Wasserstein Learning of Determinantal Point Processes

Lucas Anquetil [*]
Criteo AI Lab
lucas.anquetil@insa-rouen.fr

Mike Gartrell
Criteo AI Lab
m.gartrell@criteo.com

Alain Rakotomamonjy
Criteo AI Lab
and University of Rouen
a.rakotomamonjy@criteo.com

Ugo Tanielian
Criteo AI Lab
and Sorbonne University
u.tanielian@criteo.com

Clément Calauzènes
Criteo AI Lab
c.calauzenes@criteo.com

Abstract

Determinantal point processes (DPPs) have received significant attention as an elegant probabilistic model for discrete subset selection. Most prior work on DPP learning focuses on maximum likelihood estimation (MLE). While efficient and scalable, MLE approaches do not leverage any subset similarity information and may fail to recover the true generative distribution of discrete data. In this work, by deriving a differentiable relaxation of a DPP sampling algorithm, we present a novel approach for learning DPPs that minimizes the Wasserstein distance between the model and data composed of observed subsets. Through an evaluation on a real-world dataset, we show that our Wasserstein learning approach provides significantly improved predictive performance on a generative task compared to DPPs trained using MLE.

1 Introduction

Generative models have enjoyed a great deal of success in the recent years due to their ability to capture insights from data distributions. Those models have generally been applied to continuous data by training them using maximum likelihood estimation (MLE) or, more recently, using adversarial learning with the well-known Generative Adversarial Networks framework [12, 15].

When dealing with discrete data, generative models trained with MLE suffer from a bias due to the asymmetrical definition of MLE. Equivalent to minimizing a Kullback Leibler divergence, the MLE cost function pays extremely low cost for generating low-quality samples. Consequently, a generative model trained by MLE tends to cover the full data distribution at the expense of covering unnecessary regions [2, 23]. On the other hand, when considering adversarial learning of discrete generative models, one usually exploits the gradient of the discriminator's loss when optimizing the generator. However, since the gradient computation requires backpropagation through the generator's output, i.e. the data, adversarial approaches are difficult to apply when generating discrete data. Depending on the generative model and the structure of the data, there are some ways to overcome this issue. For instance, [19] was the first to define a sampling scheme with the use of a Gumbel softmax distribution, and several generalizations of this softmax trick have been recently proposed in the literature [26, 14].

In this work, we address the problem of training a determinantal point process (DPP), a probabilistic model for subsets drawn from a large collection of items. A DPP parameterizes a probability

[*] Currently at INSA Rouen.

1st Workshop on Learning Meets Combinatorial Algorithms @ NeurIPS 2020, Vancouver, Canada.

distribution over the combinatorial space of subsets of elements drawn from $\mathcal{J}$, which is a discrete space composed of M distinct items. DPPs are appealing models for this setting, since they are known to also capture interactions between elements within subsets. More importantly, they offer efficient polynomial-time algorithms for most probabilistic inference operations over the space of 2^M possible subsets, such as normalization, learning, and sampling [10, 18]. In order to move away from the standard MLE learning framework [8, 22], which may suffer from the flaw described above, we define a new learning scheme for DPPs based on the minimization of the Wasserstein distance between the samples generated by the DPP and the training data. Compared to MLE, one of the main benefits of this Wasserstein-based approach is that it allows us to define a transportation cost function (*e.g* a Jaccard distance) that induces a bias on the assumed structure of the space of subsets. Minimizing this cost function allows the learning to take into account differences between pairs of subsets and to reduce the distance between subsets based on their similarities. We argue that this Wasserstein-based scheme leverages more information from the data and results in a better approximation of the target distribution.

The contributions of this work are the following: **1.)** We present a new framework when learning DPPs that minimizes the Wasserstein distance between the DPP and data composed of observed subsets. This framework can be applied to any generative probabilistic model for discrete sets. **2.)** Leveraging recent work on a DPP sampling algorithm with computational complexity that is sublinear in the size of the ground set [3], and stochastic softmax tricks for gradient estimation of discrete distributions [14], we present a differentiable DPP sampling algorithm that can scale to large ground sets. **3.)** We evaluate our Wasserstein learning approach on a real-world dataset, and show substantial improvements in predictive performance compared to DPPs trained using MLE. This experimental evaluation is one of the first to focus on a generative modeling task for DPPs.

