Randomized Fractal Expansions for Production-Scale Public Collaborative-Filtering Data Sets

Francois Belletti
Google AI
belletti@google.com

Nicolas Mayoraz
Google AI
nmayoraz@google.com

Tayo Oguntebi
Google AI
tayo@google.com

Karthik Lakshmanan
Google AI
lakshmanan@google.com

Taylor Robie
Google AI
taylorrobie@google.com

Walid Krichene
Google AI
walidk@google.com

Pankaj Kanwar
Google AI
pkanwar@google.com

Yi-Fan Chen
Google AI
yifanchen@google.com

John Anderson
Google AI
janders@google.com

***Abstract*—Recommender system research suffers from a disconnect between the size of academic data sets and the scale of industrial production systems. In order to bridge that gap, we propose to generate large-scale user/item interaction data sets by expanding pre-existing public data sets. Our key contribution is a technique that expands user/item incidence matrices to large numbers of rows (users), columns (items), and non-zero values (interactions). The proposed method adapts Kronecker Graph Theory to preserve key higher order statistical properties such as the fat-tailed distribution of user engagements, item popularity, and singular value spectra of user/item interaction matrices. Preserving such properties is key to building large realistic synthetic data sets which can be employed reliably to benchmark recommender systems and the systems employed to train them.**

We further apply our stochastic expansion algorithm to the binarized MovieLens 20M data set, which comprises 20M interactions between 27K movies and 138K users. The resulting expanded data set has 1.2B ratings, 2.2M users, and 855K items, which can be scaled up or down. Furthermore, we present collaborative filtering experiments demonstrating that the generated synthetic data entails valuable insights for machine learning at scale in recommender systems. We provide code pointers to reproduce our data and our experiments.

***Index Terms*—Machine Learning, Deep Learning, Recommender Systems, Graph Theory, Simulation**

I. Introduction

Machine Learning (ML) benchmarks compare the capabilities of models, distributed training systems and linear algebra accelerators on realistic problems at scale. For these benchmarks to be effective, results need to be reproducible by many different groups which implies that publicly shared data sets need to be available.

Unfortunately, while recommendation systems constitute a key industrial application of ML at scale, large public data sets recording user/item interactions on online platforms are not yet available. For instance, although the Netflix data set [4] and the MovieLens data set [13] are public, they are orders of magnitude smaller than proprietary data [2], [8], [36].

TABLE I: MovieLens 20M [13] vs industrial dataset in [36].

	MovieLens 20M	**Industrial**
#users	138K	Hundreds of Millions
#items	27K	2M
#topics	19	600K
#observations	20M	Hundreds of Billions

Proprietary data sets and privacy: While releasing large anonymized proprietary recommendation data sets may seem an acceptable solution from a technical standpoint, it is a non-trivial problem to preserve user privacy while still maintaining useful characteristics of the dataset. For instance, [27] shows a privacy breach of the Netflix prize dataset. More importantly, publishing anonymized industrial data sets runs counter to user expectations that their data may only be used in a restricted manner to improve the quality of their experience on the platform.

Therefore, we decide not to make user data more broadly available to preserve the privacy of users. We instead choose to produce synthetic yet realistic data sets whose scale is commensurate with that of our production problems while only consuming already publicly available data.

Realistic MovieLens 1 billion+ dataset: In this work, we focus on the MovieLens dataset which only entails movie ratings posted publicly by users of the MovieLens platform and is now a standard benchmark in recommender system research. A binarized version of this dataset is obtained when all the ratings are substituted by 1.0 as in the Neural Collaborative Filtering (NCF) experiments [14]. Although the binarized version is representative of industrial Collaborative Filtering (CF) aiming at predicting which item a given user is most likely to view [8], the data set still only entails few observed interactions and more importantly a small catalogue of users/items, compared to industrial recommendation data.

Industrial recommender systems for user generated content

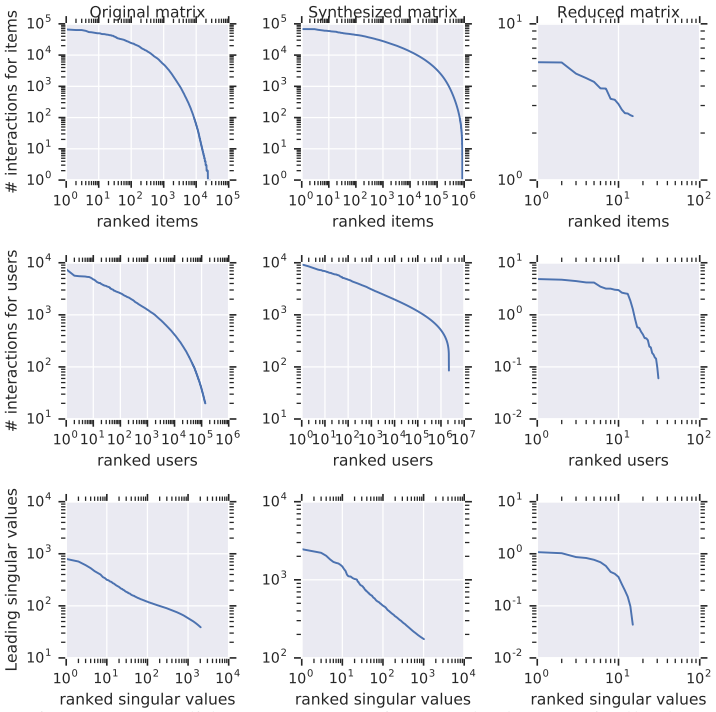

Fig. 1: We validate our expansion method numerically by checking that the distributions of row-wise sums, column-wise sums and singular values are similar between the original data set R (left column) and the synthetic data set $\widetilde{R}$ (middle column). We can observe the preservation of the linear log-log correspondence for the higher values in the distributions of interest as well as the accelerating decay for smaller values. The intermediate reduced matrix $\widehat{R}$ is also designed to be similar to R which is confirmed here as well (right column). Detailed notation is presented in sections III and IV.

