Bayesian Low-Rank Determinantal Point Processes

Mike Gartrell
Microsoft
mike.gartrell@acm.org

Ulrich Paquet*
Microsoft
ulripa@microsoft.com

Noam Koenigstein
Microsoft
noamko@microsoft.com

ABSTRACT

Determinantal point processes (DPPs) are an emerging model for encoding probabilities over subsets, such as shopping baskets, selected from a ground set, such as an item catalog. They have recently proved to be appealing models for a number of machine learning tasks, including product recommendation. DPPs are parametrized by a positive semi-definite kernel matrix. Prior work has shown that using a low-rank factorization of this kernel provides scalability improvements that open the door to training on large-scale datasets and computing online recommendations, both of which are infeasible with standard DPP models that use a full-rank kernel. A low-rank DPP model can be trained using an optimization-based method, such as stochastic gradient ascent, to find a point estimate of the kernel parameters, which can be performed efficiently on large-scale datasets. However, this approach requires careful tuning of regularization parameters to prevent overfitting and provide good predictive performance, which can be computationally expensive. In this paper we present a Bayesian method for learning a low-rank factorization of this kernel, which provides automatic control of regularization. We show that our Bayesian low-rank DPP model can be trained efficiently using stochastic gradient Hamiltonian Monte Carlo (SGHMC). Our Bayesian model generally provides better predictive performance on several real-world product recommendation datasets than optimization-based low-rank DPP models trained using stochastic gradient ascent, and better performance than several state-of-the art recommendation methods in many cases.

1. INTRODUCTION

Online shopping revenue has grown significantly in recent years. Central to the online retail experience is the recommendation task of "basket completion", where we seek to compute predictions for the next item that should be added to a shopping basket, given a set of items already present in the basket. Determinantal point processes (DPPs) offer an attractive model for basket completion, since they jointly model *set diversity* and item *quality* or *popularity*. DPPs also offer a compact parameterization and efficient algorithms for performing inference.

A distribution over sets that models diversity is of particular interest when recommendations are complementary. For example, consider a shopping basket that contains a smartphone and a SIM card. A collaborative filtering method based on user and item similarities, such as a matrix factorization model [27], would tend to provide recommendations that are similar to the items already present in the basket but not necessarily complementary. In this example, matrix factorization might recommend other similar smartphones to complete this basket, which may not be appropriate since the basket already contains a smartphone. In contrast, a complementary recommendation for this basket might be a smartphone case, rather than another smartphone. In this setting, DPPs would be used to learn the inherent item diversity present within the observed sets (baskets) that users purchase, and hence can provide such complementary recommendations.

DPPs have been used for a variety of machine learning tasks [16, 18, 19]. DPPs can be parameterized by a $M \times M$ positive semi-definite $\mathbf{L}$ matrix, where M is the size of the item catalog. There has been some work focused on learning DPPs from observed data consisting of example subsets [1, 10, 12, 17, 24], which is a challenging learning task that is conjectured to be NP-hard [18]. Some of this recent work has involved learning a nonparametric full-rank $\mathbf{L}$ matrix [12, 24] that does not constrain $\mathbf{L}$ to take a particular parametric form, while other recent work has involved learning a low-rank factorization of this nonparametric $\mathbf{L}$ matrix [10]. A low-rank factorization of $\mathbf{L}$ enables substantial improvements in runtime performance compared to a full-rank DPP model during training and when computing predictions, on the order of 10-20x or more, with predictive performance that is equivalent to or better than a full-rank model.

The low-rank DPP model presented in [10] uses stochastic gradient ascent to maximize an objective function defined in terms of a low-rank factorization of $\mathbf{L}$. While this approach for model learning is efficient on large-scale data, it has some drawbacks. Careful tuning of regularization hyperparameters is required to prevent overfitting and provide good predictive performance. This tuning can be performed using a line search over the range of possible regularization settings. This procedure is expensive since it entails train-

*Currently at Google DeepMind.

RecSys '16, September 15 - 19, 2016, Boston , MA, USA

ISBN 978-1-4503-4035-9/16/09…$15.00

DOI: http://dx.doi.org/10.1145/2959100.2959178

ing a separate model for each regularization setting and then selecting the model that performs best. The optimization-based approach also provides a point estimate that commits to a single most probable setting for each learned parameter, which can be problematic in that it does not consider uncertainty. In contrast to this approach, we present a Bayesian low-rank DPP model that provides better predictive performance and robust regularization, without the need for expensive hyperparameter tuning.

Our work makes the following contributions:

1. We present a Bayesian low-rank DPP model, which uses an efficient stochastic gradient Hamiltonian Monte Carlo (SGHMC) algorithm for learning from observed data.

2. The Bayesian low-rank DPP model does not require expensive hyperparameter tuning and provides robust regularization, in contrast to prior work on an optimization-based learning algorithm for low-rank DPPs.

3. A detailed experimental evaluation on several real-world datasets shows that our Bayesian model provides better predictive performance than existing low-rank DPP models. Our model also provides significantly better predictive performance than several other recommendation methods in many cases.

2. MODEL

DPPs originated in statistical mechanics [23], where they were used to model distributions of fermions. Fermions are particles that obey the Pauli exclusion principle, which indicates that no two fermions can occupy the same quantum state. As a result, systems of fermions exhibit a repulsion or "anti-bunching" effect, which is described by a DPP. This repulsive behavior is a key characteristic of DPPs, which makes them a capable model for diversity. We now proceed with some details of DPPs, including how they are defined and a method for efficient learning.