2 Background and related work

Determinantal Point Processes Consider a finite set $\mathcal{J} = \{1, 2, \dots, M\}$ of cardinality M, which we will also denote by $[\![M]\!]$. A DPP defines a probability distribution over all 2^M subsets. It is parameterized by a matrix $\boldsymbol{L} \in \mathbb{R}^{M \times M}$, called the *kernel*, such that the probability of each subset $J \subseteq [\![M]\!]$ is proportional to the determinant of its corresponding principal submatrix: $\Pr(Y) \propto \det(\boldsymbol{L}_J)$, where $\boldsymbol{L}_J = [\boldsymbol{L}_{ij}]_{i,j \in J}$ is the submatrix of $\boldsymbol{L}$ indexed by J. The normalization constant for this distribution can be expressed as a single $M \times M$ determinant: $\sum_{J \subseteq [\![M]\!]} \det(\boldsymbol{L}_J) = \det(\boldsymbol{L} + \boldsymbol{I})$ [18, Theorem 2.1]. Therefore, $\Pr(J) = \det(\boldsymbol{L}_J) / \det(\boldsymbol{L} + \boldsymbol{I})$.

In order to ensure that the DPP defines a probability distribution, all principal minors of $\boldsymbol{L}$ must be non-negative: $\det(\boldsymbol{L}_J) \geq 0$. Matrices that satisfy this property are called P_0-matrices [4, Definition 1]. Several decompositions of $\boldsymbol{L}$ that partially cover the P_0 space are known. One common decomposition that covers the space of symmetric P_0-matrices exploits the fact that $\boldsymbol{L} \in P_0$ if $\boldsymbol{L}$ is positive semidefinite (PSD) [29]. Any symmetric PSD matrix can be written as the Gramian matrix of some set of vectors: $\boldsymbol{L} := \boldsymbol{V}\boldsymbol{V}^\top$, where $\boldsymbol{V} \in \mathbb{R}^{M \times K}$. We restrict our work in this paper to such symmetric DPPs with this decomposition, since efficient sampling algorithms, such as [3], are only available for symmetric DPPs. There are decompositions of $\boldsymbol{L}$ that partially cover the nonsymmetric P_0 [9]; we leave an investigation of Wasserstein learning of nonsymmetric DPPs for future work.

In this work we use the DPP-VFX sampling algorithm [3], which has computational complexity sublinear in M, and is therefore one of the most efficient exact sampling methods for DPPs. DPP-VFX relies on a connection between ridge leverage scores [1] and DPPs to implement a distortion-free intermediate sampling method that enables this sublinear time complexity. Since DPP-VFX requires a base DPP sampling algorithm, we propose to use the Cholesky-based DPP sampling approach [20, 28].

Estimating gradients in discrete settings The Wasserstein learning approach requires computing gradients over discrete subset samples drawn from a DPP. Two families of approaches for discrete gradient estimation are score function estimators, such as REINFORCE [30], and continuous relaxations of discrete distributions, most of which are based on the Gumbel-Max trick [21]. REINFORCE has the drawback of high variance, making it impractical in many cases. While techniques for variance reduction exist [24], they often involve highly engineered control variates. Relaxed gradient estimators incorporate bias in order to reduce variance, and are often easier to implement [26]. We choose the relaxation approach, and leverage recent work on stochastic softmax tricks [19, 26], which is a unified framework for structured relaxations of discrete combinatorial distributions. In particular, we use stochastic softmax tricks to develop a differentiable version of the DPP-VFX sampling algorithm, with a differentiable version of the Choleksy-based approach as the base DPP sampling algorithm. As far as we are aware, this is the first instance of a differentiable DPP sampling algorithm.

3 Learning DPPs via Wasserstein minimization

The classical approach for learning a DPP kernel given a collection of subsets is to maximize the likelihood of data samples drawn from the same distribution as the one used for obtaining training examples [8, 9]. One advantage of optimizing the (log) likelihood is that the likelihood of samples has a closed form expression with respect to the model parameters. Since that expression is continuously differentiable, a gradient ascent algorithm is a natural solution for solving the problem. Instead of likelihood maximization, we propose a DPP learning approach that minimizes the Wasserstein distance between the training data and samples generated by the model. This optimization scheme seeks to improve the approximation of the generative distribution of the data.

