On Sampling Collaborative Filtering Datasets

Noveen Sachdeva
nosachde@ucsd.edu
University of California, San Diego
La Jolla, CA, USA

Carole-Jean Wu
carolejeanwu@fb.com
Facebook AI Research
Cambridge, MA, USA

Julian McAuley
jmcauley@ucsd.edu
University of California, San Diego
La Jolla, CA, USA

Abstract

We study the practical consequences of dataset sampling strategies on the ranking performance of recommendation algorithms. Recommender systems are generally trained and evaluated on *samples* of larger datasets. Samples are often taken in a naïve or ad-hoc fashion: *e.g.* by sampling a dataset randomly or by selecting users or items with many interactions. As we demonstrate, commonly-used data sampling schemes can have significant consequences on algorithm performance. Following this observation, this paper makes three main contributions: (1) *characterizing* the effect of sampling on algorithm performance, in terms of algorithm and dataset characteristics (*e.g.* sparsity characteristics, sequential dynamics, *etc.*); (2) designing SVP-CF, which is a data-specific sampling strategy, that aims to preserve the relative performance of models after sampling, and is especially suited to long-tailed interaction data; and (3) developing an *oracle*, Data-genie, which can suggest the sampling scheme that is most likely to preserve model performance for a given dataset. The main benefit of Data-genie is that it will allow recommender system practitioners to quickly prototype and compare various approaches, while remaining confident that algorithm performance will be preserved, once the algorithm is retrained and deployed on the complete data. Detailed experiments show that using Data-genie, we can discard upto 5× more data than any sampling strategy with the same level of performance.

CCS Concepts

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Feature selection.*

Keywords

Sampling; Coreset Mining; Benchmarking; Large-scale Learning

ACM Reference Format:
Noveen Sachdeva, Carole-Jean Wu, and Julian McAuley. 2022. On Sampling Collaborative Filtering Datasets. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498439

1 Introduction

Representative *sampling* of collaborative filtering (CF) data is a crucial problem from numerous stand-points and can be performed at various levels: (1) mining hard-negatives while training complex algorithms over massive datasets [5, 31]; (2) down-sampling the item-space to estimate expensive ranking metrics [3, 22]; and (3) reasons like easy-sharing, fast-experimentation, mitigating the significant environmental footprint of training resource-hungry machine learning models [11, 37, 43, 50]. In this paper, we are interested in finding a sub-sample of a dataset which has minimal effects on model utility evaluation *i.e.* an algorithm performing well on the sub-sample should also perform well on the original dataset.

Preserving *exactly* the same levels of performance on sub-sampled data over metrics, such as MSE and AUC, is a challenging problem. A simpler yet useful problem is accurately preserving the *ranking* or relative performance of different algorithms on sub-sampled data. For example, a sampling scheme that has low bias but high variance in preserving metric performance values has less utility than a different sampling scheme with high amounts of bias but low variance, since the overall algorithm ranking is still preserved.

Performing ad-hoc sampling such as randomly removing interactions, or making dense subsets by removing users *or* items with few interactions [40] can have adverse downstream repercussions. For example, sampling only the head-portion of a dataset—from a fairness and inclusion perspective—is inherently biased against minority-groups and benchmarking algorithms on this biased data is likely to propagate the original sampling bias. Alternatively, simply from a model performance view-point, accurately retaining the relative performance of different recommendation algorithms on much smaller sub-samples is a challenging research problem in itself.

Two prominent directions toward better and more representative sampling of CF data are: (1) designing principled sampling strategies, especially for user-item interaction data; and (2) analyzing the performance of different sampling strategies, in order to better grasp which sampling scheme performs "better" for which types of data. *In this paper,* we explore both directions through the lens of expediting the recommendation algorithm development cycle by:

- Characterizing the efficacy of *sixteen* different sampling strategies in accurately benchmarking various kinds of recommendation algorithms on smaller sub-samples.
- Proposing a sampling strategy, SVP-CF, which dynamically samples the "toughest" portion of a CF dataset. SVP-CF is tailor-designed to handle the inherent data heterogeneity and missing-not-at-random properties in user-item interaction data.
- Developing Data-genie, which analyzes the *performance* of different sampling strategies. Given a dataset sub-sample, Data-genie can directly estimate the likelihood of model performance being preserved on that sample.

Ultimately, our experiments reveal that SVP-CF outperforms all other sampling strategies and can accurately benchmark recommendation algorithms with roughly 50 – 60% of the original dataset

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02.
https://doi.org/10.1145/3488560.3498439

size. Furthermore, by employing DATA-GENIE to dynamically select the best sampling scheme for a dataset, we are able to preserve model performance with only 10% of the initial data, leading to a net 5.8× training time speedup.

2 Related Work

Sampling CF data. Sampling in CF-data has been a popular choice for three major scenarios. Most prominently, sampling is used for mining hard-negatives while training recommendation algorithms. Some popular approaches include random sampling; using the graph-structure [31, 52]; and ad-hoc techniques like similarity search [14], stratified sampling [5], *etc.* Sampling is also generally employed for evaluating recommendation algorithms by estimating expensive to compute metrics like Recall, nDCG, *etc.* [3, 22]. Finally, sampling is also used to create smaller sub-samples of a big dataset for reasons like fast experimentation, benchmarking different algorithms, privacy concerns, *etc.* However, the consequences of different samplers on any of these downstream applications is under-studied, and is the main research interest of this paper.

Coreset selection. Closest to our work, a coreset is loosely defined as a subset of data-points that maintains a similar "quality" as the full dataset for subsequent model training. Submodular approaches try to optimize a function $f : \mathbf{V} \mapsto \mathcal{R}_+$ which measures the utility of a subset $\mathbf{V} \subseteq \mathbf{X}$, and use it as a proxy to select the best coreset [17]. More recent works treat coreset selection as a bi-level optimization problem [2, 21] and directly optimize for the best coreset for a given downstream task. Selection-via-proxy [7] is another technique which employs a base-model as a proxy to tag the importance of each data-point. Note, however, that most existing coreset selection approaches were designed primarily for classification data, whereas adapting them for CF-data is non-trivial because of: (1) the inherent data heterogeneity; the (2) wide range of recommendation metrics; and (3) the prevalent missing-data characteristics.

Evaluating sample quality. The quality of a dataset sample, if estimated correctly is of high interest for various applications. Short of being able to evaluate the "true" utility of a sample, one generally resorts to either retaining task-dependent characteristics [32] *or* employing universal, handcrafted features like a social network's hop distribution, eigenvalue distribution, *etc.* [24] as meaningful proxies. Note that evaluating the sample quality with a limited set of handcrafted features might introduce bias in the sampled data, depending on the number and quality of such features.