2.1 Background

A point process is a distribution over configurations of points selected from a ground set $\mathcal{Y}$, which are finite subsets of $\mathcal{Y}$. In this paper we deal only with discrete DPPs, which describe a distribution over subsets of a discrete ground set of items $\mathcal{Y} = 1, 2, \ldots, M$, which we also call the item catalog. A discrete DPP on $\mathcal{Y}$ is a probability measure $\mathcal{P}$ on $2^{\mathcal{Y}}$ (the power set or set of all subsets of $\mathcal{Y}$), such that for any $A \subseteq \mathcal{Y}$, the probability $\mathcal{P}(A)$ is specified by $\mathcal{P}(A) \propto \det(\mathbf{L}_A)$. In the context of basket completion, $\mathcal{Y}$ is the item catalog (inventory of items on sale), and Y is the subset of items in a user's basket; there are $2^{|\mathcal{Y}|}$ possible baskets. The notation $\mathbf{L}_Y$ denotes the principal submatrix of the DPP kernel $\mathbf{L}$ indexed by the items in Y, which is the restriction of $\mathbf{L}$ to the rows and columns indexed by the elements of Y: $\mathbf{L}_A \equiv [\mathbf{L}_{ij}]_{i,j \in A}$. Intuitively, the diagonal entry L_{ii} of the kernel matrix $\mathbf{L}$ captures the importance or quality of item i, while the off-diagonal entry $L_{ij} = L_{ji}$ measures the similarity between items i and j.

The normalization constant for $\mathcal{P}$ follows from the observation that $\sum_{A' \subseteq \mathcal{Y}} \det(\mathbf{L}_{A'}) = \det(\mathbf{L} + \mathbf{I})$. The value $\det(\mathbf{L}_A)$ associates a "volume" to basket A from a geometric viewpoint, and its probability is normalized by the volumes of all possible baskets $A' \subseteq \mathcal{Y}$. Therefore, we have

$$\mathcal{P}(A) = \frac{\det(\mathbf{L}_A)}{\det(\mathbf{L} + \mathbf{I})} \ . \tag{1}$$

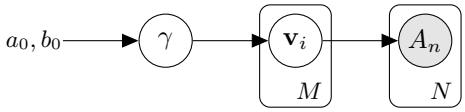

Figure 1: A graphical model for the low-rank DPP model.

We use a low-rank factorization of the $M \times M$ $\mathbf{L}$ matrix,

$$\mathbf{L} = \mathbf{V}\mathbf{V}^T \ , \tag{2}$$

for the $M \times K$ matrix $\mathbf{V}$, where M is the number of items in the item catalog and K is the number of latent trait dimensions. This low-rank factorization of $\mathbf{L}$ leads to significant efficiency improvements compared to a model that uses a full-rank $\mathbf{L}$ matrix when it comes to model learning and computing predictions [10]. This also places an implicit constraint on the space of subsets of $\mathcal{Y}$, since the model is restricted to place zero probability mass on subsets with more than K items (all eigenvalues of $\mathbf{L}$ beyond K are zero). We see this from the observation that a sample from a DPP will not be larger than the rank of $\mathbf{L}$ [11].

2.2 Model Specification

Our learning task is to fit a DPP kernel $\mathbf{L}$ based on a collection of N observed subsets $\mathcal{A} = \{A_1, \ldots, A_N\}$, where each subset A_n is composed of items from the item catalog $\mathcal{Y}$. These observed subsets in $\mathcal{A}$ constitute our training data, and our task is to infer $\mathbf{L}$ from $\mathcal{A}$. The log-likelihood for seeing $\mathcal{A}$ is

$$f(\mathbf{V}) = \log \mathcal{P}(\mathcal{A}|\mathbf{V}) = \sum_{n=1}^{N} \log \mathcal{P}(A_n|\mathbf{V}) \tag{3}$$

$$= \sum_{n=1}^{N} \log\det(\mathbf{L}_{[n]}) - N \log\det(\mathbf{L} + \mathbf{I}) \tag{4}$$

where $[n]$ indexes the observations or objects in $\mathcal{A}$. Recall from (2) that $\mathbf{L} = \mathbf{V}\mathbf{V}^T$.

Figure 1 shows the graphical model for our Bayesian low-rank DPP model. We place a multivariate Gaussian prior on each item in our model. Our prior distribution on $\mathbf{V}$ is given by

$$p(\mathbf{V}|\gamma) = \prod_{i=1}^{M} \mathcal{N}(\mathbf{v}_i; \mathbf{0}, \gamma^{-1}\mathbf{I}) \tag{5}$$

where $\mathbf{v}_i$ is the row vector from $\mathbf{V}$ for item i, and all items i share the same precision γ. We furthermore place a conjugate gamma prior on γ: $p(\gamma|a_0, b_0) = \text{Gamma}(\gamma; a_0, b_0)$. The joint distribution over $\mathcal{A}$, $\mathbf{V}$ and γ, as depicted in Figure 1, is

$$p(\mathcal{A}, \mathbf{V}, \gamma|a_0, b_0) = \mathcal{P}(\mathcal{A}|\mathbf{V})\, p(\mathbf{V}|\gamma)\, p(\gamma|a_0, b_0) \ . \tag{6}$$

To draw samples from the posterior $p(\mathbf{V}, \gamma|\mathcal{A}, a_0, b_0)$ we alternately Gibbs-sample from $p(\gamma|\mathbf{V}, a_0, b_0)$ and $p(\mathbf{V}|\mathcal{A}, \gamma)$. The first conditional distribution is

$$\begin{aligned} \gamma|\mathbf{V}, a_0, b_0 &\sim \text{Gamma}(\gamma; a, b) \\ a &= a_0 + \frac{MK}{2} \\ b &= b_0 + \frac{1}{2}\sum_i \|\mathbf{v}_i\|^2 \ , \end{aligned} \tag{7}$$

and is sampled in Line 4 in Algorithm 1.

The second conditional distribution $p(\mathbf{V}|\mathcal{A},\gamma)$ does not have the same simple form, and is:

$$p(\mathbf{V}|\mathcal{A},\gamma) \propto \exp\big(f(\mathbf{V})\big)\prod_{i=1}^{M}\mathcal{N}(\mathbf{v}_i|\mathbf{0},\gamma^{-1}\mathbf{I}) \ , \quad (8)$$

$$\log p(\mathbf{V}|\mathcal{A},\gamma) = f(\mathbf{V}) + \frac{\gamma}{2}\sum_{i=1}^{M}\mathbf{v}_i^T\mathbf{v}_i + \mathsf{const} \ , \quad (9)$$

where const indicates an additive constant independent of $\mathbf{V}$. In the following section, we consider sampling from $p(\mathbf{V}|\mathcal{A},\gamma)$.