platforms (e.g. Pinterest, Instagram, Youtube) typically have to nominate items from catalogues comprising several million distinct elements. The corresponding recommenders typically learn a vector valued embedding in $\mathbb{R}^d$ (with $d \sim 10^2$ or 10^3) for each of the millions of users and items of the catalog. Storing, accessing and training such vast embedding tables presents unique challenges as large tables will no longer easily fit in the memory of a single machine. Furthermore, the abundance of data asks for a higher training throughput to serve less stale models. By scaling up the public MovieLens data set, we want to move the problem into a regime where such issues are critical so that the corresponding benchmark is helpful for the industry.

In order to provide a new data set — more aligned with the needs of production scale recommender systems — we therefore aim at expanding publicly available data by creating a realistic surrogate abiding by the following constraints: *orders of magnitude more users and items are present in the synthetic data set whose first and second order statistics match those of the original data set.* Figure 1 presents the match between such properties of the original MovieLens 20M and our synthetic MovieLens 1B.

Adapting Kronecker Graph expansions to binarized user/item interactions: We employ the Kronecker Graph Theory introduced in [22] to achieve a suitable fractal expansion of recommendation data to benchmark user/item factorization approaches for CF [14], [19]. Consider a recommendation problem comprising m users and n items. Let $(R_{i,j})_{i=1...m,j=1...n}$ be the sparse matrix of binarized recorded interactions (i.e. 1 if user i has consumed item j and 0 otherwise). The key insight we develop is that a fractal expansion of R can preserve high level statistics of the original data set while scaling its size up by multiple orders of magnitudes. To that end, we present the following contributions to help recommender system research scale up in public benchmarks:

- we introduce a randomly shuffled Kronecker product and take steps to prevent test data from leaking into the data set employed to train CF models;
- we produce a synthetic yet realistic MovieLens 1.2 billion dataset and demonstrate that key properties of the original dataset are preserved by our technique;
- we illustrate through Matrix Factorization (MF) [19] and NCF [14] experiments that the synthetic data constitutes a valuable benchmark at scale.

II. Related work

In the present paper we assume a recommender system has to suggest relevant items to users. Although other approaches are very popular such as content based recommendations [29] or social recommendations [6], collaborative filtering remains a prevalent approach for recommenders [25], [28], [30].

Collaborative filtering (CF): The key insight behind CF is to learn affinities between users and items based on previously collected user/item interaction data. CF exists in different flavors. Neighborhood methods group users by inter-user similarity and will recommend items to a given user that have been consumed by neighbors [29]. Latent methods try to decompose user/item affinity as the result of the interaction of a few underlying representative factors characterizing the user and the item (e.g. Principal Component Analysis [17], Latent Dirichlet Allocation [5]). Matrix Factorization [19] is a Latent Factor Method that relies on solving the matrix completion problem to recommend items for users.

Given a sparse matrix user/item interaction matrix $R = (r_{i,j})_{i=1...m,j=1...n}$, user and item latent factors are classically learned by approximating R with a low rank matrix XY^T where $X \in \mathbb{R}^{m,d}$ entails the user factors and $Y \in \mathbb{R}^{n,d}$ contains the item factors. The data set R represents ratings as in the unmodified MovieLens dataset [13] or item consumption ($r_{i,j} = 1$ if and only if the user i has consumed item j [4]) — the latter being considered here. A key determiner of the number d of latent factors needed to accurately approximate R is the singular value spectrum of R. Therefore we try to preserve the spectral properties of the original data.

Deep Learning for recommender systems: CF has known many recent developments which motivate our objective of expanding public data sets in a realistic manner. Deep Neural Networks (DNNs) are now becoming common in both non-linear matrix factorization tasks [8], [14], [35] and sequen-

tial recommendations [12], [32], [37]. The mapping between user/item pairs and ratings is generally learned by training the neural model to predict user behavior in previously observed user/item interactions.

DNNs consume large quantities of data and are computationally expensive to train, therefore they give rise to commonly shared benchmarks aimed at speeding up training. For training, a Stochastic Gradient Descent method is employed [21] which requires forward model computation and back-propagation on many mini-batches of (user, item, score) examples. The matrix completion task still consists in predicting an interaction between user i and item j although $r_{i,j}$ has not been observed. The model typically iterates over billions of examples during training.

Freshness in recommender systems and training acceleration: Model freshness is generally critical to industrial recommendations [8] which implies that only limited time is available to re-train the model on newly available data. Unfortunately, public recommendation data sets are too small to provide training-time-to-accuracy benchmarks that can be realistically employed for industrial applications. Compared to industrial data sets, too few different examples are available in MovieLens 20M for instance and the number of different available items is orders of magnitude too small. In many industrial settings, the recommendation model learns embedding matrices with millions to billions of rows (users/items) and hundreds of millions of columns (latent factors) whose memory footprint dominates that of the rest of the model by several orders of magnitude. During training, the latency and bandwidth of the access to embedding matrices have a prominent influence on the final throughput in examples/second. Such computational difficulties associated with learning large embedding matrices are worthwhile solving in benchmarks. The multi-billion interaction size of the data set used for training is also a major factor that affects industrial modeling choices and infrastructure.