3 Sampling Collaborative Filtering Datasets

Given our motivation of quickly benchmarking recommendation algorithms, we now aim to *characterize* the performance of various commonly-used sampling strategies. We loosely define the performance of a sampling scheme as its ability in effectively retaining the performance-ranking of different recommendation algorithms on the full *vs.* sub-sampled data. In this section, we start by discussing the different recommendation feedback scenarios we consider, along with a representative sample of popular recommendation algorithms that we aim to efficiently benchmark. We then examine popular data sampling strategies, followed by proposing a novel, proxy-based sampling strategy (SVP-CF) that is especially suited for sampling representative subsets from long-tail CF data.

3.1 Problem Settings & Methods Compared

To give a representative sample of typical recommendation scenarios, we consider three different user feedback settings. In *explicit feedback*, each user u gives a numerical rating r_i^u to each interacted item i; the model must predict these ratings for novel (test) user-item interactions. Models from this class are evaluated in terms of the Mean Squared Error (MSE) of the predicted ratings. Another scenario we consider is *implicit feedback*, where the interactions for each user are only available for positive items (*e.g.* clicks or purchases), whilst all non-interacted items are considered as negatives. We employ the AUC, Recall@100, and nDCG@10 metrics to evaluate model performance for implicit feedback algorithms. Finally, we also consider *sequential feedback*, where each user u interacts with an ordered sequence of items $\mathcal{S}^u = (\mathcal{S}_1^u, \mathcal{S}_2^u, \ldots, \mathcal{S}_{|\mathcal{S}^u|}^u)$ such that $\mathcal{S}_i^u \in \mathcal{I}$ for all $i \in \{1, \ldots, |\mathcal{S}^u|\}$. Given $\mathcal{S} = \{\mathcal{S}^u \mid \forall u \in \mathcal{U}\}$, the goal is to identify the *next-item* for each sequence $\mathcal{S}^u$ that each user u is most likely to interact with. We use the same metrics as in implicit feedback settings. Note that following recent warnings against sampled metrics for evaluating recommendation algorithms [3, 22], we compute both Recall and nDCG by ranking *all* items in the dataset. Further specifics about the datasets used, pre-processing, train/test splits, *etc.* are discussed in-depth in Section 3.5.

Given the diversity of the scenarios discussed above, there are numerous relevant recommendation algorithms. We use the following seven recommendation algorithms, intended to represent the state-of-the-art and standard baselines:

- *PopRec:* A naïve baseline that simply ranks items according to overall train-set popularity. Note that this method is unaffected by the user for which items are being recommended, and has the *same global ranking* of all items.
- *Bias-only:* Another simple baseline that assumes no interactions between users and items. Formally, it learns: (1) a global bias α; (2) scalar biases β_u for each user $u \in \mathcal{U}$; and (3) scalar biases β_i for each item $i \in \mathcal{I}$. Ultimately, the rating/relevance for user u and item i is modeled as $\hat{r}_i^u = \alpha + \beta_u + \beta_i$.
- *Matrix Factorization (MF) [20]:* Represents both users and items in a common, low-dimensional latent-space by factorizing the user-item interaction matrix. Formally, the rating/relevance for user u and item i is modeled as $\hat{r}_i^u = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$ where $\gamma_u, \gamma_i \in \mathbb{R}^d$ are learned latent representations.
- *Neural Matrix Factorization (NeuMF) [13]:* Leverages the representation power of deep neural-networks to capture non-linear correlations between user and item embeddings. Formally, the rating/relevance for user u and item i is modeled as $\hat{r}_i^u = \alpha + \beta_u + \beta_i + f(\gamma_u \,||\, \gamma_i \,||\, \gamma_u \cdot \gamma_i)$ where $\gamma_u, \gamma_i \in \mathbb{R}^d$, '||' represents the concatenation operation, and $f : \mathbb{R}^{3d} \mapsto \mathbb{R}$ represents an arbitrarily complex neural network.
- *Variational Auto-Encoders for Collaborative Filtering (MVAE) [26]:* Builds upon the Variational Auto-Encoder (VAE) [18] framework to learn a low-dimensional representation of a user's consumption history. More specifically, MVAE encodes each

user's bag-of-words consumption history using a VAE and further decodes the latent representation to obtain the completed user preference over all items.

- *Sequential Variational Auto-Encoders for Collaborative Filtering (SVAE) [39]:* A sequential algorithm that combines the temporal modeling capabilities of a GRU [6] along with the representation power of VAEs. Unlike MVAE, SVAE uses a GRU to encode the user's consumption sequence followed by a multinomial VAE at each time-step to model the likelihood of the next item.
- *Self-attentive Sequential Recommendation (SASRec) [16]:* Another sequential algorithm that relies on the sequence modeling capabilities of self-attentive neural networks [48] to predict the occurance of the next item in a user's consumption sequence. To be precise, given a user u and their time-ordered consumption history $\mathcal{S}^u = (\mathcal{S}^u_1, \mathcal{S}^u_2, \ldots, \mathcal{S}^u_{|\mathcal{S}^u|})$, SASRec first applies self-attention on $\mathcal{S}^u$ followed by a series of non-linear feed-forward layers to finally obtain the next item likelihood.

We also list models and metrics for each of the three different CF-scenarios in Table 1. Since bias-only, MF, and NeuMF can be trained for all three CF-scenarios, we optimize them using the regularized least-squares regression loss for explicit feedback, and the pairwise-ranking (BPR [38]) loss for implicit/sequential feedback. Note however that the aforementioned algorithms are only intended to be a representative sample of a wide pool of recommendation algorithms, and in our pursuit to benchmark recommender systems faster, we are primarily concerned with the *ranking* of different algorithms on the full dataset *vs.* a smaller sub-sample.

3.2 Sampling Strategies

Given a user-item CF dataset $\mathcal{D}$, we aim to create a $p\%$ subset $\mathcal{D}^{s,p}$ according to some sampling strategy s. In this paper, to be comprehensive, we consider a sample of eight popular sampling strategies, which can be grouped into the following three categories:

3.2.1 Interaction sampling. We first discuss three strategies that sample interactions from $\mathcal{D}$. In *Random Interaction Sampling*, we generate $\mathcal{D}^{s,p}$ by randomly sampling $p\%$ of all the user-item interactions in $\mathcal{D}$. *User-history Stratified Sampling* is another popular sampling technique (see *e.g.* [39, 51]) to generate smaller CF-datasets. To match the user-frequency distribution amongst $\mathcal{D}$ and $\mathcal{D}^{s,p}$, it randomly samples $p\%$ of interactions from each user's consumption history. Unlike random stratified sampling, *User-history Temporal Sampling* samples $p\%$ of the *most recent* interactions for each user. This strategy is representative of the popular practice of making data subsets from the online traffic of the last x days [15, 31].

3.2.2 User sampling. Similar to sampling interactions, we also consider two strategies which sample users in $\mathcal{D}$ instead. To ensure a fair comparison amongst the different kinds of sampling schemes used in this paper, we retain exactly $p\%$ of the *total interactions* in $\mathcal{D}^{s,p}$. In *Random User Sampling*, we retain users from $\mathcal{D}$ at random. To be more specific, we iteratively preserve *all* the interactions for a random user until we have retained $p\%$ of the original interactions. Another strategy we employ is *Head User Sampling*, in which we iteratively remove the user with the least amount of total interactions. This method is representative of commonly used data pre-processing strategies (see *e.g.* [13, 26]) to make data suitable for parameter-heavy algorithms. Sampling the data in such a way can introduce bias toward users from minority groups which might raise concerns from a diversity and fairness perspective [30].