2.3 Learning Algorithm

We estimate the conditional distribution $p(\mathbf{V}|\mathcal{A},\gamma)$ using stochastic gradient Hamiltonian Monte Carlo (SGHMC) [7]. Hamiltonian Monte Carlo (HMC) [9, 26] is a Markov chain Monte Carlo (MCMC) method that uses the gradient of the log-density of the target distribution to efficiently explore the state space of the target. HMC defines a Hamiltonian function, an idea borrowed from physics, in terms of the target distribution that we wish to sample from. The Hamiltonian function has a potential energy term, corresponding to the target distribution, and a kinetic energy term, defined in terms of auxiliary momentum variables. By updating the momentum variables using the gradient of the log-density of the target distribution, we simulate a Hamiltonian dynamical system that enables proposals of distant states, thus allowing HMC to move rapidly through the state space of the target. We cannot simulate directly from the continuous Hamiltonian dynamics, so HMC uses a discretization of this continuous system composed of a number of "leapfrog steps".

HMC requires computation of the gradient of the log-density of the target distribution over all training instances with each iteration of the algorithm, which is expensive or infeasible for large datasets or a complex target. SGHMC addresses this issue by using stochastic gradients that are computed on minibatches, where each minibatch is composed of training instances sampled uniformly at random from the full training set. SGHMC adds a friction term to the momentum update, which counteracts the effects of noise from the stochastic gradients. The estimates computed by SGHMC samples are not unbiased any more (notice that there is no accept-reject step in Algorithm 1, and the distribution that is sampled from is different from the true posterior), but due to the effectively faster mixing, we are able to efficiently train our Bayesian low-rank DPP model on large-scale datasets.

Since we learn $p(\mathbf{V}|\mathcal{A},\gamma)$ by SGHMC, we need to efficiently compute the gradient of the log-density for this distribution. We begin by computing the gradient of log-likelihood, $\partial f/\partial\mathbf{V}$, which will be a $M \times K$ matrix. For $i \in 1,\ldots,M$ and $k \in 1,\ldots,K$, we need a matrix of scalar derivatives, $\left\{\frac{\partial f}{\partial \mathbf{V}}\right\}_{ik} = \frac{\partial f}{\partial v_{ik}}$. Taking the derivative of each term of the log-likelihood, we have

$$\begin{aligned}\frac{\partial f}{\partial v_{ik}} &= \sum_{n:i\in[n]}\frac{\partial}{\partial v_{ik}}\Big(\log\det(\mathbf{L}_{[n]})\Big) - N\frac{\partial}{\partial v_{ik}}\Big(\log\det(\mathbf{L}+\mathbf{I})\Big) \\ &= \sum_{n:i\in[n]}\mathsf{tr}\left(\mathbf{L}_{[n]}^{-1}\frac{\partial\mathbf{L}_{[n]}}{\partial v_{ik}}\right) - N\,\mathsf{tr}\left((\mathbf{L}+\mathbf{I})^{-1}\frac{\partial(\mathbf{L}+\mathbf{I})}{\partial v_{ik}}\right) \ . \end{aligned} \quad (10)$$

Algorithm 1 Sampling algorithm for learning $p(\mathbf{V}|\mathcal{A},\gamma)$

```
initialize V randomly, W = 0
samples := {}
repeat
    sample γ|V, a_0, b_0 according to (7)
    // approximately sample V|A, γ:
    for leapfrogSteps j = 1, ..., L do
        W := η∇Ũ(V) − αW + N(0, 2(α − β̂)η)
        V := V + W
    end for
    samples := {samples, V}
until sufficient samples have been taken
```

Examining the first term of the derivative, we see that

$$\mathsf{tr}\left(\mathbf{L}_{[n]}^{-1}\frac{\partial\mathbf{L}_{[n]}}{\partial v_{ik}}\right) = \mathbf{a}_i\cdot\mathbf{v}_k + \sum_{j=1}^{M}a_{ji}v_{jk} \ , \quad (11)$$

where $\mathbf{a}_i$ denotes row i of the matrix $\mathbf{A} = \mathbf{L}_{[n]}^{-1}$ and $\mathbf{v}_k$ denotes column k of $\mathbf{V}_{[n]}$. Note that $\mathbf{L}_{[n]} = \mathbf{V}_{[n]}\mathbf{V}_{[n]}^T$. Computing $\mathbf{A}$ is a relatively inexpensive operation, since the number of items in each training instance A_n is generally small for many recommendation applications.

For the second term of the derivative, we see that

$$\mathsf{tr}\left((\mathbf{L}+\mathbf{I})^{-1}\frac{\partial(\mathbf{L}+\mathbf{I})}{\partial v_{ik}}\right) = \mathbf{b}_i\cdot\mathbf{v}_k + \sum_{j=1}^{M}b_{ji}v_{jk} \quad (12)$$

where $\mathbf{b}_i$ denotes row i of the matrix $\mathbf{B} = \mathbf{I}_m - \mathbf{V}(\mathbf{I}_k + \mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T$. Computing $\mathbf{B}$ is a relatively inexpensive operation, since we are inverting a $K \times K$ matrix with cost $O(K^3)$, and K (the number of latent trait dimensions) is usually set to a small value.