Synthetic data sets and Kronecker graphs: A recent approach to creating synthetic recommendation data sets consists in making parametric assumptions on user behavior by instantiating a user model interacting with an online platform [7], [31]. Unfortunately, such methods (even calibrated to reproduce empirical facts in actual data sets) do not provide strong guarantees that the resulting interaction data is similar to the original. A challenging problem in this domain is to build user models that can provide such guarantees in the presence of Long Range Dependent dynamics [2], [33] with irregular observations in time [3], which can be validated using online experiments. In this work, instead of simulating recommendations in a parametric user-centric way as in [7], [31], we choose a non-parametric approach operating directly in the space of user/item affinity. In order to synthesize a large realistic dataset in a principled manner, we adapt the Kronecker expansions which have previously been employed to produce large realistic graphs in [22], [23]. In particular, we show how randomized block-wise shuffling operations help address limitations of naive methods which yield interaction matrices with a discernible block-wise repetitive structure.

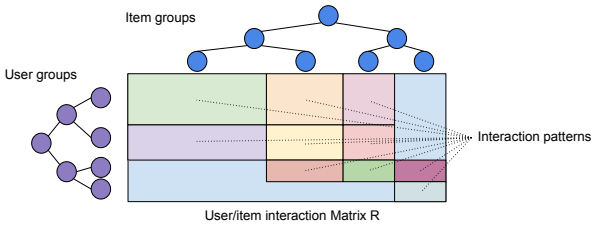

Fig. 2: Postulated hierarchical and self-similar user/item interaction patterns in recommendation data sets.

III. Fractal expansions of user/item interaction data sets

The present section delineates the insights orienting our design decisions when expanding recommendation data sets.

1) Self-similarity in user/item interactions: Interactions between users and items follow a natural hierarchy in data sets where items can be organized in topics, genres, and categories [36]. There is for instance an item-level fractal structure in MovieLens 20M with a tree-like structure of genres, sub-genres, and directors. If users were clustered according to their demographics and tastes, another hierarchy would be formed [29]. The corresponding structured user/item interaction matrix is illustrated in Figure 2. The hierarchical nature of user/item interactions makes the recommendation data set structurally self-similar (i.e. patterns that occur at more granular scales resemble those affecting coarser scales [26]).

We choose to expand the user/item interaction matrix by extrapolating this self-similar structure and simulating its growth to yet another level of granularity: original items and users are considered fictional topic and user groups in the expanded data set. The large number of fictitious topics also creates a data set closer to industrial applications with $> 100K$ different topics as in Table I. A key advantage of this fractal procedure is that it may be entirely non-parametric and designed to preserve high level properties of the original dataset. In particular, a fractal expansion re-introduces the patterns originally observed in the entire real dataset within each block of local interactions of the synthetic user/item matrix. We now show that the Kronecker operator enables such a construction.

2) Fractal expansion through Kronecker products: The Kronecker product — denoted $\otimes$ — is a matrix operator with an intrinsic self-similar structure:

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} \quad (1)$$

where $A \in \mathbb{R}^{m,n}$, $B \in \mathbb{R}^{p,q}$ and $A \otimes B \in \mathbb{R}^{mp,nq}$.

An important challenge here is the size of the original matrix we deal with: $R \in \mathbb{R}^{(138\times10^3, 27\times10^3)}$. A naive Kronecker expansion $R \otimes R$ would synthesize a rating matrix with 19 billion users which is too large. Thus, although Kronecker products seem like an ideal candidate for the mechanism at the core of the self-similar synthesis of a larger recommendation dataset, some modifications are needed to the algorithms developed in [23].

3) Reduced Kronecker expansions: We choose to synthesize a user/item rating matrix

$$\widetilde{R} = \widehat{R} \otimes R$$

where $\widehat{R}$ is a matrix derived from R but much smaller (for instance $\widehat{R} \in \mathbb{R}^{128,256}$). For reasons that will become apparent as we explore some theoretical properties of Kronecker fractal expansions, we want to construct a smaller derived matrix $\widehat{R}$ that shares similarities with R. In particular, we seek $\widehat{R}$ with a similar row-wise sum distribution (user engagement distribution), column-wise distribution (item engagement distribution) and singular value spectrum (signal to noise ratio distribution in the matrix factorization).

4) Generalized Kronecker products: Generalized fractal expansions can be defined by altering the standard Kronecker product. We consider such extensions here as candidates to engineer more interesting synthetic data sets. One drawback though is that these extensions may not preserve analytic tractability. As a generalization of $\otimes$ one can define a binary matrix operator $\otimes_F$ with $F : \mathbb{R} \times \mathbb{R}^{m,n} \times \mathbb{N} \to \mathbb{R}^{p,q}$ as follows:

$$A \otimes_F B = \begin{bmatrix} F(a_{11}, B, \omega_{11}) & \dots & F(a_{1n}, B, \omega_{1n}) \\ \vdots & \ddots & \vdots \\ F(a_{m1}, B, \omega_{m1}) & \dots & F(a_{mn}, B, \omega_{mn}) \end{bmatrix} \quad (2)$$

where $\omega_{11}, \dots, \omega_{mn}$ are random perturbations.