3.2.3 Graph sampling. Instead of sampling directly from $\mathcal{D}$, we also consider three strategies that sample from the inherent user-item bipartite interaction graph $\mathcal{G}$. In *Centrality-based Sampling*, we proceed by computing the pagerank centrality scores [35] for each node in $\mathcal{G}$, and retain all the edges (interactions) of the *top scoring nodes* until a total $p\%$ of the original interactions have been preserved. Another popular strategy we employ is *Random-walk Sampling* [24], which performs multiple random-walks with restart on $\mathcal{G}$ and retains the edges amongst those pairs of nodes that have been visited at least once. We keep expanding our walk until $p\%$ of the initial edges have been retained. We also utilize *Forest-fire Sampling* [25], which is a snowball sampling method and proceeds by randomly "burning" the outgoing edges of visited nodes. It initially starts with a random node, and then propagates to a random subset of previously unvisited neighbors. The propagation is terminated once we have created a graph-subset with $p\%$ of the initial edges.

3.3 SVP-CF: Selection-Via-Proxy for CF data

Selection-Via-Proxy (SVP) [7] is a leading coreset mining technique for classification datasets like CIFAR10 [23] and ImageNet [9]. The main idea proposed is simple and effective, and proceeds by training a relatively inexpensive base-model as a proxy to define the "importance" of a data-point. However, applying SVP to CF-data can be highly non-trivial because of the following impediments:

- *Data heterogeneity:* Unlike classification data over some input space $\mathcal{X}$ and label-space $\mathcal{Y}$, CF-data consists of numerous four-tuples $\{u, i, r^u_i, t^u_i\}$. Such multimodal data adds many different dimensions to sample data from, making it increasingly complex to define meaningful samplers.
- *Defining the importance of a data point:* Unlike classification, where we can measure the performance of a classifier by its empirical risk on held-out data, for recommendation, there are a variety of different scenarios (Section 3.1) along with a wide list of relevant evaluation metrics. Hence, it becomes challenging to adapt importance-tagging techniques like greedy k-centers [44], forgetting-events [47], *etc.* for recommendation tasks.
- *Missing data:* CF-data is well-known for (1) its sparsity; (2) skewed and long-tail user/item distributions; and (3) missing-not-at-random (MNAR) properties of the user-item interaction matrix. This results in additional problems as we are now sampling data from skewed, MNAR data, especially using proxy-models trained on the same skewed data. Such sampling in the worst-case might even lead to exacerbating existing biases in the data or even aberrant data samples.

To address these fundamental limitations in applying the SVP philosophy to CF-data, we propose SVP-CF to sample representative subsets from large user-item interaction data. SVP-CF is also specifically devised for our objective of benchmarking different recommendation algorithms, as it relies on the crucial assumption that the "easiest" part of a dataset will generally be easy *for all* algorithms. Under this assumption, even after removing such data we are still likely to retain the overall algorithms' ranking.

CF-scenario	*Algorithm*							*Metric*			
	Bias-only	MF	NeuMF	PopRec	MVAE	SVAE	SASRec	MSE	AUC	Recall@k	nDCG@k
Explicit	Yes	Yes	Yes	×	×	×	×	Yes	×	×	×
Implicit	Yes	Yes	Yes	Yes	Yes	×	×	×	Yes	Yes	Yes
Sequential	Yes	Yes	Yes	Yes	Yes	Yes	Yes	×	Yes	Yes	Yes

Table 1: Demonstrates the pertinence of each CF-scenario towards each algorithm (left) and each metric (right). Note that we can use ranking metrics for explicit feedback, however, we only use MSE as a design choice and due to it's direct relevance.

Because of the inherent data heterogeneity in user-item interaction data, we can sub-sample in a variety of different ways. We design SVP-CF to be versatile in this aspect as it can be applied to sample users, items, interactions, or combinations of them, by marginally adjusting the definition of importance of each datapoint. In this paper, we limit the discussion to only sampling users and interactions (separately), but extending SVP-CF for sampling across other data modalities should be relatively straightforward.

Irrespective of whether to sample users or interactions, SVP-CF proceeds by training an inexpensive proxy model $\mathcal{P}$ on the full, original data $\mathcal{D}$ and modifies the forgetting-events approach [47] to retain the points with the *highest* importance. To be more specific, for explicit feedback, we define the importance of each data-point *i.e.* $\{u, i, r_i^u, t_i^u\}$ interaction as $\mathcal{P}$'s average MSE (over epochs) of the specific interaction if we're sampling interactions *or* $\mathcal{P}$'s average MSE of u (over epochs) if we're sampling users. Whereas, for implicit and sequential feedback, we use $\mathcal{P}$'s average inverse-AUC while computing the importance of each data-point. For the sake of completeness, we experiment with both Bias-only and MF as two different kinds of proxy-models for SVP-CF. Since both models can be trained for all three CF-scenarios (Table 1), we can directly use them to tag the importance for each CF-scenario.

Ultimately, to handle the MNAR and long-tail problems, we also propose SVP-CF-Prop which employs user and item propensities to correct the distribution mismatch while estimating the importance of each datapoint. More specifically, let $p_{u,i} = P(r_i^u = 1 \mid \overset{*}{r}{}_i^u = 1)$ denote the probability of user u and item i's interaction actually being observed (propensity), E be the total number of epochs that $\mathcal{P}$ was trained for, $\mathcal{P}_e$ denote the proxy model after the e^{th} epoch, $\mathcal{I}_u^+ := \{j \mid r_j^u > 0\}$ be the set of positive interactions for u, and $\mathcal{I}_u^- := \{j \mid r_j^u = 0\}$ be the set of negative interactions for u; then, the importance function for SVP-CF-Prop, $\mathcal{I}_p$ is defined as follows:

$$\mathcal{I}_p(u \mid \mathcal{P}) := \frac{1}{|\mathcal{I}_u^+|} \cdot \sum_{i \in \mathcal{I}_u^+} \mathcal{I}_p(u, i \mid \mathcal{P}) \quad ; \quad \mathcal{I}_p(u, i \mid \mathcal{P}) := \frac{\Delta(u, i \mid \mathcal{P})}{p_{u,i}}$$

$$\text{where,} \qquad \Delta(u, i \mid \mathcal{P}) := \begin{cases} \sum_{e=1}^{E} \left(\mathcal{P}_e(u, i) - r_i^u\right)^2 \\ \text{(for explicit feedback)} \\ \\ \sum_{e=1}^{E} \sum_{j \sim \mathcal{I}_u^-} \frac{1}{\mathbb{1}\left(\mathcal{P}_e(u, i) > \mathcal{P}_e(u, j)\right)} \\ \text{(for implicit/sequential feedback)} \end{cases}$$

