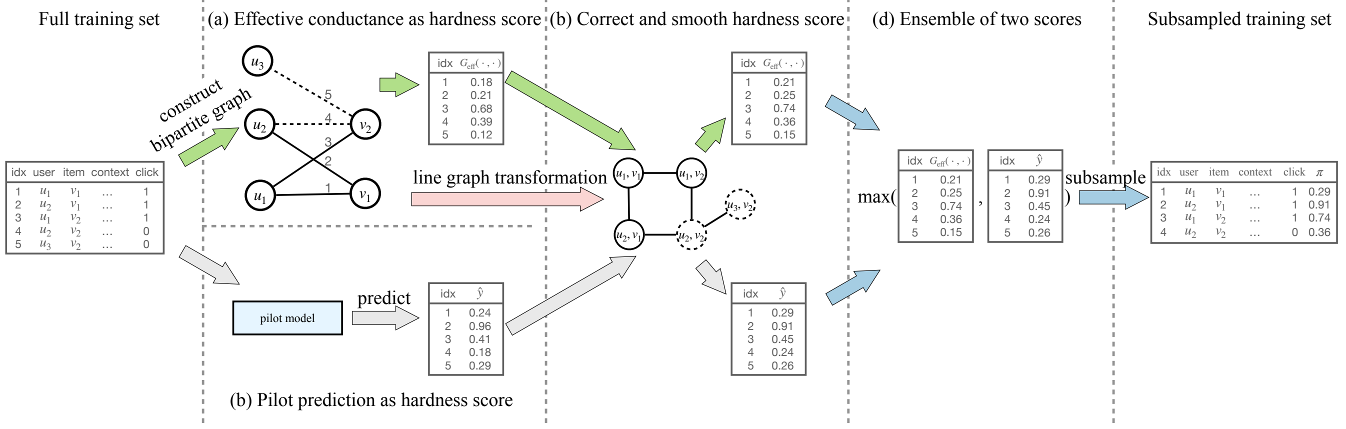

Full training set
(a) Effective conductance as hardness score
(b) Correct and smooth hardness score
(d) Ensemble of two scores
Subsampled training set
construct bipartite graph
line graph transformation
pilot model
predict
(b) Pilot prediction as hardness score
max(
subsample

Figure 2: Illustration of our model-agnostic subsampling framework. We construct the user-item bipartite graph from the training data and estimate the sample hardness using effective conductance (a); The user-item graph can be transformed into a line graph, and smooth the hardness score through propagation (c); Similarly, we can smooth the hardness score obtained from a model-based method (b); The smooth scores (a-c,b-c) can be further ensembled to calculate the final subsampling rate (d).

where $\alpha \in (0,1]$ is a pre-set average subsampling rate of negative instances. Suppose we have a pilot model $\tilde{f}(\cdot)$ corresponding to online model v1 in Figure (2). When $\tilde{f}(\cdot) = f(\cdot;\theta^*)$, that is $\tilde{f}(\cdot)$ has the same functional form as the target model, and θ^* is the true parameter. Then we can set a *model-based hardness score* $h_b(\mathbf{x}_n) = \tilde{f}(\mathbf{x}_n) = f(\mathbf{x}_n;\theta^*)$ to get a near-optimal sampling rate $\pi(\mathbf{x}_n)$ by Eqn. (2) [35]. Intuitively, a negative instance predicted with a higher score by the pilot model $\tilde{f}(\cdot)$ is more "surprising" and thus is harder for the target model $f(\cdot;\theta)$. Note that we use $\pi(\mathbf{x})$ for a positive instance to denote its counterfactual negative sampling rate. We demonstrate the HNS procedure in Algorithm 1.

Since data distribution shifts after subsampling, one needs to correct the log odds to get an unbiased estimation:

$$\hat{\theta} = \arg\max_{\theta} \sum_{n=1}^{N} \delta_n [y_n g(\mathbf{x}_n;\theta) - \log(1 + e^{g(\mathbf{x}_n;\theta)-\ell_n})], \quad (3)$$

where $\delta_n \in \{0,1\}$ is the subsampling indicator and $\ell_n := \log \pi(\mathbf{x}_n)$. [14] proves that log odds correction is more efficient than the IPW estimator [17].

However, when the pilot model is misspecified, optimal NS with pilot models is not achieved. In real scenarios, it may be error-prone to deploy model-based HNS methods since we persistently suffer from model misspecification problems due to online-offline model discrepancy and continuous integration and deployment. Thus, it is tempting to use *model-agnostic hardness score* $h_a(\cdot)$ to maintain a scalable and sustainable data subsampling service.

3 METHODOLOGY

An overview of our workflow is demonstrated in Figure 2. We introduce its technical details step by step.

3.1 MA-EC: Model-agnostic Negative Hardness Estimation via Effective Conductance

In this section, we consider the subsampling problem in the context of a bipartite graph formed from a recommendation problem. Let the bipartite graph be (U,V,E), where the node set $U = \{u_i\}_{i=1}^{M}$ represents M users, the node set $V = \{v_j\}_{j=1}^{Q}$ represents Q items, and the edge set $E = \{(u_{i_n}, v_{j_n})\}_{n=1}^{N}$ represent N user-item pairs. For each node pair n, $y_n \in \{0,1\}$ represents whether there is a positive interaction between u_{i_n} and v_{j_n}.

In our classification problem, each feature $\mathbf{x}_n = (u_{i_n}, v_{j_n}, \mathbf{c}_n)$, where $\mathbf{c}_n$ represent context features. And its label is y_n. We aim to compute the model-agnostic hardness $h_a(\cdot)$ of negative samples without referring to a pilot model.

We relate sample hardness to graph topology. Imagine the user-item graph as an electricity network, where each edge (u_{i_n}, v_{j_n}) is a conductor with conductance $G(u_{i_n}, v_{j_n})$, which measures the *edge*'s ability in transferring electrical current. We expect $G(u_{i_n}, v_{j_n})$ to be large when user u_{i_n} expresses direct preference of item v_{j_n}. Particularly, we set

$$G(u_{i_n}, v_{j_n}) = y_n. \quad (4)$$

It means that the conductance is one if there is a direct preference or 0 otherwise.

Then we consider the *effective conductance* $G_{\text{eff}}(u_{i_n}, v_{j_n})$ between u_{i_n} and v_{j_n}. It represents the *network*'s ability to transfer current from u_{i_n} to v_{j_n} (or the opposite direction). The effective conductance $G_{\text{eff}}(u_{i_n}, v_{j_n})$ is the reciprocal of effective resistance $R_{\text{eff}}(u_{i_n}, v_{j_n})$: they are defined as follows [1].

$$R_{\text{eff}}(u_{i_n}, v_{j_n}) = (\mathbf{e}[u_{i_n}] - \mathbf{e}[v_{j_n}])^\top \mathbf{L}^+ (\mathbf{e}[u_{i_n}] - \mathbf{e}[v_{j_n}]),$$
$$G_{\text{eff}}(u_{i_n}, v_{j_n}) = \frac{1}{R_{\text{eff}}(u_{i_n}, v_{j_n})}. \quad (5)$$

Here $\mathbf{e}[\cdot] \in \{0,1\}^{M+Q}$ is the one-hot encoding of a node in the graph, and $\mathbf{L}^+$ is the pseudo inverse of the Laplacian of the graph.