5) Stochastic Kronecker product and dropout for binary rating matrices: With binary ratings, a standard Kronecker product is not suitable as the ratings all take the same value of 1 and therefore multiplications by elements of $\widehat{R}$ do not produce ratings that are all still binary. We instead use the stochastic Kronecker graph approach from [23] and employ the reduced matrix's elements $\widehat{R}$ as dropout rates over the matrix R. When computing the block i, j of the expanded rating matrix, instead of using $\widehat{R}_{i,j}R$, we consider dropout$(R, rate = \widehat{R}_{i,j})$ after having re-scaled $\widehat{R}$ so that all its elements are in $[0, 1]$. For each element k, l of R, the dropout function for a rate $\widehat{R}_{i,j}$ samples independently from a Bernoulli distribution with parameter $\widehat{R}_{i,j}$. If the sampled number is 1, $R_{k,l}$, is kept unchanged, otherwise it is dropped and set to 0. Such a *block* dropout operator enjoys statistical properties that are similar to the Kronecker product [23] while being readily employable on binary data-sets. After applying sub-matrix shuffling which we introduce in the next paragraph, the stochastic Kronecker product we devise can therefore be written

$$A \otimes_{\text{rand}} B = \begin{bmatrix} \text{sh}\left(\text{dr}(1 - a_{11}, B)\right) & \dots & \text{sh}\left(\text{dr}(1 - a_{1n}, B)\right) \\ \vdots & \ddots & \vdots \\ \text{sh}\left(\text{dr}(1 - a_{m1}, B)\right) & \dots & \text{sh}\left(\text{dr}(1 - a_{mn}, B)\right) \end{bmatrix} \quad (3)$$

where "sh" denotes the random row-wise and column-wise shuffling operator and "dr" denotes the dropout operator whose first argument is the dropout rate and whose second argument is the matrix from which to zero out elements at random. We now explain why we introduce intra-block shuffling within each block of output matrix.

6) Randomized shuffling and Kronecker SVD: Another limitation of standard Kronecker products is their block-wise repetitive structure. Although the synthetic data set is still hard to factorize as the product of two low rank matrices because its singular values are still distributed similarly to the original data set, it is now easy to factorize with a Kronecker SVD [18] which takes advantage of the block-wise repetitions in Eq (1). Therefore we shuffle rows and columns of each block in Eq (1) independently at random (each sub-matrix is shuffled differently). The shuffles break the block-wise repetitive structure and prevents Kronecker SVD from producing a trivial solution to the factorization problem. As a result, the expansion technique we present appears as a reliable first candidate to train Matrix Factorization models [29] and DNN-based user/item similarity scoring models [14].

7) Preventing leaks from the test set into the training set: For matrix factorization tasks, the usual procedure to build disjoint training and test data sets for MovieLens 20M consists in selecting some ratings and removing them from the training set while adding them to the test set [14]. A naive adaptation of the test data generation procedure to our extended data set would select test items directly on the larger matrix $\widetilde{R}$. Unfortunately, as $\widetilde{R} = \widehat{R} \otimes R$, such a procedure would implicitly share data between the training and test sets through $\widehat{R}$ which incorporates information from the entire original data set R. In order to generate training and test data without leaking test data into the training set, we proceed as follows.

We consider two separate training and test sets selected from the original data set R: R_{train} and R_{test}. With $R \in \mathbb{R}^{m,n}$, we have $R_{\text{train}} \in \mathbb{R}^{m,n}$ and $R_{\text{test}} \in \mathbb{R}^{m,n}$. For the MovieLens data set, where each rating of a given item by a specific user is timestamped, a typical approach to defining training and testing sets removes the last rating of each user from the train set and adds it to the test set [14]. The smaller matrix $\widehat{R_{\text{train}}}$ is now derived from R_{train} exclusively, without incorporating any data from R_{test}. We create the extended versions of the train and test data sets separately as follows: $\widetilde{R_{\text{train}}} = \widehat{R_{\text{train}}} \otimes R_{\text{train}}$ and $\widetilde{R_{\text{test}}} = \widehat{R_{\text{train}}} \otimes R_{\text{test}}$. By construction, the procedure prevents test data from leaking into the train data.

8) Consistent randomized operations across training and testing sets: With a stochastic Kronecker $\otimes_{\text{rand}}$ featuring dropout and block-wise shuffling, additional precautions need to be taken. In order to guarantee that the randomized shuffles of rows and columns are consistent between the training and testing data, we flip the sign of the test elements in the rating matrix to keep track of their belonging to the test set. We apply all randomized operations to the resulting matrix comprising elements in $\{-1, 0, 1\}$: $\widetilde{R_{\text{temp}}} = \widehat{R_{\text{train}}} \otimes_{\text{rand}} (R_{\text{train}} - R_{\text{test}})$ where $R_{\text{train}} \in \mathbb{R}^{m,n}$ and $R_{\text{test}} \in \mathbb{R}^{m,n}$. The positive elements of R_{temp} are attributed to $\widetilde{R_{\text{train}}}$ and the negative elements are attributed to $\widetilde{R_{\text{test}}}$ after having flipped their sign $\forall i \in \{1, \dots, mp\}, j \in \{1, \dots, nq\}$,

$$\widetilde{R_{\text{train}}}_{i,j} = \widetilde{R_{\text{temp}}}_{i,j} \text{ if } \widetilde{R_{\text{temp}}}_{i,j} = 1 \text{ else } 0,$$
$$\widetilde{R_{\text{test}}}_{i,j} = -\widetilde{R_{\text{temp}}}_{i,j} \text{ if } \widetilde{R_{\text{temp}}}_{i,j} = -1 \text{ else } 0.$$

We now guarantee the consistency of randomized shuffles of the sub-blocks in $\widetilde{R_{\text{train}}}$ and $\widetilde{R_{\text{test}}}$.

IV. Statistical properties of Kronecker fractal expansions

After having introduced Kronecker products to self-similarly expand a recommendation dataset into a much larger one, we now develop theoretical insights about how the transform preserves crucial common properties with the original.

1) Salient empirical facts in MovieLens data: First, we introduce the critical properties we want to preserve. As a user/item interaction dataset on an online platform, one expects MovieLens to feature common properties of recommendation data sets such as power-law or fat-tailed distributions [36] (here a power-law or fat-tailed distribution denotes f with $f(x) \sim_{x \to +\infty} \alpha x^{-\beta}$ with $\alpha > 0$ and $\beta > 0$).