Proposition 3.1. *Given an ideal propensity-model $p_{u,i}$; $\mathcal{I}_p(u, i \mid \mathcal{P})$ is an unbiased estimator of $\Delta(u, i \mid \mathcal{P})$.*

Proof.

$$\begin{aligned}
&\mathop{\mathbb{E}}_{u \sim \mathcal{U}} \mathop{\mathbb{E}}_{i \sim \mathcal{I}} \left[\mathcal{I}_p(u, i \mid \mathcal{P})\right] \\
&= \frac{1}{|\mathcal{U}||\mathcal{I}|} \sum_{u \sim \mathcal{U}} \sum_{i \sim \mathcal{I}} \mathcal{I}_p(u, i \mid \mathcal{P}) \cdot P(r_i^u = 1) \\
&= \frac{1}{|\mathcal{U}||\mathcal{I}|} \sum_{u \sim \mathcal{U}} \sum_{i \sim \mathcal{I}} \frac{\Delta(u, i \mid \mathcal{P})}{p_{u,i}} \cdot \Big(P(\overset{*}{r}{}_i^u = 0) \cdot \cancelto{0}{P(r_i^u = 1 \mid \overset{*}{r}{}_i^u = 0)} \\
&\qquad\qquad + P(\overset{*}{r}{}_i^u = 1) \cdot P(r_i^u = 1 \mid \overset{*}{r}{}_i^u = 1)\Big) \\
&= \frac{1}{|\mathcal{U}||\mathcal{I}|} \sum_{u \sim \mathcal{U}} \sum_{i \sim \mathcal{I}} \Delta(u, i \mid \mathcal{P}) \cdot P(\overset{*}{r}{}_i^u = 1) \\
&= \mathop{\mathbb{E}}_{u \sim \mathcal{U}} \mathop{\mathbb{E}}_{i \sim \mathcal{I}} \left[\Delta(u, i \mid \mathcal{P})\right]
\end{aligned}$$

□

Propensity model. A wide variety of ways exist to model the propensity score of a user-item interaction [15, 41, 42, 54]. The most common ways comprise using machine learning models like naïve bayes and logistic regression [42], or by fitting handcrafted functions [15]. For our problem statement, we make a simplifying assumption that the data noise is one-sided *i.e.* $P(r_i^u = 1 \mid \overset{*}{r}{}_i^u = 0)$ or the probability of a user interacting with a *wrong* item is *zero*, and model the probability of an interaction going missing to decompose over the user and item as follows:

$$\begin{aligned}
p_{u,i} &= P(r_i^u = 1 \mid \overset{*}{r}{}_i^u = 1) \\
&= P(r^u = 1 \mid \overset{*}{r}{}^u = 1) \cdot P(r_i = 1 \mid \overset{*}{r}_i = 1) \;=\; p_u \cdot p_i
\end{aligned}$$

Ultimately, following [15], we assume the user and item propensities to lie on the following sigmoid curves:

$$p_u := \frac{1}{1 + C_u \cdot e^{-A \cdot log(N_u + B)}} \quad ; \quad p_i := \frac{1}{1 + C_i \cdot e^{-A \cdot log(N_i + B)}}$$

Where, N_u and N_i represent the total number of interactions of user u and item i respectively, A and B are two fixed scalars, $C_u = (log(|\mathcal{U}|) - 1) \cdot (B + 1)^A$ and $C_i = (log(|\mathcal{I}|) - 1) \cdot (B + 1)^A$.

3.4 Performance of a sampling strategy

To quantify the performance of a sampling strategy s on a dataset $\mathcal{D}$, we start by creating various $p\%$ subsets of $\mathcal{D}$ according to s and call them $\mathcal{D}^{s,p}$. Next, we train and evaluate all the relevant recommendation algorithms on both $\mathcal{D}$ and $\mathcal{D}^{s,p}$. Let the *ranking* of all algorithms according to CF-scenario f and metric m trained on $\mathcal{D}$ and $\mathcal{D}^{s,p}$ be $\mathcal{R}_{f,m}$ and $\mathcal{R}_{f,m}^{s,p}$ respectively, then the performance measure $\Psi(\mathcal{D}, s)$ is defined as the average correlation between

$\mathcal{R}_{f,m}$ and $\mathcal{R}^{s,p}_{f,m}$ measured through Kendall's Tau over all possible CF-scenarios, metrics, and sampling percents:

$$\Psi\left(\mathcal{D}, s\right) = \lambda \cdot \sum_{f} \sum_{m} \sum_{p} \tau\left(\mathcal{R}_{f,m}, \mathcal{R}^{s,p}_{f,m}\right)$$

Where λ is an appropriate normalizing constant for computing the average, sampling percent $p \in \{80, 60, 40, 20, 10, 1\}$, CF-scenario f, metric m and their pertinence towards each other can all be found in Table 1. Ψ has the same range as Kendall's Tau *i.e.* $[-1, 1]$ and a higher Ψ indicates strong agreement between the algorithm ranking on the full and sub-sampled datasets, whereas a large negative Ψ implies that the algorithm order was effectively reversed.

3.5 Experiments

Datasets. To promote dataset diversity in our experiments, we use six public user-item rating interaction datasets with varying sizes, sparsity patterns, and other characteristics. We use the Magazine, Luxury, and Video-games categories of the Amazon review dataset [33], along with the Movielens-100k [12], BeerAdvocate [28], and GoodReads Comics [49] datasets. A brief set of data statistics is also presented in Table 2. We simulate all three CF-scenarios (Section 3.1) via different pre-processing strategies. For explicit and implicit feedback, we follow a randomized 80/10/10 train-test-validation split for each user's consumption history in the dataset, and make use of the leave-one-last [29] strategy for sequential feedback *i.e.* keep the last two interactions in each user's time-sorted consumption history in the validation and test-set respectively. Since we can't control the initial construction of datasets, and to minimize the initial data bias, we follow the least restrictive data pre-processing [8, 40]. We only weed out the users with less than 3 interactions, to keep at least one occurrence in the train, validation, and test sets.

Dataset	# Interactions	# Users	# Items	Avg. User history length
Amazon Magazine	12.7k	3.1k	1.3k	4.1
ML-100k	100k	943	1.7k	106.04
Amazon Luxury	126k	29.7k	8.4k	4.26
Amazon Video-games	973k	181k	55.3k	5.37
BeerAdvocate	1.51M	18.6k	64.3k	81.45
Goodreads Comics	4.37M	133k	89k	32.72

Table 2: Data statistics of the *six* datasets used in this paper.

Training details. We implement all algorithms in PyTorch[1] and train on a single GPU. For a fair comparison across algorithms, we perform hyper-parameter search on the validation set. For the three smallest datasets used in this paper (Table 2), we search the latent size in $\{4, 8, 16, 32, 50\}$, dropout in $\{0.0, 0.3, 0.5\}$, and the learning rate in $\{0.001, 0.006, 0.02\}$. Whereas for the three largest datasets, we fix the learning rate to be 0.006. Note that despite the limited number of datasets and recommendation algorithms used in this study, given that we need to train all algorithms with hyper-parameter tuning for all CF scenarios, % data sampled according to all different sampling strategies discussed in Section 3.2, there are a total of $\sim 400k$ unique models trained, equating to a cumulative train time of over $400k \times \sim 1\text{min} \approx 9$ months.