We now proceed with computing the gradient of the log-density for $p(\mathbf{V}|\mathcal{A},\gamma)$ shown in Equation 9. Looking at one component of this gradient, we have:

$$\frac{\partial\log p(\mathbf{V}|\mathcal{A},\gamma)}{\partial v_{ik}} = \gamma v_{ik} + \frac{\partial f}{\partial v_{ik}} \quad (13)$$

Our SGHMC algorithm is shown in Algorithm 1. In this algorithm, $\nabla\tilde{U}(\mathbf{V})$ is a noisy estimate of the log-density gradient of $p(\mathbf{V}|\mathcal{A},\gamma)$ computed for a minibatch, $\eta > 0$ is the learning rate, $\alpha \in [0,1]$ is the momentum coefficient, and $\mathbf{W}$ is the auxiliary momentum variable. $\hat{\boldsymbol{\beta}}$ is an estimate of the noise from the gradient, which we ignore by setting $\hat{\boldsymbol{\beta}} = 0$ and relying on small η, as explained in [7]. We find that setting $\alpha = 0.01$ and $\eta = 1.0\times10^{-5}$ or $\eta = 1.0\times10^{-6}$, with a minibatch size of 1000 instances, works well for the datasets we tested.

2.4 Predictions

We compute singleton next-item predictions, given a set of observed items. An example of this class of problem is "basket completion", where we seek to compute predictions for the next item that should be added to shopping basket, given a set of items already present in the basket.

We use a k-DPP to compute next-item predictions. A k-DPP is a distribution over all subsets $A \in \mathcal{Y}$ with cardinality k, where $\mathcal{Y}$ is the ground set, or the set of all items in the item catalog. Next item predictions are done via a conditional density. We compute the probability of the observed basket A, consisting of k items. For each possible item to

be recommended, given the basket, the basket is enlarged with the new item to $k+1$ items. For the new item, we determine the probability of the new set of $k+1$ items, given that k items are already in the basket, using a Monte Carlo estimate from the samples. Ignoring burn-in samples, and letting s index the S remaining $\mathbf{V}^{(s)}$ samples in samples,

$$p(A_{+1}|A,\mathcal{A},a_0,b_0) = \int \mathcal{P}(A_{+1}|A,\mathbf{V})\, p(\mathbf{V}|\mathcal{A},a_0,b_0)\, d\mathbf{V} \approx \frac{1}{S}\sum_{s=1}^{S} \mathcal{P}(A_{+1}|A,\mathbf{V}^{(s)}) \quad (14)$$

where A_{+1} indicates set A enlarged to contain a new item b from the catalog $\mathcal{Y}$. The samples in $\mathbf{V}$ implicitly marginalize out γ from the posterior density. We run the sampler to generate 2,000 samples, and discard the first 1,800 samples as burn-in. From [10], we see that the conditional probability for an item b in the singleton set B, given the appearance of items in A, is

$$\mathcal{P}(A_{+1} = A \cup B|A) = \frac{L^A_{bb}}{e_1(\lambda^A_1,\lambda^A_2,\ldots,\lambda^A_N)} \quad (15)$$

where L^A_{bb} denotes diagonal element bb from the k-DPP kernel matrix conditioned on A, $\mathbf{L}^A$; $\lambda^A_1,\lambda^A_2,\ldots,\lambda^A_N$ are the eigenvalues of $\mathbf{L}^A$; and $e_1(\lambda^A_1,\lambda^A_2,\ldots,\lambda^A_N)$ is the first elementary symmetric polynomial on these eigenvalues. See [10] for full details on how to efficiently compute conditional densities for a k-DPP given an observed basket.

3. EVALUATION

In this section we evaluate the performance of Bayesian low-rank DPP model on on the task of basket completion for several real-world datasets. We compare to several competing recommendation methods, including an optimization-based low-rank DPP [10] and two matrix factorization models [13, 27], and find that our approach provides better predictive performance in many cases.

We formulate the basket-completion task as follows. Let A_{test} be a subset of $n \geq 2$ co-purchased items (i.e, a basket) from the test-set. In order to evaluate the basket completion task, we pick an item $i \in A_{\text{test}}$ at random and remove it from A_{test}. We denote the remaining set as $A_{\text{test}-1}$. Formally, $A_{\text{test}-1} = A_{\text{test}} \setminus \{i\}$. Given a ground set of possible items $\mathcal{Y} = 1, 2, ..., M$, we define the candidates set $\mathcal{C}$ as the set of all items except those already in $A_{\text{test}-1}$; i.e., $\mathcal{C} = \mathcal{Y} \setminus A_{\text{test}-1}$. Our goal is to identify the missing item i from all other items in $\mathcal{C}$.

3.1 Datasets

Our experiments are based on several datasets:

1. **Amazon Baby Registries -** Amazon[1] is one of the world's leading online retail stores. The Amazon Baby Registries dataset [12] is a public dataset consisting of 111,006 registries or "baskets" of baby products. The choice of this dataset was motivated by the fact that it has been used by several prominent DPP studies [10, 12, 24]. The registries are collected from 15 different categories (such as "feeding", "diapers", "toys", etc.), and the items in each category are disjoint. We maintain consistency with prior work by evaluating each of its categories separately using a random split of 70% of the data for training and 30% for testing. The low-rank DPP models trained on this dataset were built with $K = 30$ trait dimensions.

In addition to the above evaluation, we also constructed a dataset composing of the concatenation of the three most popular categories: apparel, diaper, and feeding. This three-category dataset allows us to simulate data that could be observed for department stores that offer a wide range of items in different product categories. Its construction is deliberate, and concatenates three disjoint subgraphs of basket-item purchase patterns. This dataset serves to highlight differences between DPPs and models based on matrix factorization (MF). Collaborative filtering-based MF models – which model each basket and item with a latent vector – will perform poorly for this dataset, as the latent trait vectors of baskets and items in one subgraph could be arbitrarily rotated, without affecting the likelihood or predictive error in any of the other subgraphs. MF models are invariant to global rotations of the embedded trait vectors. However, for the concatenated dataset, these models are also invariant to arbitrary rotations of vectors in each disjoint subgraph, as there are no shared observations between the three categories. A global ranking based on inner products could then be arbitrarily affected by the basket and item embeddings arising from each subgraph.

2. **MS Store -** This dataset is based on data from Microsoft's web-based store [2]. The dataset is composed of 243,147 baskets consisting of commonly purchased items from a catalog of 2097 different hardware and software products. We randomly sampled of 80% of the data for training and kept the remaining 20% for testing. Recall from Section 2.1 that a low-rank DPP places zero probability mass on subsets with more than K items, where K is the number of trait dimensions in $\mathbf{V}$ or the rank of $\mathbf{L}$. With this constraint in mind, we use $K = 15$ trait dimensions for the low-rank DPP models trained on this data, since the largest observed basket in this dataset is composed of 15 items.