The Wasserstein distance is a distance between probability distributions defined on a given metric space. We let $\mathcal{X}_n = \{x_1, \cdots, x_n\}$ denote the training dataset of size n with empirical distribution $\mu = \sum_{i=1}^{n} a_i \delta_{\boldsymbol{x}_i}$, where δ refers to the Dirac distribution, and $\mathcal{Y}_n = \{y_1, \cdots, y_n\}$ be the collection of n sets sampled from the DPP model with distribution $\nu = \sum_{i=1}^{m} b_i \delta_{\boldsymbol{y}_i}$, where the a_i and b_i follow a uniform distribution. Given a transportation cost d defined on $2^M \times 2^M$, the Wasserstein distance between μ and ν seeks an optimal coupling P defined on $[1, n]^2$ that minimizes the cost of transporting mass from μ to ν [27]. When dealing with discrete sets of items, we argue that the use of the Jaccard distance [17] as as transportation cost function is a good choice. The Jaccard distance between two sets takes into account both the difference in length and in the items chosen: $d_J(X, Y) = (|X \cup Y| - |X \cap Y|)/|X \cup Y|$, where $X, Y \in 2^M$. Since the cost function needs to be differentiable, we use a differentiable proxy for Jaccard distance. For $x \in \mathcal{X}_n$ and $y \in \mathcal{Y}_n$, the differentiable Jaccard distance d_S is defined as follows:

$$d_S(\boldsymbol{x}, \boldsymbol{y}) = 1 - \frac{\boldsymbol{x}^\top \boldsymbol{y}}{M - (1 - \boldsymbol{x})^\top (1 - \boldsymbol{y})} \;, \tag{1}$$

where $\boldsymbol{x}, \in \{0, 1\}^M$ is a binary indicator vector and $\boldsymbol{y} \in [0, 1]^M$ is a continuous relaxation of a binary vector, with $y_k, k \in [1, M]$ being the inclusion probability of item k in the sample. By combining the definition of the Wasserstein distance with the chosen cost function in (1), we define the following Wasserstein optimization problem for DPPs:

$$\underset{\boldsymbol{V} \in \mathbb{R}^{M \times K}}{\operatorname{argmin}} \sum_{i,j=1}^{n} P^\star_{i,j} d_S(\boldsymbol{x}_i, \boldsymbol{y}_j) + \alpha \|\boldsymbol{V}\|_F^2 \quad \text{with} \quad P^\star = \underset{P \in \Pi(\mu,\nu)}{\operatorname{argmin}} \sum_{i,j=1}^{n} P_{i,j} d_S(\boldsymbol{x}_i, \boldsymbol{y}_j) \;, \tag{2}$$

where $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ is the training data, $\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n\}$ is a collection of n subsets drawn from the DPP, and $\alpha \geq 0$ is a tunable hyperparameter for regularization. Recall that we use the decomposition $\boldsymbol{L} = \boldsymbol{V}\boldsymbol{V}^\top$ for the DPP kernel.

Algorithm 1 Wasserstein learning

```
  Input: training data, $\boldsymbol{V} \in \mathbb{R}^{M \times K}$, maxIter
  for maxIter steps do
      Sample subsets from training data.
      Sample subsets from DPP.
      Compute $P^\star$ in Eq. 2 with [6].
      Update $\boldsymbol{V}$ using Eq. 2
  end for
```

For the sake of completeness, the algorithm used to solve this optimization scheme is described in Algorithm 1. Solving the optimization problem defined in Eq. 2 with backpropagation requires computing the gradient on minibatches with respect to the parameters $\boldsymbol{V}$, and thus a differentiable sampling algorithm is needed. We use an estimation of the Wasserstein distance on a minibatch [5] by computing a Earth Mover distance, and consider a new differentiable formulation of the DPP-VFX sampling algorithm [3], shown in Algorithm 2 in Appendix A. We apply the Gumbel softmax trick to the base Poisson, and multinomial, and Bernoulli sampling steps in DPP-VFX. Combined with a differentiable DPP Cholesky-based sampler (Algorithm 6), this sampler generates continuous relaxations of binary indicator vectors for subsets; see Appendix A for details.