[1]Code is available at https://github.com/noveens/sampling_cf

Data sampling. To compute the Ψ-values as defined in Section 3.4, we construct $\{80, 60, 40, 20, 10, 1\}$% samples for each dataset and sampling strategy. To keep comparisons as fair as possible, for all sampling schemes, we only sample on the train set and never touch the validation and test sets.

3.5.1 How do different sampling strategies compare to each other? Results with Ψ-values for all sampling schemes on all datasets are in Table 3. Even though there are only six datasets under consideration, there are a few prominent patterns. First, the average Ψ for most sampling schemes is around 0.4, which implies a statistically significant correlation between the ranking of algorithms on the full *vs.* sub-sampled datasets. Next, SVP-CF generally outperforms all commonly used sampling strategies by some margin in retaining the ranking of different recommendation algorithms. Finally, the methods that discard the tail of a dataset (head-user and centrality-based) are the worst performing strategies overall, which supports the recent warnings against dense sampling of data [40].

3.5.2 How does the relative performance of algorithms change as a function of sampling rate? In an attempt to better understand the impact of sampling on different recommendation algorithms used in this study (Section 3.1), we visualize the probability of a recommendation algorithm moving up in the overall method ranking with data sampling. We estimate the aforementioned probability using Maximum-Likelihood-Estimation (MLE) on the experiments already run in computing $\Psi(\mathcal{D}, s)$. Formally, given a recommendation algorithm r, CF-scenario f, and data sampling percent p:

$$P_{MLE}(r \mid f, p) = \lambda \cdot \sum_{\mathcal{D}} \sum_{s} \sum_{m} 0.5 + \frac{\mathcal{R}_{f,m}(r) - \mathcal{R}^{s,p}_{f,m}(r)}{2 \cdot (n-1)}$$

where λ is an appropriate normalizing constant, and n represents the total number of algorithms. A heatmap visualizing P_{MLE} for all algorithms and CF-scenarios is shown in Figure 1. We see that simpler methods like Bias-only and PopRec have the highest probability across data scenarios of increasing their ranking order with extreme sampling. Whereas parameter-heavy algorithms like SASRec, SVAE, MVAE, *etc.* tend to decrease in the ranking order, which is indicative of overfitting on smaller data samples.

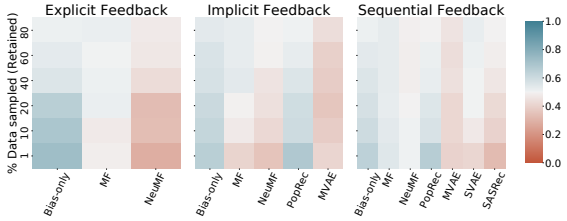

Figure 1: Heatmap of the probability of an algorithm moving in the overall ranking with extreme sampling. A high value indicates that the algorithm is most probable to *move up* in the sampled data ranking order, whereas a low value indicates that the algorithm is most probable to *move down*.

3.5.3 How much data to sample? Since Ψ is averaged over all $p \in \{80, 60, 40, 20, 10, 1\}$% data samples, to better estimate a reasonable amount of data to sample, we stratify Ψ according to each value of p and note the average Kendall's Tau. As we observe from Figure 2,

	Sampling strategy	Datasets: Amazon Magazine	ML-100k	Amazon Luxury	Amazon Video-games	BeerAdvocate	Goodreads Comics	*Average*
Interaction sampling	Random	0.428	0.551	0.409	0.047	0.455	0.552	0.407
	Stratified	0.27	0.499	0.291	-0.01	0.468	0.538	0.343
	Temporal	0.289	0.569	0.416	-0.02	0.539	0.634	0.405
	SVP-CF *w/* MF	0.418	0.674	0.398	0.326	0.425	0.662	0.484
	SVP-CF *w/* Bias-only	0.38	0.684	0.431	0.348	0.365	0.6	0.468
	SVP-CF-Prop *w/* MF	0.381	0.617	0.313	0.305	0.356	0.608	0.43
	SVP-CF-Prop *w/* Bias-only	0.408	0.617	0.351	0.316	0.437	0.617	0.458
User sampling	Random	0.436	0.622	0.429	0.17	0.344	0.582	0.431
	Head	0.369	0.403	0.315	0.11	-0.04	-0.02	0.19
	SVP-CF *w/* MF	0.468	0.578	0.308	0.13	0.136	0.444	0.344
	SVP-CF *w/* Bias-only	0.49	0.608	0.276	0.124	0.196	0.362	0.343
	SVP-CF-Prop *w/* MF	0.438	0.683	0.307	0.098	0.458	0.592	0.429
	SVP-CF-Prop *w/* Bias-only	0.434	0.751	0.233	0.107	0.506	0.637	0.445
Graph	Centrality	0.307	0.464	0.407	0.063	0.011	0.343	0.266
	Random-walk	0.596	0.5	0.395	0.306	0.137	0.442	0.396
	Forest-fire	0.564	0.493	0.415	0.265	0.099	0.454	0.382

Table 3: Ψ-values for all datasets and sampling strategies. Higher Ψ is better. The best Ψ for every dataset is underlined. The Ψ-values for each sampling scheme *averaged over all datasets* is appended to the right.

there is a steady increase in the performance measure when more data is retained. Next, despite the results in Figure 2 being averaged over *sixteen* sampling strategies, we still notice a significant amount of performance retained after sampling just 50 – 60% of the data.

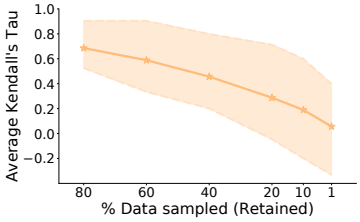

Figure 2: Comparison of the average Kendall's Tau with % data sampled. A higher Tau indicates better retaining power of the ranking of different recommendation algorithms.

3.5.4 Are different metrics affected equally by sampling? In an attempt to better understand how the different implicit and sequential feedback metrics (Section 3.1) are affected by sampling, we visualize the average Kendall's Tau for all sampling strategies (except SVP-CF for brevity) and all % data sampling choices separately over the AUC, Recall, and nDCG metrics in Figure 3. As expected, we observe a steady decrease in the model quality across the accuracy metrics over the different sampling schemes. This is in agreement with the analysis from Figure 2. Next, most sampling schemes follow a *similar* downwards trend in performance for the three metrics with AUC being slightly less affected and nDCG being slightly more affected by extreme sampling.