To characterize such a behavior in the MovieLens data set, we take a look at the distribution of the total ratings along the item axis and the user axis. In other words, we compute row-wise and column-wise sums for the matrix R and observe their distributions. The corresponding ranked distributions are exposed in Figure 1 and do exhibit a clear power-law behavior for rather popular items. However, we observe that tail items have a higher popularity decay rate. Similarly, the engagement decay rate increases for the group of less engaged users.

The other approximate power-law we find in Figure 1 lies in the singular value spectrum of the MovieLens dataset. We compute the top k singular values [15] of the MovieLens rating matrix R by power iteration, which can scale to its large $(138K, 27K)$ dimension. The dominant singular values of R and the corresponding singular vectors help approximate R by $R \simeq U\Sigma V$ where Σ is diagonal of dimension (k, k), U is column-orthogonal of dimension (m, k) and V is row-orthogonal of dimension (k, n) — which yields the rank k matrix closest to R in Frobenius norm.

Examining the distribution of the 2048 top singular values of R in the MovieLens dataset (which has at most $27K$ non-zero singular values) in Figure 1 highlights a clear power-law behavior in the highest magnitude part of the spectrum of R.

As explained in the introduction, our requirements for the transform applied to R are threefold: *we want to preserve the distributions of row-wise sums of R, column-wise sums of R and singular value distribution of R.* Additional requirements, beyond first and second order high level statistics, will further increase our confidence in the realism of the expanded synthetic dataset.

2) Analytic tractability through standard Kronecker products: Although we use a randomized version of the Kronecker product which does not offer the same level of analytic tractability, the choice of such a transform is deeply anchored in some of the theoretical properties of the standard Kronecker product. We now expose how — in its standard deterministic version — the fractal transform design we rely on preserves the key statistical properties of the previous section.

Definition 1: Consider $A \in \mathbb{R}^{m,n} = (a_{i,j})_{i=1\dots m, j=1\dots n}$, we denote the set $\left\{\sum_{j=1}^{n} a_{i,j}\right\}$ of row-wise sums of A by $\mathcal{R}(A)$, the set $\{\sum_{i=1}^{m} a_{i,j}\}$ of column-wise sums of A by $\mathcal{C}(A)$, and the set of non-zero singular values of A by $\mathcal{S}(A)$.

First we focus on conservation properties in terms of row-wise and column-wise sums which correspond respectively to marginalized user engagement and item popularity distributions. In the following, $\times$ denotes the Minkowski product of two sets, i.e. $A \times B = \{a \times b \mid \forall a \in A, \forall b \in B\}$.

Proposition 1: Consider $A \in \mathbb{R}^{m,n}$ and $B \in \mathbb{R}^{p,q}$. Then $\mathcal{R}(A \otimes B) = \mathcal{R}(A) \times \mathcal{R}(B)$ and $\mathcal{C}(A \otimes B) = \mathcal{C}(A) \times \mathcal{C}(B)$.

The proposition above is immediate to prove. The following theorem is a well known property of Kronecker products therefore we do not present its proof.

Theorem 1: Consider $A \in \mathbb{R}^{m,n}$ and $B \in \mathbb{R}^{p,q}$. Then $\mathcal{S}(A \otimes B) = \mathcal{S}(A) \times \mathcal{S}(B)$.

In practice, we use a randomized version of the Kronecker product whose block-wise shuffles do not have an analytically tractable effect on the high order statistics of the rating matrix. Therefore, we rely in section V-4 on a statistical examination of the properties of the extended synthetic data set we produce with our randomized fractal operator to verify that our original theoretical insights from the deterministic case are still valid. In particular, we demonstrate that original high order statistics of the new data set we produce preserve the original properties of the binary MovieLens 20M.

3) Constructing a reduced $\widehat{R}$ matrix similar to R: We use analytic insights from the deterministic standard Kronecker product to dictate our design of $\widehat{R}$. We validate ex-post that those insights apply to our generalized transform. Considering that the quasi power-law properties of R imply — as in [23] — that $\mathcal{S}(R) \times \mathcal{S}(R)$ has a similar distribution to $\mathcal{S}(R)$, we seek a small $\widehat{R}$ whose high order statistical properties are similar to those of R. As we want to generate a dataset with several billion user/item interactions, millions of distinct users and millions of distinct items, we are looking for a matrix $\widehat{R}$ with a few hundred or thousand rows and columns. The reduced matrix $\widehat{R}$ we seek is therefore orders of magnitude smaller than R. In our experiments, it is noteworthy that naive uniform user and item sampling strategies have not yielded smaller matrices $\widehat{R}$ with similar properties to R in our experiments. Different random projections [1], [10], [24] could more generally be employed.

We now describe the technique we employed to produce a reduced size matrix $\widehat{R}$ with first and second order properties close to R which in turn led to constructing an expansion matrix $\widetilde{R} = \widehat{R} \otimes R$ similar to R. We want the dimensions of $\widehat{R}$ to be (m', n') with $m' << m$ and $n' << n$. Consider again the approximate Singular Value Decomposition (SVD) [15] of R with the $k = min(m', n')$ principal singular values of R:

$$R \simeq U\Sigma V \tag{4}$$

where $U \in \mathbb{R}^{n,k}$ (resp. $V \in \mathbb{R}^{k,m}$) has orthogonal columns (resp. rows), and $\Sigma \in \mathbb{R}^{k,k}$ is diagonal with non-negative terms. To reduce the number of rows and columns of R while preserving its top k singular values a trivial solution would consist in replacing U and V by a small random orthogonal matrices with few rows and columns respectively.