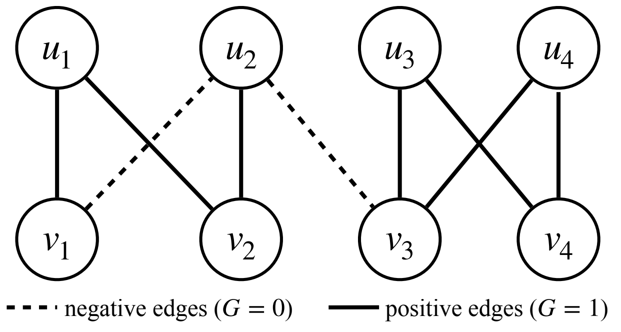


Figure 3: Interpret effective conductance on a bipartite graph. All positive edges are assigned conductances G=1, and all negative edges have $G = 0$. Consider two user-item pairs (u_2, v_1), (u_2, v_3) and their effective conductances $G_{\text{eff}}(u_2, v_1)$= 1/3, $G_{\text{eff}}(u_2, v_3)$=0. Effective conductances demonstrate user preference. We observe a 3-hop path between u_2 and v_1 ($u_2 - v_2 - u_1 - v_1$), but no path between u_2 and v_3, thus u_2 may prefer v_3 over v_1. (u_2, v_1) corresponds to a harder negative sample than (u_2, v_3).

Intuitively, if there are many conductible paths between u_{i_n} and v_{j_n}, then the effective conductance $G_{\text{eff}}(u_{i_n}, v_{j_n})$ is large.

Estimating sample hardness via effective conductance. Figure 3 demonstrates that effective conductance positively relates to sample hardness. Define the hardness score as

$$h_a(\mathbf{x}_n) = G_{\text{eff}}(u_{i_n}, v_{j_n}) - G(u_{i_n}, v_{j_n}). \quad (6)$$

For a negative sample, $G(u_{i_n}, v_{j_n}) = 0$. The effective conductance $G_{\text{eff}}(u_{i_n}, v_{j_n})$ is high when there are multiple high-conductance paths from u_{i_n} to v_{j_n}, demonstrating user's indirect preference to the item. When the indirect preference is high but (u_{i_n}, v_{j_n}) turns out to be negative, we identify it as a hard negative sample. For a positive sample, $h_a(\mathbf{x}_n)$ denotes its counterfactual hardness score by subtracting the direct conductor $G(u_{i_n}, v_{j_n})$ from $G_{\text{eff}}(u_{i_n}, v_{j_n})$ to eliminate the prior information given by the label. The hardness score is used to calculate the counterfactual NSR for log odds correction in Eqn. (3). We do not drop positive samples.

Implementation. Direct calculation of effective conductance is time-consuming. Instead, we first approximate the commute time distance $\text{comm}(u, v)$ through random walk using scientific computing tools [31]. Then we use the transformation $G_{\text{eff}}(u, v) = 2|E|/\text{comm}(u, v)$ [3] to convert the commute time into effective conductance.

3.2 Smoothing Hardness Scores through Edge Propagation

The effective conductance derived from the sparsified graph is noisy, leading to an inaccurate estimation of hardness scores[1]. We use the idea of graph propagation to smooth the hardness score. While existing machine learning literature focuses on propagating node attributes (e.g., node features or labels) to smooth out node-level uncertainty [19, 20], propagating edge attributes is underexplored.

We reduce edge propagation to node propagation by transforming user-item bipartite graph (U, V, E) into its corresponding line graph $L(U, V, E) = (V_L, E_L)$, where $V_L = E$ and E_L is the collection of edge pairs that share the same node [15].

Uncertainty propagation. Denote $G_{\text{eff}} := (G_{\text{eff}}(u_{i_n}, v_{j_n}))_{n=1}^N \in \mathbb{R}^N$, $Y := (y_n)_{n=1}^N \in \{0, 1\}^N$ to be the vector of the effective conductance scores and edge labels, respectively. Similar to [20], we normalize the effective conductance G_{eff} as the estimated score Z and calculate the uncertainty score B as the absolute residual between Z and Y

$$Z = \frac{G_{\text{eff}} - \min(G_{\text{eff}})}{\max(G_{\text{eff}}) - \min(G_{\text{eff}})}, \quad B = |Y - Z|. \quad (7)$$

Here we use the min-max normalization to restrict the hardness score to be within the range $[0, 1]$. Denote $S = D_L^{-1/2} A_L D_L^{-1/2}$, where A_L, D_L are the adjacency matrix and the degree matrix of the line graph, respectively. We smooth the uncertainty by solving the following optimization problem [42]:

$$\hat{B} = \underset{W}{\text{argmin}}\ \text{tr}(W^T (I - S) W) + \mu ||W - B||_F^2. \quad (8)$$

The first term in Eqn. (8) restricts the difference of uncertainties in neighboring nodes. And the second term constrains the smoothed uncertainty to be close to the initial uncertainty, with the coefficient μ controlling the strength of the constraint. With the smoothed uncertainty vector $\hat{B}$, we correct the hardness estimation by reversing Eqn. (7)

$$\hat{Z} = Y + (-1)^Y \hat{B}. \quad (9)$$

[19, 42] also introduced an iterative approximation approach. Let $\gamma = 1/(1 + \mu)$ and

$$B^{t+1} = (1 - \gamma) B + \gamma S B^t, \quad B^0 = B. \quad (10)$$

Then $B^t \longrightarrow \hat{B}$ when $t \longrightarrow \infty$.

However, this iterative approach is not scalable as the transformed line graph has $|E_L| = (\sum_{u_i \in U} \text{Deg}(u_i)^2 + \sum_{v_j \in V} \text{Deg}(v_j)^2)/2 - N$ edges in total, where $\text{Deg}(\cdot)$ represents node degree. Alternatively, we can directly propagate edge uncertainty along the original graph (U, V, E), which only contains $|E| = (\sum_{u_i \in U} \text{Deg}(u_i) + \sum_{v_j \in V} \text{Deg}(v_j))/2$ edges in total. The propagation rule over edges is as follows

$$B_n^{t+1} = (1 - \gamma) B_n + \gamma \frac{m^t(u_{i_n}) + m^t(v_{j_n}) - 2Z_n}{\text{Deg}(u_{i_n}) + \text{Deg}(v_{j_n}) - 2}, \quad (11)$$
$$\text{where } m^t(u) = \sum_{n:\ u = u_{i_n}} B_n^t,\ m^t(v) = \sum_{n:\ v = v_{j_n}} B_n^t,\ B_n^0 = B_n.$$

Using message passing mechanisms, we store the aggregated uncertainty $m^t(u)$ in u and then update the uncertainty B^{t+1} by applying the rule above.