4 Data-genie: Which sampler is best for me?

Although the results presented in Section 3.5 are indicative of correlation between the ranking of recommendation algorithms on the

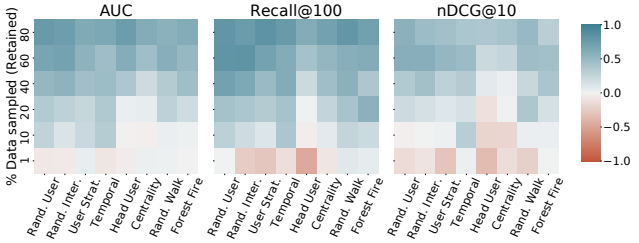

Figure 3: Heatmap of the average Kendall's Tau for different samplers stratified over metrics and % data sampled.

full dataset *vs.* smaller sub-samples, there still is no 'one-size-fits-all' solution to the question of *how to best sub-sample a dataset for retaining the performance of different recommendation algorithms?* In this section, we propose Data-genie, that attempts to answer this question from a statistical perspective, in contrast with existing literature that generally has to resort to sensible heuristics [1, 5, 24].

4.1 Problem formulation

Given a dataset $\mathcal{D}$, we aim to gauge how a *new* sampling strategy will perform in retaining the performance of different recommendation algorithms. Having already experimented with sixteen different sampling strategies on six datasets (Section 3.5), we take a frequentist approach in predicting the performance of any sampling scheme. To be precise, to predict the performance of sampling scheme s on dataset $\mathcal{D}$, we start by creating $\mathcal{D}$'s subset according to s and call it $\mathcal{D}^s$. We then represent $\mathcal{D}$ and $\mathcal{D}^s$ in a low-dimensional latent space, followed by a powerful regression model to directly estimate the performance of s on $\mathcal{D}$.

4.2 Dataset representation

We experiment with the following techniques of embedding a user-item interaction dataset into lower dimensions:

4.2.1 Handcrafted. For this method, we cherry-pick a few representative characteristics of $\mathcal{D}$ and the underlying user-item bipartite interaction graph $\mathcal{G}$. Inspired by prior work [24], we represent $\mathcal{D}$ as a combination of five features. We first utilize the frequency distribution of all users and items in $\mathcal{D}$. Next, we evaluate the distribution of the top−100 eigenvalues of $\mathcal{G}$'s adjacency matrix. All of these three distributions are generally long-tailed and heavily skewed. Furthermore, to capture notions like the diameter of $\mathcal{G}$, we compare the distribution of the number of hops h *vs.* the number of pairs of nodes in $\mathcal{G}$ reachable at a distance less than h [36]. This distribution, unlike others is monotonically increasing in h. Finally, we also compute the size distribution of all connected components in $\mathcal{G}$, where a connected component is defined to be the maximal set of nodes, such that a path exists between any pair of nodes. Ultimately, we ascertain $\mathcal{D}$'s final representation by concatenating 10 evenly-spaced samples from each of the aforementioned distributions along with the total number of users, items, and interactions in $\mathcal{D}$. This results in a 53−dimensional embedding for each dataset. Note that unlike previous work of simply *retaining* the discussed features as a proxy of the quality of data subsets [24], Data-genie instead uses these features to learn a regression model *on-top* which can dynamically establish the importance of each feature in the performance of a sampling strategy.

4.2.2 Unsupervised GCN. With the recent advancements in the field of Graph Convolutional Networks [19] to represent graph-structured data for a variety of downstream tasks, we also experiment with a GCN approach to embed $\mathcal{G}$. We modify the InfoGraph framework [46], which uses graph convolution encoders to obtain patch-level representations, followed by sort-pooling [53] to obtain a fixed, low-dimensional embedding for the entire graph. Since the nodes in $\mathcal{G}$ are the union of all users and items in $\mathcal{D}$, we randomly initialize 32−dimensional embeddings using a Xavier-uniform prior [10]. Parameter optimization is performed in an unsupervised fashion by maximizing the mutual information [34] amongst the graph-level and patch-level representations of nodes in the same graph. We validate the best values of the latent dimension and number of layers of the GCN from $\{4, 8, 16, 32\}$ and $\{1, 3\}$ respectively.

4.3 Training & Inference

Having discussed different representation functions $\mathcal{E} : \mathcal{D} \mapsto \mathbb{R}^d$ to embed a CF-dataset in Section 4.2, we now discuss Data-genie's training framework agnostic to the actual details about $\mathcal{E}$.

Optimization problem. As a proxy of the performance of a sampler on a given dataset, we re-use the Kendall's Tau for each CF-scenario, metric, and sampling percent used while computing the $\Psi(\mathcal{D}, s)$ in Section 3.5. To be specific, given $\mathcal{D}_f^{s,p}$ which is a p% sample of f−type feedback data $\mathcal{D}_f$, sampled according to sampling strategy s, we aim to estimate $\tau(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p})$ without ever computing the actual ranking of algorithms on either the full or sampled datasets:

$$\hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p}\right) = \Phi\left(\mathcal{E}(\mathcal{D}_f), \mathcal{E}(\mathcal{D}_f^{s,p}), m\right), \tag{1}$$

where Φ is an arbitrary neural network, and m is the metric of interest (see Table 1). We train Φ by either (1) regressing on the Kendall's Tau computed for each CF scenario, metric, and sampling percent used while computing the $\Psi(\mathcal{D}, s)$ scores in Section 3.5; or (2) performing BPR-style [38] pairwise ranking on two sampling schemes $s_i > s_j \iff \tau(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_i,p}) > \tau(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_j,p})$. Formally, the two optimization problems are defined as follows:

$$\arg\min_{\Phi} \sum_{\mathcal{D}_f} \sum_{s} \sum_{p} \sum_{m} \left(\tau\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p}\right) - \hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p}\right)\right)^2$$

(Data-genie-regression)

$$\arg\min_{\Phi} \sum_{\mathcal{D}_f} \sum_{s_i > s_j} \sum_{p} \sum_{m} -\ln \sigma\left(\hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_i,p}\right) - \hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_j,p}\right)\right)$$

(Data-genie-ranking)

$$\text{where, } \hat{\tau}\left(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p}\right) = \Phi\left(\mathcal{E}(\mathcal{D}_f), \mathcal{E}(\mathcal{D}_f^{s,p}), m\right).$$

The critical differences between the two aforementioned optimization problems are the downstream use-case and Φ's training time. If the utility of Data-genie is to rank different sampling schemes for a given dataset, then Data-genie-ranking is better suited as it is robust to the noise in computing $\tau(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s_i,p})$ like improper hyper-parameter tuning, local minima, *etc.* On the other hand, Data-genie-regression is better suited for the use-case of estimating the exact values of τ for a sampling scheme on a given dataset. Even though both optimization problems converge in less than 2 minutes given the data collected in Section 3.5, the complexity of optimizing Data-genie-ranking is still squared *w.r.t.* the total number of sampling schemes, whilst that of Data-genie-regression is linear.