3. **Belgian Retail Supermarket -** This public dataset includes 88,163 baskets consisting of 16,470 unique supermarket items. It was collected in a Belgian retail supermarket over three non-consecutive time periods, as described in [3, 4]. Again, we randomly sampled 80% of the data for training and kept the remaining 20% for testing. We use $K = 76$ trait dimensions for the low-rank DPP model trained on this data, since the largest observed basket in this dataset is composed of 76 items.

Since we are interested in the basket completion task, which requires baskets containing at least two items, we remove all baskets containing only one item from each dataset before splitting the data into training and test sets.

3.2 Competing Methods

We evaluate against several baselines:

1. **Full-rank DPP -** This DPP model is parameterized by a full-rank $\mathbf{L}$ matrix, and uses a fixed-point optimization algorithm called Picard iteration [24] for learning $\mathbf{L}$. As described in [10], a full-rank DPP does not scale well to datasets containing large item catalogs during training or when computing predictions.

[1] www.amazon.com

[2] microsoftstore.com

2. **Low-rank DPP trained using stochastic gradient ascent (SGA) -** This DPP model is parameterized by a low-rank $\mathbf{L}$ matrix that is factorized using a $\mathbf{V}$ matrix composed of latent item trait vectors, and has a likelihood function identical to our Bayesian low-rank DPP. In contrast to our Bayesian approach, this optimization-based model is trained using stochastic gradient ascent, and uses regularization based on item popularity. We selected the regularization hyperparameter for this model using a line search performed with the training set.

3. **Poisson Factorization (PF) -** Poisson factorization (PF) is a prominent variant of matrix factorization designed specifically for implicit ratings [13]. The likelihood of the PF model is based on the Poisson distribution, which is useful with implicit datasets (e.g. datasets based on click or purchase events). The evaluations in this paper are based on the publicly available implementation[3] from [5]. In PF, Gamma priors are placed on the trait vectors. Following [6, 13], we set the Gamma shape and rate hyperparameters to 0.3.

4. **Reco Matrix Factorization (RecoMF) -** RecoMF is a matrix factorization model powering the Xbox Live recommendation system [27]. The likelihood term of RecoMF uses a sigmoid function to model the odds of a user liking or disliking an item in the dataset. Unlike PF, RecoMF requires the generation of synthetic negative training instances, and uses a scheme for sampling negatives based on popularity. RecoMF places Gaussian priors on the trait vectors, and gamma hyperpriors on each. We use the hyperparameter settings described in [27], which have been found to provide good performance for implicit recommendation data.

5. **Associative Classifier (AC) -** Since association rules are often used for market basket analysis [2, 15], we consider an associative classifier as a competing method. We use the publicly available implementation [8] of the Classification Based on Associations (CBA) algorithm [22]. We set minimum support and minimum confidence thresholds of 1.0% and 20.0%, respectively.

We use a flexible prior in our Bayesian low-rank DPP model by setting $a_0 = \sqrt{K}$ and $b_0 = 1$, and find that the model is not sensitive to these settings.

The matrix-factorization models are parameterized in terms of users and items. Since we have no explicit users in our data, we construct "virtual" users from the contents of each basket for the purposes of our evaluation, where a new user u_m is constructed for each basket b_m. Therefore, the set of items that u_m has purchased is simply the contents of b_m. Additionally, we use $K = 40$ trait dimensions for the matrix-factorization models.

3.3 Metrics

In the following evaluation we consider three measures:

1. **Mean Percentile Rank (MPR) -** Computing the Percentile Rank of an item requires the ability to rank the item j against all other items in $\mathcal{C}$. Therefore, the MPR evaluation results don't include the AC model, which ranks only those items for which an association rule was found. For other models we ranked the items according to their probabilities of completing the missing set Y_{n-1}. Namely, given an item i from the candidates set $\mathcal{C}$, we denote by p_i the probability $P(Y_n \cup \{i\}|Y_{n-1})$. The Percentile Rank (PR) of the missing item j is defined by

$$\mathrm{PR}_j = \frac{\sum_{j' \in \mathcal{C}} \mathbb{I}(p_j \geq p_{j'})}{|\mathcal{C}|} \times 100\%$$

where $\mathbb{I}(\cdot)$ is an indicator function and $|\mathcal{C}|$ is the number of items in the candidates set. The Mean Percentile Rank (MPR) is the average PR of all the instances in the test-set:

$$\mathrm{MPR} = \frac{\sum_{t \in \mathcal{T}} \mathrm{PR}_t}{|\mathcal{T}|}$$

where $\mathcal{T}$ is the set of test instances. MPR is a recall-oriented metric commonly used in studies that involve implicit recommendation data [14, 21]. MPR = 100 always places the held-out item for the test instance at the head of the ranked list of predictions, while MPR = 50 is equivalent to random selection.

2. **Precision@k -** We define precision@k as

$$\text{precision@}k = \frac{\sum_{t \in \mathcal{T}} \mathbb{I}[\mathrm{rank}_t \leq k]}{|\mathcal{T}|}$$

where rank_t is the predicted rank of the held-out item for test instance t. In other words, precision@k is the fraction of instances in the test set for which the predicted rank of the held-out item falls within the top k predictions.