Score propagation. Instead of propagating uncertainty, we can directly propagate the scores $\hat{Z}$ by iterating

$$Z^{t+1} = (1 - \gamma) \hat{Z} + \gamma S Z^t, \quad Z^0 = \hat{Z} \quad (12)$$

until convergence. After obtaining the final hardness scores, we rescale them to match the average subsampling rates α. We find

[1]Since negative edges are ignored to calculate effective conductance. According to Eqn. (6), any isolated positive edges without a multi-hop connection between its endpoints will have zero hardness score without graph propagation.

that smoothing hardness scores benefits both model-agnostic and model-based methods.

3.3 Combining Model-agnostic and Model-based Methods

We have described our model-agnostic data subsampling methods in previous sections. Some hard instances may be overlooked by model-agnostic methods while they can be captured by model-based methods. Hence, in this section, we show how to combine these two methods to marry the merits of both and achieve a better sampling performance.

Given a sample $\mathbf{x}$, model-agnostic and model-based subsampling methods calculate their corresponding sampling rate $\pi_{\mathcal{D}}(\mathbf{x})$ and $\pi_\phi(\mathbf{x})$ respectively. Particularly $\pi_\phi(\mathbf{x})$ is the subsampling rate for $\mathbf{x}$ by using pilot model $h_b(\cdot) := \tilde{f}(\cdot;\phi)$, and $\pi_{\mathcal{D}}(\mathbf{x})$ is the subsampling rate using model-agnostic hardness score $h_a(\cdot)$ in Eqn. (6):

$$\begin{aligned}\pi_\phi(\mathbf{x}) &\propto \max\Big(\min(\rho_\phi h_b(\mathbf{x}), \varrho_\phi), 1\Big),\\ \pi_{\mathcal{D}}(\mathbf{x}) &\propto \max\Big(\min(\rho_{\mathcal{D}} h_a(\mathbf{x}), \varrho_{\mathcal{D}}), 1\Big).\end{aligned} \tag{13}$$

$(\varrho_\phi, \varrho_{\mathcal{D}})$ is the minimum sampling rate, and $(\rho_\phi, \rho_{\mathcal{D}})$ are tuned to meet the average subsampling rate α.

We propose three simple yet effective heuristic strategies to combine these two methods and get the final sampling rate: *maximum*, *mean*, and *product*.

$$\pi_{\text{max}}(\mathbf{x}) = \rho_{\text{max}}\max\big(\pi_{\mathcal{D}}(\mathbf{x}), \pi_\phi(\mathbf{x})\big); \tag{14}$$
$$\pi_{\text{mean}}(\mathbf{x}) = \frac{(\pi_{\mathcal{D}}(\mathbf{x}) + \pi_\phi(\mathbf{x}))}{2};$$
$$\pi_{\text{prod}}(\mathbf{x}) = \min\Big(\max\big(\rho_{\text{prod}}\pi_{\mathcal{D}}(\mathbf{x})\pi_\phi(\mathbf{x})\big),\ \varrho_{\text{prod}}), 1\Big).$$

Note ϱ_{prod} is an extra hyperparameter when applying product combination. And $(\rho_{\text{max}}, \rho_{\text{prod}})$ are tuned to normalize the average sample rate to α.

After the subsampling rate combination, each $\mathbf{x}$ will be sampled with probability in Eqn. (14). All the sampled instances will follow the normal training protocol to optimize the training objective as shown in Eqn. (3) which guarantees the final result to be well-calibrated.

3.4 Theoretical bottlenecks of model-based subsampling

Above, we propose a principled approach to determine the sample's hardness by analyzing graph structures within the data without referring to a pilot model. We advocate this model-agnostic sampling method due to the dilemma of unavoidable model misspecification in the online-offline discrepancy setup. The critical assumption of a correctly specified pilot model is needed to derive a theoretically efficient data subsampling method. Under pilot model misspecifications, it is even non-trivial to guarantee model consistency (the subsampled estimator has the same limit as the full data estimator) for the log odds correction estimator (Eqn. 3) [9, 14, 29]. Therefore, pilot model misspecification is a realistic but underexplored problem in statistics.

	#users	#items	#instances	#pos.:#neg.
KuaiRec	5,765	10,679	11,959,745	1:35
MIND	94,057	22,771	8,236,715	1:25

Table 1: Dataset statistics.

4 EXPERIMENTS

We compare various subsampling methods on downstream target model performance. First, for model-based sampling, we show that pilot misspecification will lead to discrepancies in model performance. Second, we report empirical results over two datasets to demonstrate the superiority of our model agnostic subsampling method. Third, we conduct extensive ablation studies to investigate the effectiveness of model-agnostic hardness score, score propagation, and the benefit of ensembling model-agnostic and model-based methods. Finally, we discuss effective resistance and its relationship to negative sampling.

4.1 Datasets and data pre-processing

We demonstrate our empirical results using KuaiRec [10] and Microsoft News Dataset (MIND) [37]. The statistics of the pre-processed datasets are shown in Table 1. For both datasets, we use 80% of the data for training, 10% for validation, and 10% for testing. We report all experimental results for 8 runs with random initializations on one random data split. We set the average subsampling rate $\alpha = 0.2$ on training data for both datasets.

KuaiRec. KuaiRec is a recommendation dataset collected from the video-sharing mobile app "Kuaishou". The dataset is generally a sparse user-item interaction matrix with a fully-observed small submatrix. We crop the fully-observed submatrix and only consider the rest entries in the sparse matrix since those data are collected under natural settings. We use the label "watch_ratio", which represents the total duration of a user watching a video divided by the video duration. Since a user may watch a video multiple times, the watch ratio can be larger than 1. In the experiment, we consider a user like a video (positive) if the "watch_ratio" is larger than 3.

MIND. MIND is a large-scale news recommendation dataset with binary labels indicating users' impressions of recommended news. We do not use the content data in each news corpus during the experiment. The MIND dataset does not require extra pre-processing.

4.2 Baselines and model selections

Baseline subsampling methods. We consider two baselines – the first baseline is the model-agnostic uniform negative sampling; the second is a model-based near-optimal sampling method (Opt-Sampling) [35], which relies on the prediction scores of a pilot model as the hardness score to calculate sample rates.
We study the scenario where data subsampling rates are pre-computed and do not change during model training. This scenario is aligned with industrial applications such as recommendation systems, where negative data are subsampled before saving in distributed storage to get rid of huge saving costs. Our method contrasts previous studies to re-compute and subsample data on the fly. For example, SRNS [8]

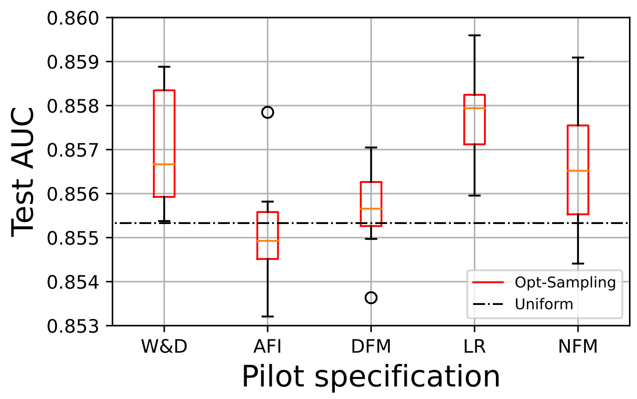


Figure 4: Pilot misspecifications lead to inconsistent model performance in Opt-Sampling.

quantifies sample hardness by estimating the variance of the model prediction along training epochs, which is dynamically updated each time model sees the sample. IRGAN [36] uses a GAN model where the generator draws negative samples from the negative pool and mixes them with positive samples. A discriminator classifies data and guides the generator to learn to sample hard negatives. These methods are not comparable to our work. To the best of our knowledge, Opt-Sampling [35] is the only representative method that allows for a fair comparison in the specific scenario considered by this work.