Architecture. To compute $\Phi(\mathcal{E}(\mathcal{D}_f), \mathcal{E}(\mathcal{D}_f^{s,p}), m)$ we concatenate $\mathcal{E}(\mathcal{D}_f)$, $\mathcal{E}(\mathcal{D}_f^{s,p})$, one-hot embedding of m; and pass it through two relu-activated MLP projections to obtain $\hat{\tau}(\mathcal{R}_{f,m}, \mathcal{R}_{f,m}^{s,p})$. For Data-genie-regression, we also pass the final output through a tanh activation, to reflect the range of Kendall's Tau *i.e.* $[-1, 1]$.

Inference. Since computing both $\mathcal{E}$ and Φ are agnostic to the datasets and the sampling schemes, we can simply use the trained $\mathcal{E}$ and Φ functions to rank *any* sampling scheme for *any* CF dataset. Computationally, given a trained Data-genie, the utility of a sampling scheme can be computed simply by computing $\mathcal{E}$ twice, along with a single pass over Φ, completing in the order of milliseconds.

4.4 Experiments

Setup. We first create a train/validation/test split by randomly splitting all possible metrics and sampling % pairs (m, p) into 70/15/15% proportions. Subsequently for each dataset $\mathcal{D}$, CF-scenario f, and (m, p) in the validation/test-set, we ask Data-genie to rank all 16 samplers (Table 3) for p% sampling of f−type feedback for $\mathcal{D}$ and use metric m for evaluation by sorting $\hat{\tau}$ for each sampler, as defined in Equation (1). To evaluate Data-genie, we use the P@1 metric between the actual sampler ranking computed while computing Ψ−scores in Section 3.5, and the one estimated by Data-genie.

4.4.1 How accurately can Data-genie predict the best sampling scheme? In Table 4, we compare all dataset representation choices $\mathcal{E}$, and multiple Φ architectures for the task of predicting the best sampling strategy. In addition to the regression and ranking architectures discussed in Section 4.3, we also compare with linear least-squares regression and XGBoost regression [4] as other choices

$\mathcal{E}$	Φ	MSE	P@1
Random		0.2336	25.2
User sampling *w/* Bias-only SVP-CF-Prop		–	30.6
Handcrafted	Least squares regression	0.1866	31.7
”	XGBoost regression	0.1163	43.9
”	Data-genie-regression	<u>0.1008</u>	51.2
”	Data-genie-ranking	–	51.2
Unsupervised GCN	Least squares regression	0.1838	39.1
”	XGBoost regression	0.1231	43.9
”	Data-genie-regression	0.1293	48.8
”	Data-genie-ranking	–	<u>53.7</u>

Table 4: Results for predicting the best sampling scheme for a particular dataset over a germane metric. The MSE-value next to randomly choosing the sampling scheme represents the variance of the test-set. Best values are <u>underlined</u>.

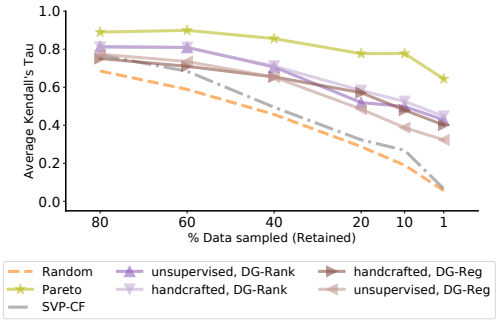

Figure 4: Comparison of the average Kendall's Tau with % data sampled for different sampling-selection strategies. A higher Tau indicates better retaining power of the ranking of different recommendation algorithms.

of Φ. In addition, we compare Data-genie with simple baselines: (1) randomly choosing a sampling strategy; and (2) the best possible static sampler choosing strategy—always predict user sampling *w/* Bias-only SVP-CF-Prop. First and foremost, irrespective of the $\mathcal{E}$ and Φ choices, Data-genie outperforms both baselines. Next, both the handcrafted features and the unsupervised GCN features perform quite well in predicting the best sampling strategy, indicating that the graph characteristics are well correlated with the final performance of a sampling strategy. Finally, Data-genie-regression and Data-genie-ranking both perform better than alternative Φ–choices, especially for the P@1 metric.

4.4.2 Can we use Data-genie to sample more data without compromising performance? In Figure 4, we compare the impact of Data-genie in sampling more data by dynamically choosing an appropriate sampler for a given dataset, metric, and % data to sample. More specifically, we compare the percentage of data sampled with the Kendall's Tau averaged over all datasets, CF-scenarios, and relevant metrics for different sampling strategy selection approaches. We compare Data-genie with: (1) randomly picking a sampling strategy averaged over 100 runs; and (2) the Pareto frontier as a skyline which always selects the best sampling strategy for any CF-dataset. As we observe from Figure 4, Data-genie is better than predicting a sampling scheme at random, and is much closer to the Pareto frontier. Next, pairwise ranking approaches are marginally better than regression approaches irrespective of $\mathcal{E}$. Finally, Data-genie can appraise the best-performing recommendation algorithm with a suitable amount of confidence using only 10% of the original data. This is significantly more efficient compared to having to sample 50 – 60% if we were to always sample using a fixed strategy.

5 Discussion

In this work, we discussed two approaches for representative sampling of CF-data, especially for accurately retaining the *relative* performance of different recommendation algorithms. First, we proposed SVP-CF, which is better than commonly used sampling strategies, followed by introducing Data-genie which *analyzes* the performance of different samplers on different datasets. Detailed experiments suggest that Data-genie can confidently estimate the downstream utility of any sampler within a few milliseconds, thereby enabling practitioners to benchmark different algorithms on 10% data sub-samples, with an average 5.8× time speedup.

To realize the real-world environmental impact of Data-genie, we discuss a typical weekly RecSys development cycle and its carbon footprint. Taking the Criteo Ad dataset as inspiration, we assume a common industry-scale dataset to have ∼ 4B interactions. We assume a hypothetical use case that benchmarks for *e.g.* 25 different algorithms, each with 40 different hyper-parameter variations. To estimate the energy consumption of GPUs, we scale the 0.4 minute MLPerf [27] run of training NeuMF [13] on the Movielens-20M dataset over an Nvidia DGX-2 machine. The total estimated run-time for all experiments would be $25 \times 40 \times \frac{4B}{20M} \times \frac{0.4}{60} \approx 1340$ hours; and following [45], the net CO_2 emissions would roughly be $10 \times 1340 \times 1.58 \times 0.954 \approx 20k$ lbs. To better understand the significance of this number, a brief CO_2 emissions comparison is presented in Table 5. Clearly, Data-genie along with saving a large amount of experimentation time and cloud compute cost, can also significantly reduce the carbon footprint of this *weekly process* by more than an average human's *yearly* CO_2 emissions.

Consumption	CO_2e (lbs.)
1 person, NY↔SF flight	2k
Human life, 1 year avg.	11k
Weekly RecSys development cycle	20k
” *w/* Data-genie	3.4k

Table 5: CO_2 emissions comparison [45]

Despite having significantly benefited the run-time and environmental impacts of benchmarking algorithms on massive datasets, our analysis heavily relied on the experiments of training seven recommendation algorithms on six datasets and their various samples. Despite the already large experimental cost, we strongly believe that the downstream performance of Data-genie could be further improved by simply considering more algorithms and diverse datasets. In addition to better sampling, analyzing the fairness aspects of training algorithms on sub-sampled datasets is an interesting research direction, which we plan to explore in future work.