4 Experiments

We perform experiments on the Amazon Baby Registries dataset. This dataset consists of registries or "baskets" of baby products, and has been used in prior work on DPP learning [7, 9, 11, 22]. The

Table 1: Wasserstein distance (WD), and test log-likelihood (test ll) for all datasets, for the symmetric DPP (SDPP), nonsymmetric DPP (NDPP), and the Wasserstein DPP (WDPP). WD results show 95% confidence estimates obtained via bootstrapping. Bold values indicate the best performance.

	Amazon: Apparel ($M = 100$)			**Amazon: Diaper** ($M = 100$)			**Amazon: Feeding** ($M = 100$)		
Metric	SDPP	NDPP	WDPP	SDPP	NDPP	WDPP	SDPP	NDPP	WDPP
WD	0.76 ± 0.01	0.76 ± 0.01	**0.58** ± 0.01	0.72 ± 0.01	0.73 ± 0.01	**0.63** ± 0.01	0.69 ± 0.01	0.69 ± 0.01	**0.65** ± 0.01
Test ll	-10.09	-9.60	-17.78	-10.54	-9.98	-14.27	-12.13	-11.67	-17.65

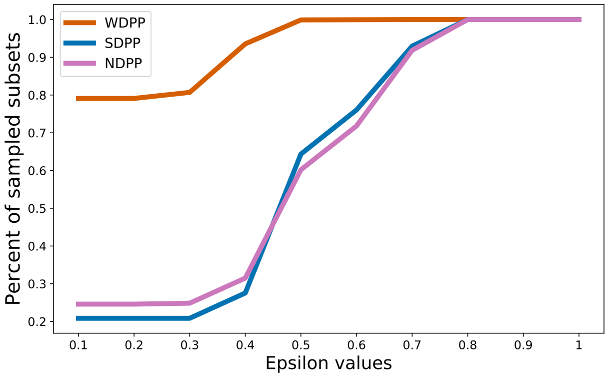

Figure 1: Precision plot for the generated subsets from each model that have a Jaccard distance of at most ϵ with at least one subset in the test set, for $\epsilon \in (0, 1]$, for the Amazon apparel dataset.

registries contain items from 15 different categories, such as "apparel", with a catalog of up to 100 items per category. We evaluate on the most popular apparel category, which contains 14,970 registries, as well as the popular diaper and feeding categories.

4.1 Setup and evaluation metrics

A small set consisting of 300 randomly-selected baskets is kept for validation, and a further random selection of 2000 baskets is used for testing. We implement our models using PyTorch [25]; Adam [16] is used for optimization, in conjunction with the solver from the POT package [6].

We use the low-rank symmetric DPP (SDPP) [8] and the low-rank nonsymmetric DPP (NDPP) [9], both trained using MLE, as baseline models for all experiments. We evaluate these baselines and our Wasserstein DPP model (WDPP) model on a subset generation task, where we estimate the Wasserstein distance (WD) between subsets sampled from the model and subsets in the test set by computing the Earth Mover's distance between these two subset collections using POT [6].

4.2 Results

Consistent with prior work, we see that the MLE NDPP outperforms the MLE SDPP on the test log-likelihood metric. However, we also observe that MLE is not directly connected to the generative task, and higher performance on test log-likelihood does not result in higher performance on the WD metric. As expected, since the Wasserstein learning approach directly optimizes a proxy for the generative task, the WDPP model significantly outperforms the baseline models in terms of WD. To provide some evidence of the connection between the WD metric and the quality of generated subsets, Fig. 1 shows the percentage of generated subsets from each model that have a Jaccard distance of at most ϵ with at least one subset in the test set, for $\epsilon \in (0, 1]$. For any given ϵ, we see that WDPP outperforms MLE models. This highlights that the WDPP, by being able to take the Jaccard distance into account, exploits the underlying structure of the combinatorial space 2^M, while the MLE-trained models do not and thus treat all subsets as completely different. We present additional experimental results in Appendix C. These results provide further evidence that, compared to MLE DPPs, the WDPP model recovers significantly more structure, and is able to generate subsets that are substantially closer to observed data.