Model architecture. We choose the wide and deep model (W&D) [6] as our training target model to validate the effectiveness of our model-agnostic method. For model-based subsampling methods, we need to pre-train a pilot model to estimate the sample hardness. To test the effect of pilot misspecification, we consider five types of pilot models: (1) W&D model [6]; (2) Linear logistic regression (LR); (3) Automatic feature interaction selection model (AFI) [22]; (4) Neural factorization machines (NFM) [16]; (5) Deep factorization machine (DFM) [12]. In the rest of the experiments, unless otherwise specified, W&D is used as the pilot model since it shares the same architecture as the target model (consistent pilot). All pilot models are trained using 10% of the training data.

4.3 Pilot misspecification

Model-based subsampling methods can be sensitive to a misspecified pilot model. We fix the target model architecture and tune the pilot model architectures to study their effect on the KuaiRec dataset. Pilot models affect target model performance only through their generated samples. Figure 4 demonstrates the target model performance with different pilot model specifications. Target models' test AUC varies from 0.8557 (AFI) to 0.8577 (LR) when changing pilots. The AUC difference is significant since the standard deviation is around 0.001, demonstrating a potential loss in large-scale recommendation systems processing millions of data points daily. This result consolidates the negative effect of pilot misspecification, which justifies using model-agnostic subsampling approaches.

	W&D	AFI	DFM	LR	NFM
Pilot Train AUC↑	0.9917	0.9883	0.9922	0.9487	0.9999
Pilot Test AUC↑	0.8023	0.7919	0.7968	0.8192	0.8082

Table 2: LR as a pilot model gives best pilot test AUC, thus yielding better subsampling performance

	Uniform	Opt-Samp.	MA-EC
Test AUC↑	0.8474±0.0012	0.8511±0.0009	0.8519±0.0005

Table 3: LR as the target model: Model specification that is the best for a pilot model does not indicate it is the best for the target model.

In Figure 4, there is a big difference in target models' test performance among misspecified pilot models. We further study the quality of pilot models w.r.t. their predictive performance. Table 2 demonstrates pilot models' training and testing AUC. Deep pilot models may overfit the training data since we use 1/10 data to pretrain pilot models. LR is less vulnerable to overfitting and achieves the highest test AUC. There is a caveat to using a misspecified pilot model, e.g. AFI in our case since the outcome can be worse than uniform sampling even when AFI can make reasonable predictions. Using a consistent pilot model (W&D) is a safer choice: even when its predictive performance is worse than a misspecified NFM pilot model, the target model can benefit from pilot model consistency and achieve better testing AUC.

Note that W&D is a strong architecture on the KuaiRec dataset. Table 3 shows that when using LR as the target model and trained on the same amount of data, its performance is worse than the one using W&D as the target model: W&D achieves 0.8553 in testing AUC with uniform sampling while LR gets 0.8474. Table 3 demonstrates both model-based sampling with a consistent pilot (Opt.Samp.) and model-agnostic sampling (MA-EC) can improve model performance. MA-EC consistently helps, disregarding the model specification.

4.4 Experiment results on two datasets

We train the target model with different data sampling strategies and evaluate the model performance based on the area under the receiver operating characteristic curve (AUC). The results of all training configurations are presented in Figure 5. The comparison between the methods shows that the MA-EC strategy consistently outperforms the uniform sampling baseline in both datasets. Furthermore, when applied to the KuaiRec dataset, the MA-EC strategy achieved comparable performance to the Optimal Sampling method. However, in the MIND dataset, the Optimal Sampling approach did not improve upon the uniform sampling baseline and was outperformed by the MA-EC strategy. These results demonstrate the effectiveness of the MA-EC method in achieving improved performance in different datasets.

Additionally, the results were improved by incorporating a smoothing technique for hardness estimation through propagation. The performance of combining the model-agnostic and model-based

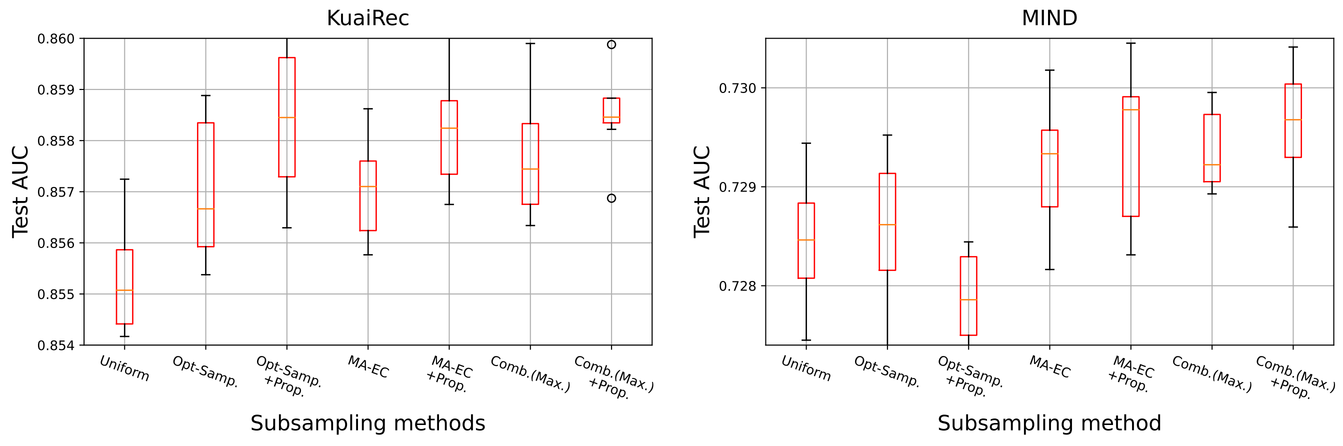


Figure 5: Experimental results on KuaiRec and MIND datasets.