MovieLens	20M	1B train set	1B test set
Interactions	20M	1.22B	12.7M
Users	138K	2.20M	2.20M
Items	27K	855K	855K

TABLE II: Synthetic expanded MovieLens: MovieLens 1B.

Unfortunately, such a method would only seemingly preserve the spectral properties of R as the principal singular vectors would be widely changed. Such properties are important: one of the key advantages of employing Kronecker products in [23] is the preservation of the network values, i.e. the distributions of singular vector components of a graph's adjacency matrix.

To obtain a matrix $\widetilde{U} \in \mathbb{R}^{n',k}$ with fewer rows than U but column-orthogonal and similar to U in the distribution of its values we use the following procedure. We re-size U down to n' rows with $n' < n$ by down-scaling through local averaging (using skimage.transform.resize in the scikit-image library [34]). Let $\bar{U} \in \mathbb{R}^{n',k}$ be the corresponding resized version of U. We then construct $\widetilde{U}$ as the column orthogonal matrix in $\mathbb{R}^{n',k}$ closest in Frobenius norm to $\bar{U}$. Therefore as in [11] we compute

$$\widetilde{U} = \bar{U} \left(\bar{U}^T \bar{U} \right)^{-1/2}. \tag{5}$$

We apply a similar procedure to V to reduce its number of columns which yields a row orthogonal matrix $\widetilde{V} \in \mathbb{R}^{k,m'}$ with $m' < m$. The orthogonality of $\widetilde{U}$ (column-wise) and $\widetilde{V}$ (row-wise) guarantees that the singular value spectrum of

$$\widehat{R} = \widetilde{U} \Sigma \widetilde{V} \tag{6}$$

consists exactly of the $k = min(m', n')$ leading singular values of R. Like R, $\widehat{R}$ is re-scaled to take values in $[-1, 1]$. The whole procedure is validated empirically in Figure 1.

V. Experiments on MovieLens 20M and 1B

The MovieLens 20M data comprises 20M ratings given by 138K users to 27K items. In the present section, we demonstrate how the fractal Kronecker expansion technique we devised and presented helps scale up this dataset to orders of magnitude more users, items and interactions.

1) Pre-processing of MovieLens 20M: The first pre-processing step we apply to MovieLens 20M is setting all non missing rating values to 1. The second pre-processing step filters out users who have fewer than 2 ratings with distinct timestamps. The filter enables the splitting of MovieLens 20M into a train set R_{train} consisting of all the ratings of each users except the last one in chronological order. For each user, the rating with the last timestamp is put in the test set R_{test}. Such preliminary steps are standard for ML tasks such as NCF [14]. After the pre-processing steps, we expand MovieLens 20M with the randomized Kronecker product presented in 3.

2) Size of expanded data set: We construct a reduced rating matrix $\widehat{R}$ of size $(16, 32)$. The size of the synthetic data set is detailed in Table II. It is noteworthy that our method also enables practitioners and researchers to experiment with larger and smaller data sets as well as changing their sparsity.

The high number of interactions and items enables the training of DNNs such as the NCF model [14] with a scale which is now more representative of industrial settings. Moreover, the increased data set size helps construct benchmarks for deep learning frameworks and accelerators closer to production settings in terms of user base size, item vocabulary size and number of observations.

3) Empirical properties of reduced $\widehat{R}$ matrix: The objective of the construction technique for $\widehat{R}$ was to produce a matrix sharing key properties of R though smaller in size. To that end, we aimed at constructing a matrix $\widehat{R}$ of dimension $(16, 32)$ with properties close to those of R in terms of column-wise sum, row-wise sum and singular value spectrum distributions. We now check that the construction procedure we devised does produce a $\widehat{R}$ with the properties we expected. As the impact of the re-sizing step is unclear analytically, we resort to numerical experiments to validate our method.

In Figure 1, one can assess that the first and second order properties of R and $\widehat{R}$ do match. In particular, the higher magnitude column-wise and row-wise sum distributions follow a "power-law" behavior similar to that of the original matrix. Similar observations can be made about the singular value spectra of $\widehat{R}$ and R. There is therefore now a reasonable likelihood that our adapted Kronecker expansion — although somewhat differing from the method originally presented in [23] — will enjoy the same benefits in terms of enabling data set expansion while preserving high order statistical properties. A major interrogation is whether block-wise shuffling will change such properties or not. We answer it with a thorough empirical analysis.

4) Empirical properties of the expanded data set $\widetilde{R}$: We now verify empirically that the expanded rating matrix $\widetilde{R} = \widehat{R} \otimes R$ does share common first and second order properties with the original rating matrix R. The new data size is two orders of magnitude larger in terms of number of rows and columns and four orders of magnitude larger in terms of number of non-zero terms. Notice here that because of the dropout, the density of $\widetilde{R}$ is about 20% that of R.

In Figure 1, one can confirm that the spectral properties of the expanded data set as well as the user engagement (row-wise sums) and item popularity (column-wise sums) are similar to those of the original data set. Such observations demonstrate that the theoretical insights from Proposition 1 and Theorem 1 are indeed informative of the high order statistics of the synthetic data set we generate. Our ex-post empirical study indicates that the resulting data set is representative — in its fat-tailed data distribution and quasi "power-law" singular value spectrum — of problems encountered in ML for CF. Furthermore, the expanded data set reproduces some unique properties of the original data, in particular the accelerating decay of values in ranked row-wise and column-wise sums as well as in the singular values spectrum. It is remarkable that block-wise shuffles did not disrupt these properties.