Acknowledgments

This work was partly supported by NSF Award #1750063.

References

[1] Francois Belletti, Karthik Lakshmanan, Walid Krichene, Nicolas Mayoraz, Yi-Fan Chen, John Anderson, Taylor Robie, Tayo Oguntebi, Dan Shirron, and Amit Bleiwess. 2019. Scaling Up Collaborative Filtering Data Sets through Randomized Fractal Expansions. arXiv:1905.09874 [cs.LG]

[2] Zalán Borsos, Mojmir Mutny, and Andreas Krause. 2020. Coresets via Bilevel Optimization for Continual Learning and Streaming. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc.

[3] R. Cañamares and P. Castells. 2020. On Target Item Sampling In Offline Recommender System Evaluation. In *14th ACM Conference on Recommender Systems*.

[4] Tianqi Chen and Carlos Guestrin. [n.d.]. XGBoost: A Scalable Tree Boosting System. In *KDD '16*.

[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 Conference on Knowledge Discovery and Data Mining (KDD '17)*.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:1412.3555 [cs.NE]

[7] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2020. Selection via Proxy: Efficient Data Selection for Deep Learning. In *ICLR*.

[8] M. Dacrema, P. Cremonesi, and D. Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*.

[10] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

[11] U. Gupta, Y. Kim, S. Lee, J. Tse, H. S. Lee, G. Wei, D. Brooks, and C. Wu. 2021. Chasing Carbon: The Elusive Environmental Footprint of Computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*.

[12] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* (2015).

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*.

[14] H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. 2019. Slice: Scalable Linear Extreme Classifiers Trained on 100 Million Labels for Related Searches. In *Proceedings of the 12th ACM Conference on Web Search and Data Mining*.

[15] H. Jain, Y. Prabhu, and M. Varma. 2016. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking and Other Missing Label Applications. In *Proceedings of the SIGKDD Conference on Knowledge Discovery and Data Mining*.

[16] W. Kang and J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining*.

[17] V. Kaushal, R. Iyer, S. Kothawade, R. Mahadev, K. Doctor, and G. Ramakrishnan. 2019. Learning From Less Data: A Unified Data Subset Selection and Active Learning Framework for Computer Vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*.

[18] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014*. arXiv:http://arxiv.org/abs/1312.6114v10 [stat.ML]

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009).

[21] Andreas Krause, Marco Tagliasacchi, and Zalán Borsos. 2021. Semi-supervised batch active learning via bilevel optimization. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing*.

[22] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*.

[23] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[24] Jure Leskovec and Christos Faloutsos. 2006. Sampling from Large Graphs. In *KDD '06*.

[25] J. Leskovec, J. Kleinberg, and C. Faloutsos. 2005. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *Proceedings of the 11th ACM SIGKDD Conference on Knowledge Discovery in Data Mining (KDD '05)*.

[26] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*.

[27] P. Mattson, C. Cheng, G. Diamos, C. Coleman, P. Micikevicius, D. Patterson, H. Tang, G. Wei, P. Bailis, V. Bittorf, D. Brooks, D. Chen, D. Dutta, U. Gupta, K. Hazelwood, A. Hock, X. Huang, D. Kang, D. Kanter, N. Kumar, J. Liao, D. Narayanan, T. Oguntebi, G. Pekhimenko, L. Pentecost, Vijay Janapa, R., T. Robie, T. St John, C. Wu, L. Xu, C. Young, and M. Zaharia. 2020. MLPerf Training Benchmark. In *Proceedings of Machine Learning and Systems*.

[28] Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning Attitudes and Attributes from Multi-Aspect Reviews. In *ICDM '12*.

[29] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2020. Exploring Data Splitting Strategies for the Evaluation of Recommendation Models. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20)*.

[30] Shira Mitchell, Eric Potash, Solon Barocas, Alexander D'Amour, and Kristian Lum. 2018. Prediction-based decisions and fairness: A catalogue of choices, assumptions, and definitions. *arXiv preprint arXiv:1811.07867* (2018).

[31] A. Mittal, N. Sachdeva, S. Agrawal, S. Agarwal, P. Kar, and M. Varma. 2021. ECLARE: Extreme classification with label graph correlations. In *Proceedings of The ACM International World Wide Web Conference*.

[32] Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen Carley. 2013. Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose. *Proceedings of the International AAAI Conference on Web and Social Media* 7, 1 (Jun. 2013). https://ojs.aaai.org/index.php/ICWSM/article/view/14401

[33] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[34] S. Nowozin, B. Cseke, and R. Tomioka. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *NeurIPS*.

[35] L. Page, S. Brin, R. Motwani, and T. Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[36] Christopher R. Palmer, Phillip B. Gibbons, and Christos Faloutsos. 2002. ANF: A Fast and Scalable Tool for Data Mining in Massive Graphs. In *Proceedings of the 8th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '02)*.

[37] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. arXiv:2104.10350 [cs.LG]

[38] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*.

[39] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. 2019. Sequential Variational Autoencoders for Collaborative Filtering. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM '19)*.

[40] Noveen Sachdeva and Julian McAuley. 2020. How Useful Are Reviews for Recommendation? A Critical Review and Potential Improvements. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*.

[41] Noveen Sachdeva, Yi Su, and Thorsten Joachims. 2020. Off-Policy Bandits with Deficient Support. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*.

[42] T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *Proceedings of The 33rd International Conference on Machine Learning*.

[43] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. arXiv:1907.10597 [cs.CY]

[44] Ozan Sener and Silvio Savarese. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *ICLR*.

[45] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy.

[46] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

[47] M. Toneva, A. Sordoni, R. Combes, A. Trischler, Y. Bengio, and G. Gordon. 2019. An Empirical Study of Example Forgetting during Deep Neural Network Learning. In *ICLR*.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is All you Need. In *NeurIPS*.

[49] M. Wan and J. McAuley. 2018. Item Recommendation on Monotonic Behavior Chains. In *Proceedings of the 12th ACM Conference on Recommender Systems*.

[50] C. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. A. Behram, J. Huang, C. Bai, M. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H. S. Lee, B. Akyildiz, M. Balandat, J. Spisak, R. Jain, M. Rabbat, and K. Hazelwood. 2021. Sustainable AI: Environmental Implications, Challenges and Opportunities. arXiv:2111.00364 [cs.LG]

[51] Amatriain X, Jaimes A, Oliver N, and Pujol J.M. 2011. Data Mining Methods for Recommender Systems. In *Recommender Systems Handbook*. Springer.

[52] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD '18*.

[53] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *AAAI*.

[54] Ziwei Zhu, Yun He, Yin Zhang, and James Caverlee. 2020. Unbiased Implicit Recommendation and Propensity Estimation via Combinational Joint Learning. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20)*.