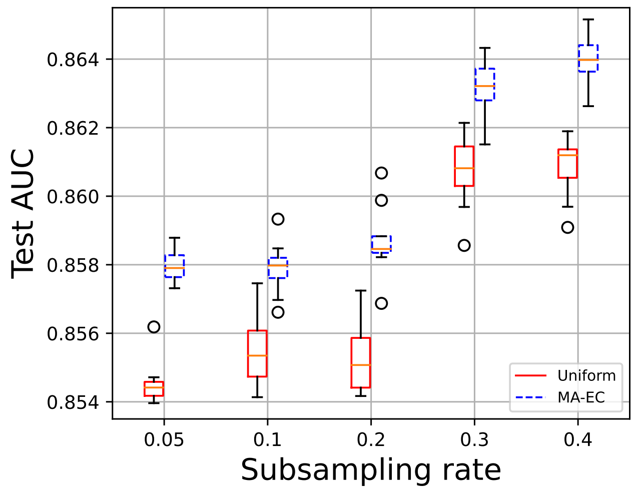


Figure 6: Test AUC with different subsampling rates. MA-EC consistently leads to better model performance over different subsampling rates.

methods using the maximum strategy was also reported. It was observed that the ensemble approach outperformed each individual method in both datasets. The smoothed scores from Opt-Sampling and MA-EC were also combined using the ensemble approach, resulting in the best performance as shown in the last column of the results. These findings highlight the benefits of combining multiple techniques to achieve even better performance in the target model training process. The use of smoothing and ensembling strategies demonstrates the potential for further improvements in data sampling methods and the effectiveness of combining multiple approaches to achieve better results.

$\pi_\phi(\mathbf{x})$ as major score	
no flip	flip $\pi_\phi(\mathbf{x}) < 0.2$ & $\pi_\mathcal{D}(\mathbf{x}) > 0.8$
0.8570 ± 0.0013	0.8576 ± 0.0007

$\pi_\mathcal{D}(\mathbf{x})$ as major score	
no flip	flip $\pi_\mathcal{D}(\mathbf{x}) < 0.2$ & $\pi_\phi(\mathbf{x}) > 0.8$
0.8570 ± 0.0009	0.8574 ± 0.0011

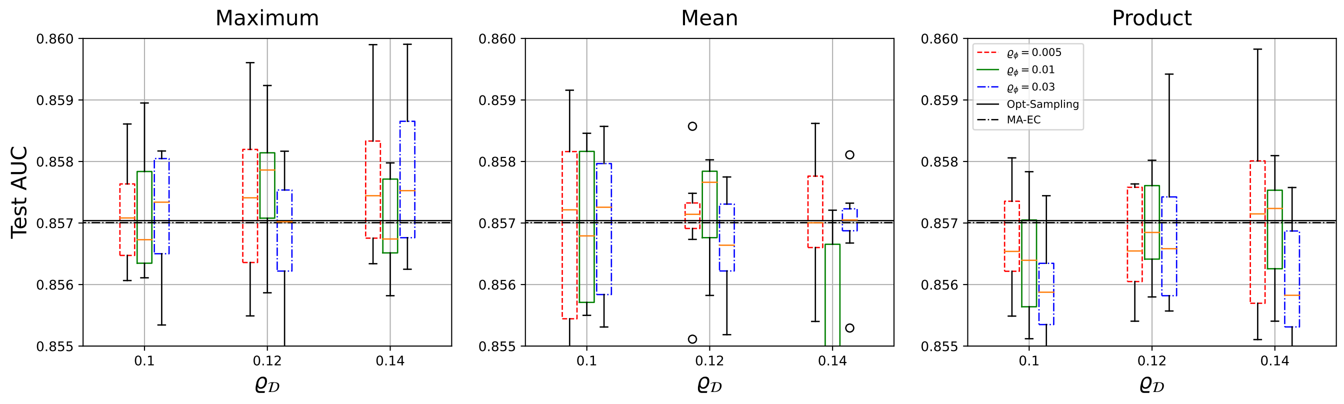
Table 4: Control experiment. Some negative samples have low subsampling rates in one method but have large subsampling rates in the other. Flipping those subsampling rates from small to large improves model performance.

4.5 Ablation studies

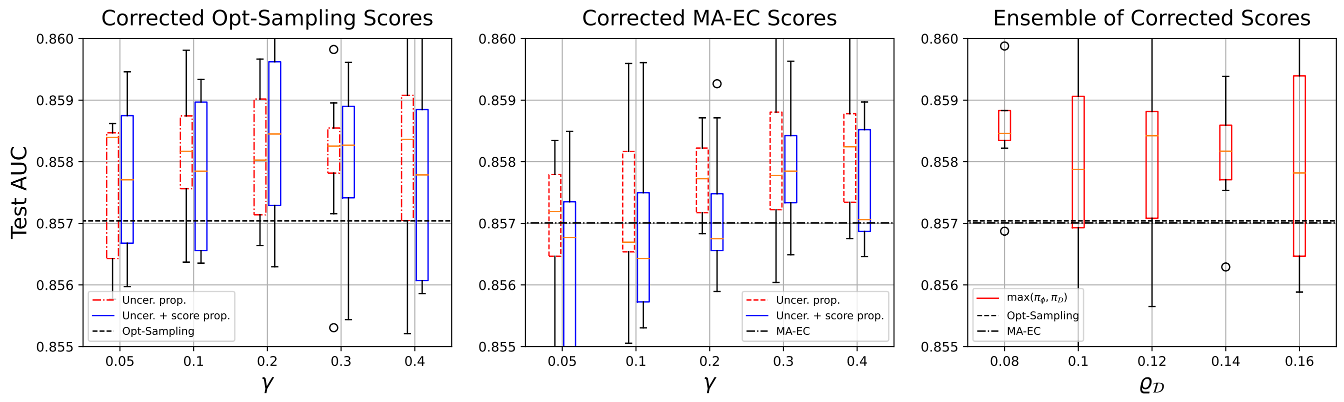
We use the KuaiRec dataset to conduct extensive ablation studies. First, we study if our method consistently outperforms baselines under different subsampling rates. Second, we show that MA-EC and Opt-Sampling can estimate sample hardness from different perspectives by designing a variable control experiment. And we investigate the effectiveness of different ensemble strategies. Third, we show how smoothing scores help further improve model performance.

Subsampling rate. We compare uniform sampling with the best of our methods by ensembling smoothed scores from Opt-Sampling and MA-EC. Figure 6 demonstrates our methods consistently outperform uniform sampling under different subsampling rates.

Ensemble strategies. To investigate whether hardness scores from MA-EC and Opt-Sampling complement each other, we design the following control experiment: (1) we assign instances with subsampling rates from each method; (2) for instances that have inconsistent subsampling rates between two methods, we flip their subsampling rates into the other method. For example, we assign instances that have $\pi_\mathcal{D}(\mathbf{x}) < 0.2$ and $\pi_\phi(\mathbf{x}) > 0.8$ with the subsampling rate $\pi_\phi(\mathbf{x})$, and the rest instances with $\pi_\mathcal{D}(\mathbf{x})$. Table 4 shows



(a) Model performance of maximum, mean, and product ensembles.



(b) Model performance of corrected scores in Opt-Sampling, MA-EC, and their ensemble.

Figure 7: Abaltion studies on ensemble strategies and score correction.

the results of the control experiments, where we achieve better model performance by assigning most of the sample with one set of scores and flipping part of the sample scores. This verifies that some hard negative instances might be overlooked by one method and can be discovered by the other.