A. Collaborative filtering on large synthetic data

As final step to assess the usefulness of our data set, we train well known collaborative filtering baselines and show that the challenges we face in production with recommendations at scale are also found in the experiments we devise.

1) Matrix factorization on the synthetic data set: In our first ML experiment, we apply the Matrix Factorization (MF) technique [19] to the MovieLens 1B data set we generated. We train a MF model with cross-entropy softmax loss, which is common when learning a multi-label classification problem as defined by our binary data sets. Negative sampling is essential to reduce spurious recommendations while keeping the training time within a few hours. In this experiment, we use in-batch negative sampling, with batch size of 1024, i.e. each batch has 1 positive item and 1023 negative items for each user. In addition to negative sampling, we add to the loss a regularization term known as *gravity* as proposed in [20], which encourages the prediction for all user-item pairs towards zero. To prevent the most popular items to overwhelm the optimization, both components of the loss, cross-entropy softmax and gravity, are weighted by item weights that are proportional to $1/\sqrt{f_j}$, where f_j is the frequency of item j in the data. The weighted softmax used is similar to what is described in [16], where the item frequencies are pre-computed instead of estimated on the fly. It is noteworthy that the size of the data set as well as the memory footprint of embedding tables had us train with distributed Tensorflow over more than 40 machines. The number of factors used to represent each user and each item — embedding dimension — is the most important hyper-parameter for MF.

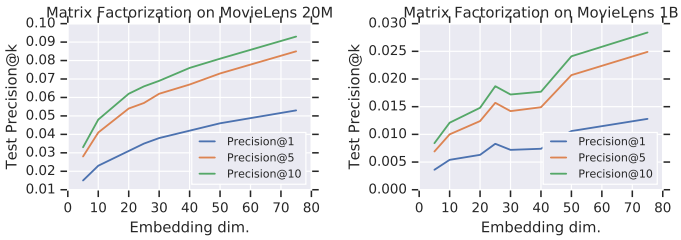

Fig. 3: MF experiments on the original data set MovieLens 20M (left) and the synthetic MovieLens 1B (right): Precision@k on the test data set. Higher embedding dimensions improve retrieval performance similarly in both data sets.

Figure 3 shows that similar improvements in retrieval performance (Precision@k on the test set) appear in both the original and the synthetic data set as we increase embedding dimensions which indicates that key properties of the original data set were indeed conferred to the larger data set by our fractal expansion technique. Also, as the synthetic data set we generate is sparser and entails another level of complexity in user/item interaction patterns, it should be more challenging to learn. Our experiments confirm that retrieval performance is much lower with the synthetic MovieLens 1B than the original MovieLens 20M. The MF experiment therefore shows that the synthetic data set we created does present most of the challenges of CF at scale while preserving key properties of the original data for ML.

2) Neural Collaborative Filtering on the synthetic data set: In our second experiment, we use the Deep Learning based Neural Collaborative Filtering (NCF) model [14] and apply it to the data set we generated. A major difficulty we have faced when training NCF on MovieLens 1B is that the size of the data set made training unreasonably slow unless hardware accelerators (GPUs) were used. In practice, we used Google Cloud VMs with 8 Nvidia V100 GPUs, SSD hard-drives, 72 CPUs and 465GB of RAM. Due to the requirements of per-epoch negative sampling and data pre-processing, relatively large amounts of compute and memory were necessary to prepare data to send to the GPU in a timely manner. As is true in the industrial setting, using multiple hardware accelerators helped us increase our batch size (to 524288) which helped us train our model to convergence in a few hours. In the NCF implementation, negative sets are fixed at each epoch. We had to proceed to a particular optimization of negative sampling to make our training speed acceptable. More precisely, we use aliased sampling [9] to have O(1) sampling consuming 2 pseudo-random-numbers per negative as opposed to O(N) sampling consuming a single pseudo-random-number. Most importantly, this approach was amenable to multi-threading. The next significant systems bottleneck arose from global operations across the entire data set. Shuffling-based operations, in particular, became very expensive. With the optimizations described and others, the previously described system demonstrated per-epoch total training times of about 17 minutes. We use ADAM to train the algorithm with a learning rate of 0.0045, $b_1 = 0.25$, $b_2 = 0.5$, $\epsilon = 10^{-8}$, 64 dimensions for the embeddings fed into the inner product layer and layers of size $256, 256, 128, 64$ for the MLP part of the network. The results are presented in Figure 4.

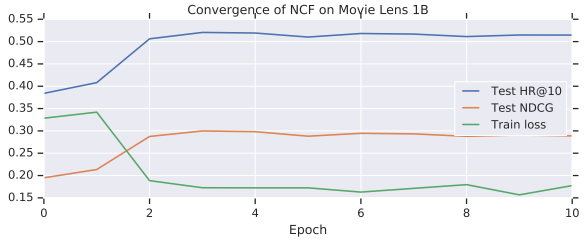

Fig. 4: Convergence of NCF on MovieLens 1B. We present the performance on the test set in the form of the Hit-rate@10 and NDCG as in [14]. The new data set is sparser and retrieval performance decreases as a result.

As in the original experiments [14], we monitor the increase of the Hit-Rate@10 and NDCG on the test set as a function of the number of epochs run on the training data. We observe that NCF does perform well on the data set we generated which indicates that the synthetic data set we present should be considered to benchmark NCF-like models and the systems employed to train them. As expected, the Hit-rate is lower (0.51 as opposed to 0.7) with our data set whose sparsity is five times higher than MovieLens 20M. For a DNN as well, the fractal

expansion procedure we present synthesizes a production-scale collaborative filtering problem with properties similar to the original MovieLens 20M. Code for data generation and NCF is located at https://github.com/mlperf/training.