3. **Popularity-weighted precision@k -** Datasets used to evaluate recommendation systems typically contain a popularity bias [28], where users are more likely to provide feedback on popular items. Due to this popularity bias, conventional metrics such as MPR and precision@k are typically biased toward popular items. Using ideas from [28], we define popularity-weighted precision@k:

$$\text{popularity-weighted precision@}k = \frac{\sum_{t \in \mathcal{T}} w_t \mathbb{I}[\mathrm{rank}_t \leq k]}{\sum_{t \in \mathcal{T}} w_t}$$

where w_t is the weight assigned to the held-out item for test instance t, defined as $w_t \propto \frac{1}{C(t)^\beta}$,where $C(t)$ is the number of occurrences of the held-out item for test instance t in the training data, and $\beta \in [0, 1]$. The weights are normalized, so that $\sum_{j \in \mathcal{Y}} w_j = 1$. This popularity-weighted precision@k measure assumes that item popularity follows a power-law. By assigning more weight to less popular items, for $\beta > 0$, this measure serves to bias precision@k towards less popular items. For $\beta = 0$, we obtain the conventional precision@k measure. We set $\beta = 0.5$ in our evaluation.

Figures 2, 3, and 4 show the performance of each method and dataset for our evaluation measures. Our Bayesian low-rank DPP model is denoted as "SGHMC low-rank DPP" in these figures, in reference to its learning algorithm. Note that we could not feasibly train the full-rank DPP or AC models on the Belgian dataset, since these models do not scale to datasets with large item catalogs. The Bayesian SGHMC low-rank DPP generally outperforms the optimization-based SGA low-rank DPP by a moderate amount on most metrics and datasets, which illustrates the advantage

[3]Note that [5] is actually an implementation of PF with a social component, which was disabled in the course of our evaluations since the data does not include a social graph.

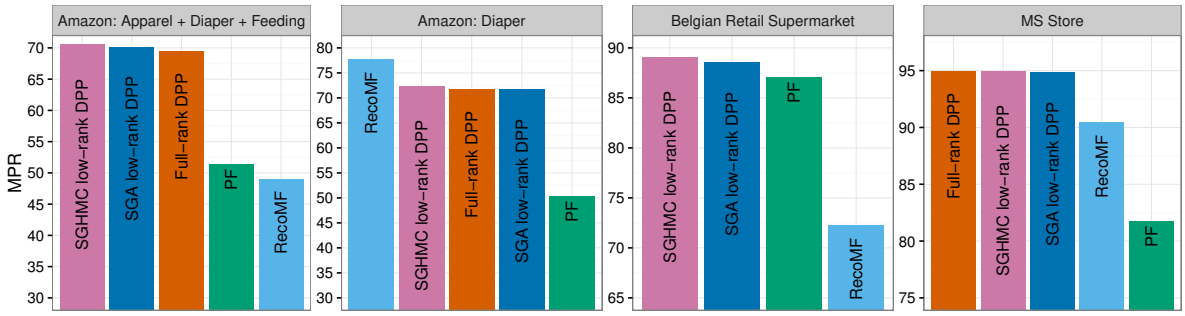

Figure 2: Mean Percentile Rank (MPR)

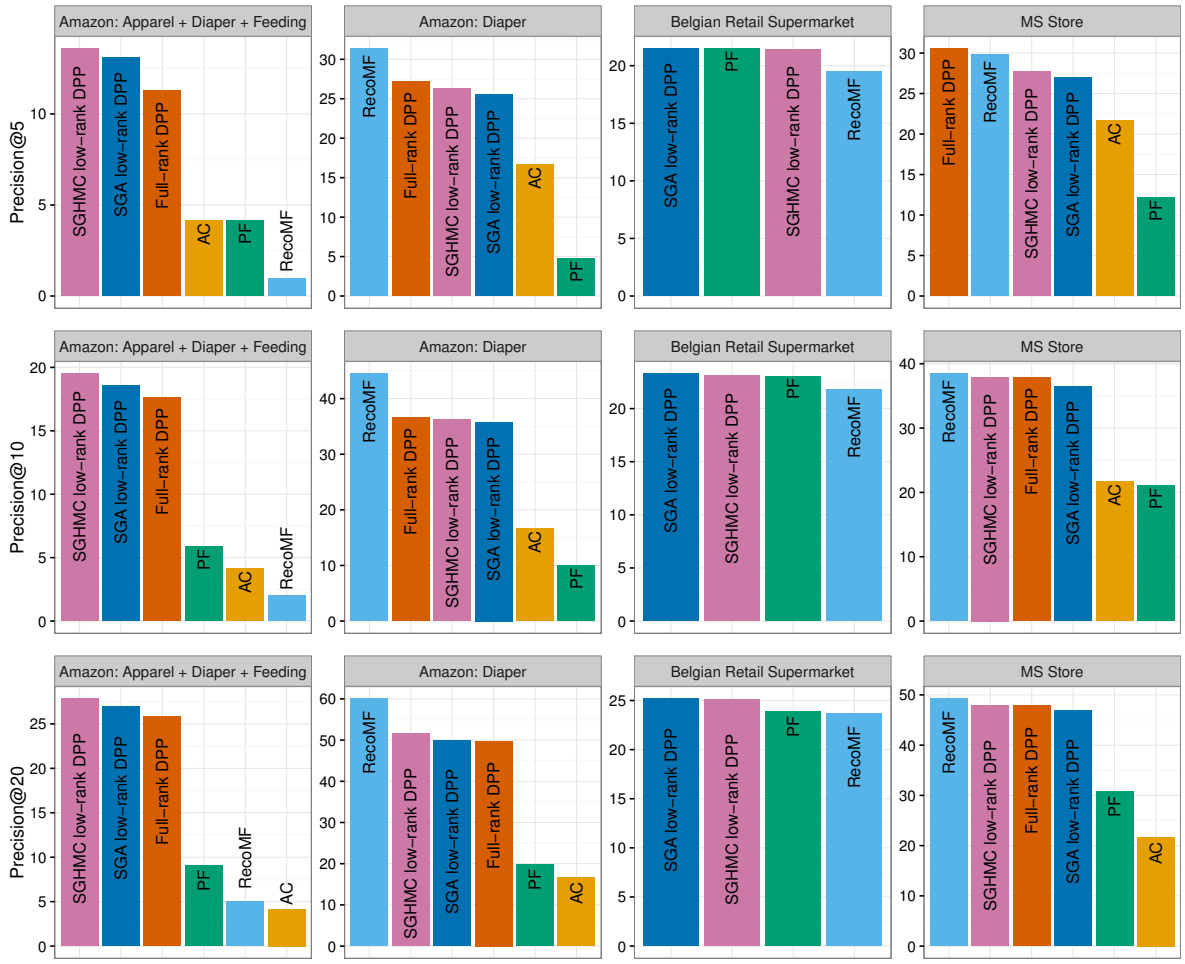

Figure 3: Precision@k

of our Bayesian approach. We attribute this improvement to the more robust regularization provided by a Bayesian model. By averaging over all settings of the parameters that are compatible with both the observed data and the prior, our Bayesian low-rank DPP model deals with uncertainty more effectively than the SGA low-rank DPP model, which commits to a single most probable setting for each parameter. An additional advantage of our Bayesian approach is that it provides a predictive distribution instead of just a single point estimate, which enables the confidence in the prediction to be estimated according to its variance. We can therefore make use of this predictive confidence when making recommendations.