The control experiment justifies ensembling MA-EC and Opt-Sampling. We experiment with ensemble strategies (maximum, mean, and product). Figure 7(a) shows the box plot of the three ensemble methods. For each method, we present nine configurations of the hyperparameters $(\varrho_{\mathcal{D}}, \varrho_\phi)$, where $\varrho_{\mathcal{D}} \in \{0.1, 0.12, 0.14\}$ and $\varrho_\phi \in \{0.005, 0.01, 0.03\}$. For product strategy, we set hyperparameter $\varrho_{\text{prod}} = 0.005$ for all experiments. We observe that the maximum strategy consistently gives comparable or better results than Opt-Sampling and MA-EC. While in mean and product strategies, we do not observe significant improvement. For the product strategy, the model performance even deteriorates. MA-EC needs to compute effective conductance to calculate subsampling rates. Effective conductance computation is not our bottleneck since it can be reused once computed. MA-EC is model-agnostic, and thus can support training with different target models.

Score correction. We investigate the effectiveness of correcting the hardness scores via graph propagation. We are interested in applying score correction in Opt-Sampling and MA-EC and their ensemble. Additionally, in applying both score correction and score ensemble, we can try either *ensembling corrected scores* or *correcting ensemble scores.* As we find the latter consistently results in worse performance, we only present the result of the former in this work. Figure 7(b) reports the model performance of our ablation study. For the experiments of correcting scores estimated from Opt-Sampling and MA-EC, we explore the propagation coefficient $\gamma \in \{0.05, 0.1, 0.2, 0.3, 0.4\}$. For each coefficient, we iterate to smooth the scores until convergence. Uncertainty propagation significantly improves model performance for both subsampling approaches. In score propagation, which runs on scores corrected by uncertainty propagation, model performance slightly improves in Opt-Sampling and worsens in MA-EC. In the ensemble of corrected scores, we combine the hardness scores from the best configurations of both methods via maximum strategy. The reported result demonstrates that the ensemble strategy improves model performance over not only the original scores but also the corrected scores.

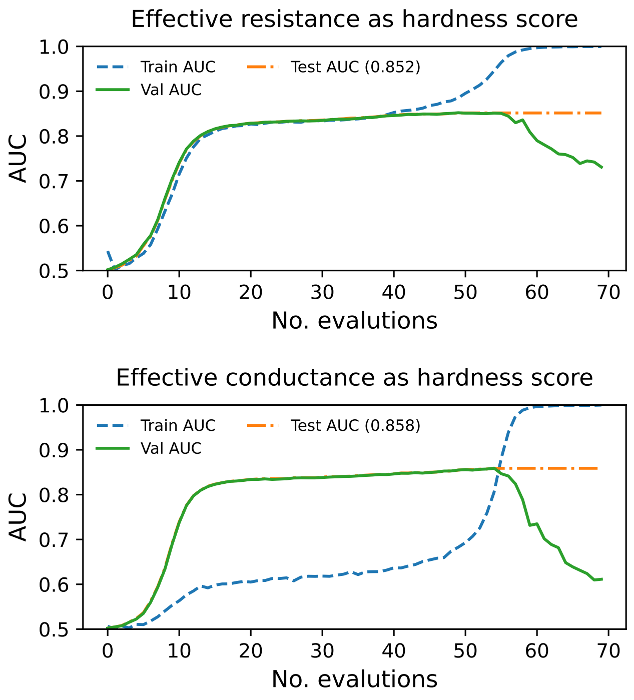


Figure 8: Comparison of a single target model run using effective conductance and effective resistance as hardness scores for negative sampling.

4.6 A note on effective resistance

The effective resistance $R_{\text{eff}}(u, v) = 1/G_{\text{eff}}(u, v)$ is often used for graph sparsification [30]. An edge with high effective resistance is considered important in maintaining graph topology. Since the definitions of edge importance using effective conductance and effective resistance run against each other, we demonstrate that defining edge importance with effective resistance is not applicable in our scenario.

We compare two model-agnostic (MA) subsampling methods with effective resistance (MA-ER) and effective conductance (MA-EC) as the hardness scores. The effective resistance is computed on the graph where all edges have unit resistances. We demonstrate that MA-ER fails to capture hard negative instances. On the KuaiRec dataset, MA-ER yields an average test AUC of 0.8535, which is worse than uniform sampling (0.8553). To unravel how MA-ER and MA-EC affect model training, we randomly select a run from each method to visualize the model training metrics. In Figure 8, when using MA-ER, training AUC remains the same as testing AUC before convergence. Besides, the model converges earlier. In sharp contrast, in MA-EC, there is a huge gap between training AUC and testing AUC. The gap shows that training instances are overall harder than those in the test set. This verifies that MA-EC discovers hard negatives while MA-ER does not.

5 RELATED WORKS

Negative sampling. Hard negative sampling is pervasively used in recommendation systems. PinSage [39] shows using curriculum learning with negative sampling is effective, implying hard negatives are helpful in the late stage of training. Metasearch offline datastream combines documents with in-batch mismatched queries with the highest similarity scores as hard negative samples [18]. Hard negative sampling is also justified in theory. Fithian and Hastie [9] uses a pilot model to select examples whose responses were rare given their features preferentially. Han et al. [14] explores local uncertainty sampling for multi-class classification. [34] proves that for generalized linear logistic regression models, the optimal negative sampling rate is proportional to the pilot prediction value. There are also approaches [5, 38] that utilize item popularity to estimate the importance of negative instances.

Graph sparsification. Graph sparsification tries to drop nodes and edges while preserving the graph structure. Ghosh et al. [11], Spielman and Srivastava [30] studied edge sampling with effective resistance to preserve graph Laplacian. Satuluri et al. [26] used the Jaccard similarity score to measure node similarity and pick the top-K associated edges for each node by their similarity. Other methods include sampling edges by counting the number of its associate triangles [13], quadrangles [24], using local graph information [21], or using deep neural networks to coarsen the graph [2]. Graph sparsification has been explored to preserve or improve downstream task performance [33, 41]. However, these method requires an end-to-end training framework and thus cannot be used for continuous deployment where subsampling rates are static. To the best of our knowledge, model-agnostic methods based on graph sparsification are under-explored in the literature.

6 CONCLUSION

We propose model-agnostic hard negative subsampling methods using the effective conductance on the user-item bipartite graph as the hardness score. We further exploit the graph structure by score propagation. Our model-agnostic complements model-based optimal sampling and provide a sustainable and consistent data subsampling solution to real-world recommendation systems with long-term impact. We discuss the cost upon deployment, its social impact, and future directions.

Deployment cost. Deploying a model-agnostic sampling service requires a database with graph computing engines that support effective conductance lookup given pairs of user-item ids, which is computationally cheap in practice. Online serving will not be affected, so there is no change in serving latency. Offline models are trained only on sub-sampled data to enjoy reduced computational costs. It takes 150 and 80 seconds to estimate the sampling rate for KuaiRec and MIND datasets, respectively. The cost of running MA-EC sampling is negligible compared to model training.