VI. Conclusion

In conclusion, this paper presents an attempt at synthesizing a realistic large-scale recommendation data sets without having to make compromises in terms of user privacy. We use a small size publicly available data set, MovieLens 20M, and expand it to orders of magnitude more users, items and observed ratings. We modify the original Kronecker Graph generation method to enable a randomized expansion of the original data by orders of magnitude that yields a synthetic data set matching industrial recommendation data sets in scale: MovieLens 1B. Our numerical experiments demonstrate the data set we create has key properties to those of the original MovieLens 20M binarized rating matrix in particular for MF and NCF. We hope the wide availability of larger realistic data sets will enable new developments in ML for recommenders.

References

[1] Achlioptas, D. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences 66*, 4 (2003), 671–687.

[2] Belletti, F., Beutel, A., Jain, S., and Chi, E. Factorized recurrent neural architectures for longer range dependence. In *International Conference on Artificial Intelligence and Statistics* (2018), pp. 1522–1530.

[3] Belletti, F., Sparks, E., Bayen, A., and Gonzalez, J. Random projection design for scalable implicit smoothing of randomly observed stochastic processes. In *Artificial Intelligence and Statistics* (2017), pp. 700–708.

[4] Bennett, J., Lanning, S., et al. The netflix prize. In *Proceedings of KDD cup and workshop* (2007), vol. 2007, New York, NY, USA, p. 35.

[5] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research 3*, Jan (2003), 993–1022.

[6] Chaney, A. J., Blei, D. M., and Eliassi-Rad, T. A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems* (2015), ACM, pp. 43–50.

[7] Chaney, A. J., Stewart, B. M., and Engelhardt, B. E. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. *arXiv preprint arXiv:1710.11214* (2017).

[8] Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (2016), ACM, pp. 191–198.

[9] Fishman, G. S., and Moore III, L. R. Sampling from a discrete distribution while preserving monotonicity. *The American Statistician 38*, 3 (1984), 219–223.

[10] Fradkin, D., and Madigan, D. Experiments with random projections for machine learning. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), ACM, pp. 517–522.

[11] Golub, G. H., and Van Loan, C. F. *Matrix computations*, vol. 3. JHU Press, 2012.

[12] Hariri, N., Mobasher, B., and Burke, R. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems* (2012), ACM, pp. 131–138.

[13] Harper, F. M., and Konstan, J. A. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis) 5*, 4 (2016), 19.

[14] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web* (2017), International World Wide Web Conferences Steering Committee, pp. 173–182.

[15] Horn, R. A., Horn, R. A., and Johnson, C. R. *Matrix analysis*. Cambridge university press, 1990.

[16] Jean, S., Cho, K., Memisevic, R., and Bengio, Y. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[17] Jolliffe, I. Principal component analysis. In *International encyclopedia of statistical science*. Springer, 2011, pp. 1094–1096.

[18] Kamm, J., and Nagy, J. G. Optimal kronecker product approximation of block toeplitz matrices. *SIAM Journal on Matrix Analysis and Applications 22*, 1 (2000), 155–172.

[19] Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 8 (2009), 30–37.

[20] Krichene, W., Mayoraz, N., Rendle, S., Zhang, L., Yi, X., Hong, L., Chi, E., and Anderson, J. Efficient training on very large corpora via gramian estimation. In *Seventh International Conference on Learning Representations (ICLR), 2019* (2019).

[21] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature 521*, 7553 (2015), 436.

[22] Leskovec, J., Chakrabarti, D., Kleinberg, J., and Faloutsos, C. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *European Conference on Principles of Data Mining and Knowledge Discovery* (2005), Springer, pp. 133–145.

[23] Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., and Ghahramani, Z. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research 11*, Feb (2010), 985–1042.

[24] Li, P., Hastie, T. J., and Church, K. W. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), ACM, pp. 287–296.

[25] Linden, G., Smith, B., and York, J. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 1 (2003), 76–80.

[26] Mandelbrot, B. B. *The fractal geometry of nature*, vol. 1. WH freeman New York, 1982.

[27] Narayanan, A., and Shmatikov, V. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105* (2006).

[28] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (1994), ACM, pp. 175–186.

[29] Ricci, F., Rokach, L., and Shapira, B. Recommender systems: introduction and challenges. In *Recommender systems handbook*. Springer, 2015, pp. 1–34.

[30] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (2001), ACM, pp. 285–295.

[31] Schmit, S., and Riquelme, C. Human interaction with recommendation systems. *arXiv preprint arXiv:1703.00535* (2017).

[32] Shani, G., Heckerman, D., and Brafman, R. I. An mdp-based recommender system. *Journal of Machine Learning Research 6*, Sep (2005), 1265–1295.

[33] Tang, J., Belletti, F., Jain, S., Chen, M., Beutel, A., Xu, C., and Chi, E. H. Towards neural mixture recommender for long range dependent user sequences. *arXiv preprint arXiv:1902.08588* (2019).

[34] Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., and Yu, T. scikit-image: image processing in python. *PeerJ 2* (2014), e453.

[35] Wang, H., Wang, N., and Yeung, D.-Y. Collaborative deep learning for recommender systems. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), ACM, pp. 1235–1244.

[36] Zhao, Q., Chen, J., Chen, M., Jain, S., Beutel, A., Belletti, F., and Chi, E. H. Categorical-attributes-based item classification for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems* (2018), ACM, pp. 320–328.

[37] Zhou, B., Hui, S. C., and Chang, K. An intelligent recommender system using sequential web access patterns. In *IEEE conference on cybernetics and intelligent systems* (2004), vol. 1, IEEE Singapore, pp. 393–398.