We see that the RecoMF model outperforms all other models on all metrics for the Amazon Diaper dataset. For all other datasets, the Bayesian low-rank DPP model outperforms non-DPP models on MPR by a sizeable margin, and provides consistently provide high MPR across all datasets. For the precision@k metrics, the Bayesian low-rank DPP often leads or provides good performance that is close to the leader.

Limitations. The popularity-weighted precision@k results in Figure 4 highlight a limitation of the DPP models. For this metric RecoMF generally provides the best performance, with the DPP models in second place. This behavior may result from the scheme for sampling negatives by popu-

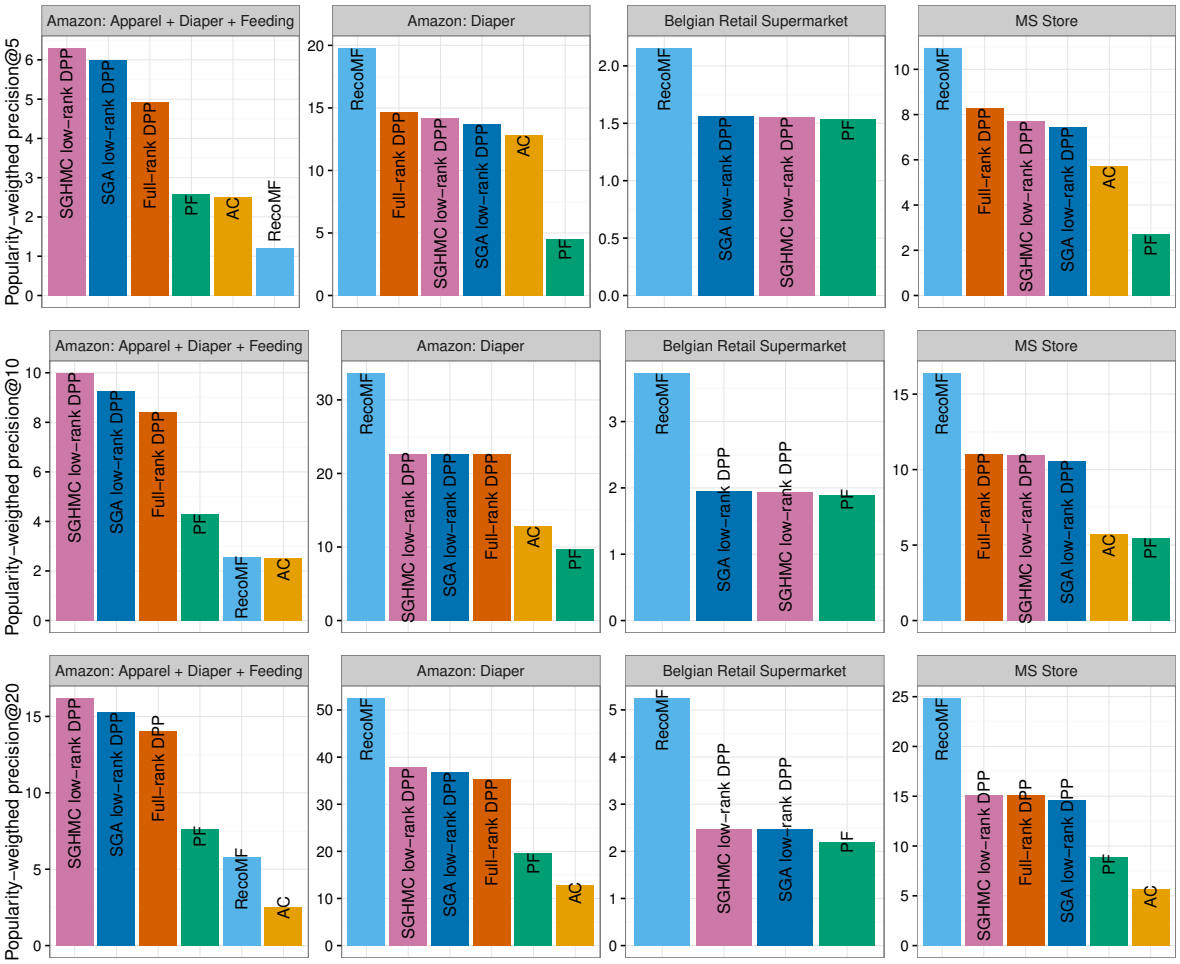

Figure 4: Popularity-weighted precision@k. These results show a limitation of the DPP models. Since this metric biases precision@k towards less popular items, we see that the RecoMF model generally provides better predictive performance for less popular items.

larity in RecoMF, which tends to improve recommendations for less popular items [27]. We conjecture that a different choice of prior for our Bayesian low-rank DPP model may improve our performance on this metric. It is also important to note the limitations of this metric, including the assumption that item popularity follows a power-law, and the power-law exponent β setting of 0.5 used when computing the metric for each dataset. Due to these limitations, the popularity-weighted precision@k results we present here may not fully reflect the empirical popularity bias actually present in the data.