Social impact. With less data, one can train models with less GPU time and use less data storage. Given the quantity of data and the requirement to iterate the model periodically in the industry, the data subsampling method can significantly reduce the carbon footprint.

Future work. Our proposed sampling methods can be used to select better neighbors for learning graph embeddings. We can further improve MA-EC by (i) exploiting context information to identify hard negative samples. (ii) utilizing negative edges when calculating effective conductance. We leave those as future work.

REFERENCES

[1] Charles K Alexander, Matthew NO Sadiku, and Matthew Sadiku. 2007. Fundamentals of electric circuits. McGraw-Hill Higher Education Boston.

[2] Chen Cai, Dingkang Wang, and Yusu Wang. 2021. Graph coarsening with neural networks. arXiv preprint arXiv:2102.01350 (2021).

[3] Ashok K Chandra, Prabhakar Raghavan, Walter L Ruzzo, Roman Smolensky, and Prasoon Tiwari. 1996. The electrical resistance of a graph captures its commute and cover times. computational complexity 6, 4 (1996), 312–340.

[4] Nitesh V Chawla. 2009. Data mining for imbalanced datasets: An overview. Data mining and knowledge discovery handbook (2009), 875–886.

[5] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On sampling strategies for neural network-based collaborative filtering. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 767–776.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In Proceedings of the 1st workshop on deep learning for recommender systems. 7–10.

[7] Fan RK Chung. 1997. Spectral graph theory. Vol. 92. American Mathematical Soc.

[8] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and robustify negative sampling for implicit collaborative filtering. Advances in Neural Information Processing Systems 33 (2020), 1094–1105.

[9] William Fithian and Trevor Hastie. 2014. Local case-control sampling: Efficient subsampling in imbalanced data sets. Annals of statistics 42, 5 (2014), 1693.

[10] Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. 2022. KuaiRec: A Fully-observed Dataset and Insights for Evaluating Recommender Systems. arXiv preprint arXiv:2202.10842 (2022).

[11] Arpita Ghosh, Stephen Boyd, and Amin Saberi. 2008. Minimizing effective resistance of a graph. SIAM review 50, 1 (2008), 37–66.

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. arXiv preprint arXiv:1703.04247 (2017).

[13] Michael Hamann, Gerd Lindner, Henning Meyerhenke, Christian L Staudt, and Dorothea Wagner. 2016. Structure-preserving sparsification methods for social networks. Social Network Analysis and Mining 6, 1 (2016), 1–22.

[14] Lei Han, Kean Ming Tan, Ting Yang, and Tong Zhang. 2020. Local uncertainty sampling for large-scale multiclass logistic regression. The Annals of Statistics 48, 3 (2020), 1770–1788.

[15] Frank Harary and Robert Z Norman. 1960. Some properties of line digraphs. Rendiconti del circolo matematico di palermo 9, 2 (1960), 161–168.

[16] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. 355–364.

[17] Daniel G Horvitz and Donovan J Thompson. 1952. A generalization of sampling without replacement from a finite universe. Journal of the American statistical Association 47, 260 (1952), 663–685.

[18] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2553–2561.

[19] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. 2020. Combining label propagation and simple models out-performs graph neural networks. arXiv preprint arXiv:2010.13993 (2020).

[20] Junteng Jia and Austion R Benson. 2020. Residual correlation in graph neural network regression. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 588–598.

[21] Can M Le. 2021. Edge Sampling Using Local Network Information. J. Mach. Learn. Res. 22 (2021), 88–1.

[22] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2636–2645.

[23] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. 2019. Measuring Calibration in Deep Learning.. In CVPR workshops, Vol. 2.

[24] Arlind Nocaj, Mark Ortmann, and Ulrik Brandes. 2014. Untangling hairballs. In International Symposium on Graph Drawing. Springer, 101–112.

[25] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2020. Contrastive learning with hard negative samples. arXiv preprint arXiv:2010.04592 (2020).

[26] Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. 2011. Local graph sparsification for scalable clustering. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. 721–732.

[27] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. Advances in neural information processing systems 28 (2015).

[28] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. 2017. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. IEEE Access 5 (2017), 3909–3943.

[29] Xinwei Shen, Kani Chen, and Wen Yu. 2021. Surprise sampling: Improving and extending the local case-control sampling. (2021).

[30] Daniel A Spielman and Nikhil Srivastava. 2008. Graph sparsification by effective resistances. In Proceedings of the fortieth annual ACM symposium on Theory of computing. 563–568.

[31] Christian L Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. 2016. NetworKit: A tool suite for large-scale complex network analysis. Network Science 4, 4 (2016), 508–530.

[32] Daniel Ting and Eric Brochu. 2018. Optimal subsampling with influence functions. Advances in neural information processing systems 31 (2018).

[33] Guihong Wan and Harsha Kokel. 2021. Graph sparsification via meta-learning. DLG@ AAAI (2021).

[34] HaiYing Wang. 2020. Logistic regression for massive data with rare events. In International Conference on Machine Learning. PMLR, 9829–9836.

[35] HaiYing Wang, Aonan Zhang, and Chong Wang. 2021. Nonuniform Negative Sampling and Log Odds Correction with Rare Events Data. Advances in Neural Information Processing Systems 34 (2021), 19847–19859.

[36] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. 515–524.

[37] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 3597–3606.

[38] Ga Wu, Maksims Volkovs, Chee Loong Soon, Scott Sanner, and Himanshu Rai. 2019. Noise contrastive estimation for one-class collaborative filtering. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. 135–144.

[39] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 974–983.

[40] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. 785–788.

[41] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In International Conference on Machine Learning. PMLR, 11458–11468.

[42] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. Advances in neural information processing systems 16 (2003).

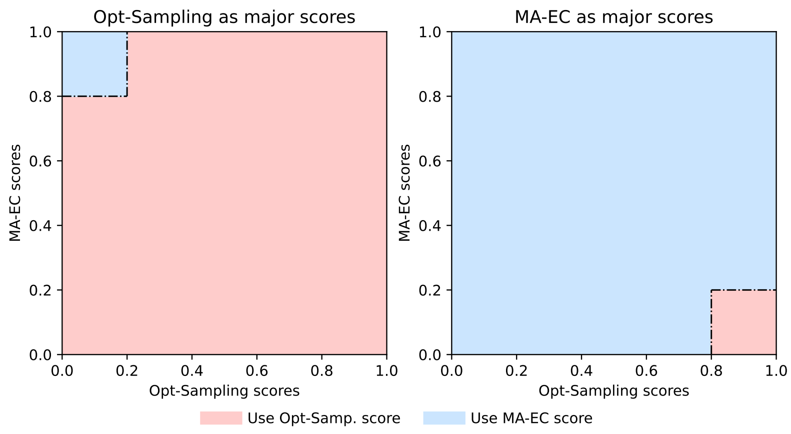


Figure 9: Illustration of the control experiment

	KuaiRec				
	#epoch	batch size	learning rate	Optimizer	size of training set
pilot model training	15	512	0.001	Adam	10%
target model training	100	1024	0.0001	Adam	80%
	MIND				
	#epoch	batch size	learning rate	Optimizer	size of training set
pilot model training	80	512	0.001	Adam	10%
target model training	200	1024	0.0001	Adam	80%

Table 5: Training details of the two datasets

A IMPLEMENTATION DETAILS

A.1 Dataset pre-processing

Here we demonstrate the data pre-processing of the two datasets. The KuaiRec dataset contains a sparse interaction matrix and a dense interaction matrix, where the dense interaction submatrix of the sparse matrix. The sparse matrix contains 7,176 users and 10,728 items, while the dense matrix contains 1,411 users and 3,327 items. As mentioned, the dense matrix is not collected under the natural recommendation setting, and shaving off the submatrix from the whole matrix is also unrealistic. Specifically, we create a submatrix by removing all the user columns from the sparse matrix if they appear in the dense matrix. The origin MIND-small dataset contains a training set and a validation set. In our setting, we first merge the two sets and re-split them into training, validation, and testing set. For both datasets, user-item pairs might appear multiple times in the dataset under a different context. When constructing the user-item bipartite graph, we treat them as *one* edge with a conductance of 1 if any of the pairs is positive and 0 otherwise. We assign all duplicate pairs with the same subsampling rate.

A.2 Model specifications and hyperparameters

For pilot models, we adopt the default configuration in TorchFM[2] except for D&W. The training details of the pilot and the target models are shown in table 5. To compute effective conductance over the graph, we run random walk simulation until the update of commute time is smaller than 0.1. In edge propagation, the best results are reported with $\gamma = 0.2$ and $\gamma = 0.4$ for Opt-Sampling and MA-EC, respectively, in the KuaiRec dataset. And $\gamma = 0.05$ for both methods in the MIND dataset.

A.3 Control experiment

We illustrate the control experiment in Figure 9. In each setting, one set of the scores is used as the major sampling scores, and only part of the instances will use the other ones. For example, when Opt-Sampling is used as major scores (red rectangle), instances whose scores lie in the upper left (blue rectangle) have inconsistent hardness over the two methods. And since MA-EC considers them hard negative samples, we “flip” the subsampling score of those instances from Opt-Sampling to MA-EC.

[2]See https://github.com/rixwew/pytorch-fm

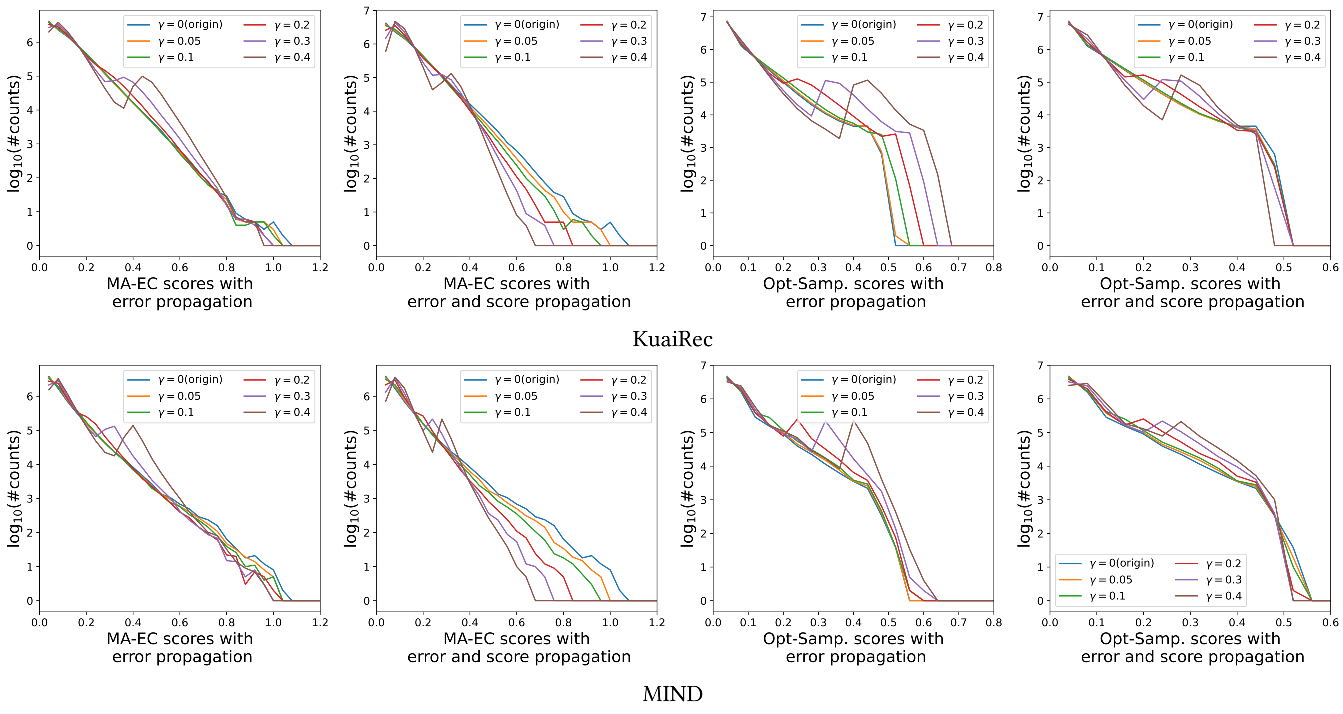


Figure 10: Distribution of propagated scores.

A.4 Computing resources

For all the experiments, we train the models on NVIDIA A100 and NVIDIA V100 GPUs. For estimating the effective conductance on the graph, we utilize 16 CPU cores and 40 gigabytes of memory to run the simulations for both datasets. molecule generation tasks, the inference time of each model is measured on 1 TITAN RTX GPU and 20 CPU cores.

B ADDITIONAL RESULTS

B.1 Model performance on prediction tasks

We further investigate the performance gain of MA-EC in other offline metrics. Specifically, we consider Normalized Discounted Cumulative Gain (NDCG) scores and adaptive calibration error (ACE). We consider uniform sampling, Opt-Sampling, and MA-EC on the KuaiRec dataset in Table 6. We see that MA-EC and Opt-Sampling outperform uniform in all metrics by around 0.015 in terms of NDCG, illustrating the benefit of non-uniform sampling. As ACE measures whether the probabilities predicted by the classifier are calibrated [23], we can see that our model is well-calibrated even though the subsampled training set has a different population than the actual training set.

	Uniform	Opt-Samp.	MA-EC
ACE↓	0.0070±0.0005	0.0073±0.0010	0.0071±0.0014
NDCG@5↑	0.5209±0.0034	0.5342±0.0021	0.5390±0.0034
NDCG@10↑	0.5223±0.0019	0.5375±0.0018	0.5403±0.0023
NDCG@30↑	0.5221±0.0022	0.5364±0.0026	0.5397±0.0030

Table 6: Model performance on offline metrics

B.2 Smoothness of propagated scores

For both datasets, we visualize the distributions of the corrected hardness scores for both MA-EC and Opt-Sampling methods. Specifically, we divide the instance scores into 50 bins and count the number of instances (both positives and negatives) for each bin. We generate the histograms and visualize them in Figure 10.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009