4. RELATED WORK

Several algorithms for learning the DPP kernel matrix from observed data have been proposed. Ref. [12] presented one of the first methods for learning a non-parametric form of the full-rank kernel matrix, which involves an expectation-maximization (EM) algorithm. This work also considers using projected gradient ascent on the DPP log-likelihood function, but finds that this is not a viable approach since it usually results in degenerate estimates due to the projection step. In [24], a fixed-point optimization algorithm for full-rank DPP learning is described, called Picard iteration. Picard iteration has the advantage of being simple to implement and performing much faster than EM during training. Ref. [10] shows that a low-rank DPP model can be trained far more quickly than Picard iteration and therefore EM, while enabling much faster computation of predictions than is possible with any full-rank DPP model.

Ref. [1] presented Bayesian methods for learning a DPP kernel, with particular parametric forms for the similarity and quality components of the kernel. Markov chain Monte Carlo (MCMC) methods are used for sampling from the posterior distribution over kernel parameters. Furthermore, [1] uses a full-rank DPP kernel and thus shares the scalability issues common to any full-rank DPP model. In contrast to this work, and similar to [10, 12, 24], our approach uses a non-parametric form of the kernel and therefore does not assume any particular parametrization.

A method for partially learning the DPP kernel is studied in [17]. The similarity component of the DPP kernel is fixed, and a parametric from of the function for the quality component of the kernel is learned. This is a convex optimization problem, unlike the task of learning the full kernel, which is a more challenging non-convex optimization problem.

A number of approaches to the basket completion problem that we focus on in this paper have been proposed. Ref. [25] describes a user-neighborhood-based collaborative filtering

method, which uses rating data in the form of binary purchases to compute the similarity between users, and then generates a purchase prediction for a user and item by computing a weighted average of the binary ratings for that item. A technique that uses logistic regression to predict if a user will purchase an item based on binary purchase scores obtained from market basket data is described in [20]. Additionally, other collaborative filtering approaches could be applied to the basket completion problem, such as the one-class matrix factorization model in [27] and Poisson factorization [13], as we illustrate in this paper.

5. CONCLUSIONS

We have presented a Bayesian method for learning a low-rank factorization of the DPP kernel from observed data. Previous low-rank DPP approaches have focused on learning a point estimate of kernel using an optimization method, which requires expensive tuning of regularization hyperparameters to prevent overfitting and provide good predictive performance. We have shown that our Bayesian low-rank DPP model generally provides better predictive performance than an optimization-based low-rank DPP model without the need for hyperparameter tuning. Our experimental evaluation using several real-world datasets in the domain of recommendations for shopping baskets also shows that in many cases our model provides better predictive performance than competing methods, including two state-of-the-art models based on matrix factorization.

6. ACKNOWLEDGEMENTS

We thank Gal Lavee and Shay Ben Elazar for many helpful discussions. We thank Nir Nice for supporting this work.

7. REFERENCES

[1] R. H. Affandi, E. Fox, R. Adams, and B. Taskar. Learning the parameters of determinantal point process kernels. In *ICML*, pages 1224–1232, 2014.

[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of SIGMOD 1993*, pages 207–216, 1993.

[3] T. Brijs. Retail market basket data set. In *Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, 2003.

[4] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In *KDD*, pages 254–260, 1999.

[5] A. J. Chaney. Social Poisson factorization (SPF). https://github.com/ajbc/spf, 2015.

[6] A. J. Chaney, D. M. Blei, and T. Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In *RecSys*, pages 43–50, 2015.

[7] T. Chen, E. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In *ICML*, pages 1683–1691, 2014.

[8] F. Coenen. TLUCS KDD implementation of CBA (classification based on associations). http://www.csc.liv.ac.uk/~frans/KDD/Software/CMAR/cba.html, 2004. Department of Computer Science, The University of Liverpool, UK.

[9] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[10] M. Gartrell, U. Paquet, and N. Koenigstein. Low-rank factorization of determinantal point processes for recommendation. *arXiv preprint arXiv:1602.05436*, 2016.

[11] J. Gillenwater. *Approximate inference for determinantal point processes*. PhD thesis, University of Pennsylvania, 2014.

[12] J. A. Gillenwater, A. Kulesza, E. Fox, and B. Taskar. Expectation-maximization for learning determinantal point processes. In *NIPS*, pages 3149–3157, 2014.

[13] P. Gopalan, J. M. Hofman, and D. M. Blei. Scalable recommendation with hierarchical Poisson factorization. In *UAI*, 2015.

[14] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[15] S. Kotsiantis and D. Kanellopoulos. Association rules mining: A recent overview. *GESTS International Transactions on Computer Science and Engineering*, 32(1):71–82, 2006.

[16] A. Kulesza and B. Taskar. Structured determinantal point processes. In *NIPS*, pages 1171–1179, 2010.

[17] A. Kulesza and B. Taskar. Learning determinantal point processes. In *UAI*, 2011.

[18] A. Kulesza and B. Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2-3):123–286, 2012.

[19] J. T. Kwok and R. P. Adams. Priors for diversity in generative latent variable models. In *NIPS*, pages 2996–3004, 2012.

[20] J.-S. Lee, C.-H. Jun, J. Lee, and S. Kim. Classification-based collaborative filtering using market basket data. *Expert Systems with Applications*, 29(3):700–704, 2005.

[21] Y. Li, J. Hu, C. Zhai, and Y. Chen. Improving one-class collaborative filtering by incorporating rich user information. In *CIKM*, pages 959–968, 2010.

[22] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *KDD*, 1998.

[23] O. Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, pages 83–122, 1975.

[24] Z. Mariet and S. Sra. Fixed-point algorithms for learning determinantal point processes. In *ICML*, pages 2389–2397, 2015.

[25] A. Mild and T. Reutterer. An improved collaborative filtering approach for predicting cross-category purchases based on binary market basket data. *Journal of Retailing and Consumer Services*, 10(3):123–133, 2003.

[26] R. M. Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162, 2011.

[27] U. Paquet and N. Koenigstein. One-class collaborative filtering with random graphs. In *WWW*, pages 999–1008, 2013.

[28] H. Steck. Item popularity and recommendation accuracy. In *RecSys*, pages 125–132, 2011.