5 Conclusion

We have presented a new Wasserstein learning approach for DPPs. Unlike conventional MLE learning approaches for DPPs, this learning approach optimizes a proxy for discrete subset generation. Empirical results indicate that the proposed approach leads to substantially improved generative performance compared to MLE. This approach is fully general, and can be readily applied to other families of models for discrete subsets. We leave such an investigation for future work.

References

[1] Ahmed Alaoui and Michael W Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *NIPS*, pages 775–783, 2015.

[2] Wei-Lun Chao, Boqing Gong, Kristen Grauman, and Fei Sha. Large-margin determinantal point processes. In *UAI*, pages 191–200, 2015.

[3] Michal Derezinski, Daniele Calandriello, and Michal Valko. Exact sampling of determinantal point processes with sublinear time preprocessing. In *NeurIPS*, 2019.

[4] Li Fang. On the Spectra of P- and P_0-Matrices. 1989.

[5] Kilian Fatras, Younes Zine, Rémi Flamary, Rémi Gribonval, and Nicolas Courty. Learning with minibatch wasserstein : asymptotic and gradient properties. In *AISTATS*, 2020.

[6] Rémi Flamary and Nicolas Courty. Pot python optimal transport library, 2017. URL `https://pythonot.github.io/`.

[7] Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. Bayesian low-rank determinantal point processes. In *RecSys*, 2016.

[8] Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. Low-rank factorization of Determinantal Point Processes. In *AAAI*, 2017.

[9] Mike Gartrell, Victor-Emmanuel Brunel, Elvis Dohmatob, and Syrine Krichene. Learning nonsymmetric determinantal point processes. In *NeurIPS*, 2019.

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

[11] Jennifer Gillenwater, Alex Kulesza, Emily Fox, and Ben Taskar. Expectation-maximization for learning Determinantal Point Processes. 2014.

[12] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and J. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[13] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2016.

[14] Weonyoung Joo, Dongjun Kim, Seungjae Shin, and Il-Chul Moon. Generalized gumbel-softmax gradient estimator for various discrete random variables. *arXiv preprint arXiv:2003.01847*, 2020.

[15] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pages 4401–4410, 2019.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

[17] Sven Kosub. A note on the triangle inequality for the jaccard distance. *Pattern Recognition Letters*, 120:36–38, 2019.

[18] Alex Kulesza, Ben Taskar, et al. Determinantal Point Processes for Machine Learning. *Foundations and Trends® in Machine Learning*, 2012.

[19] Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

[20] Claire Launay, Bruno Galerne, and Agnès Desolneux. Exact sampling of determinantal point processes without eigendecomposition. *Journal of Applied Probability*, 2020.

[21] Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *NIPS*, pages 3086–3094, 2014.

[22] Zelda Mariet and Suvrit Sra. Fixed-point algorithms for learning Determinantal Point Processes. In *ICML*, 2015.

[23] Zelda Mariet, Mike Gartrell, and Suvrit Sra. Learning determinantal point processes by corrective negative sampling. In *AISTATS*, pages 2251–2260, 2019.

[24] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, 2014.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019.

[26] Max B Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. Gradient estimation with stochastic softmax tricks. *arXiv preprint arXiv:2006.08063*, 2020.

[27] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[28] Jack Poulson. High-performance sampling of generic determinantal point processes. *Philosophical Transactions of the Royal Society A*, 378(2166):20190059, 2020.

[29] John E. Prussing. The Principal Minor Test for Semidefinite Matrices. *Journal of Guidance, Control, and Dynamics*, 1986.

[30] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

A Differentiable DPP-VFX Sampling Algorithm

As indicated in Section 2, we have leveraged stochastic softmax tricks (SST) [26] to develop a differentiable version of the DPP-VFX sampling algorithm [3]. Compared to other DPP sampling algorithms, DFF-VFX can be substantially faster, since it has time complexity sublinear in M. DPP-VFX uses a connection between ridge leverage scores [1] and DPPs to implement a distortion-free intermediate sampling method that enables this sublinear time complexity. The first step of the sampling algorithm downsamples the items in $[\![M]\!]$ i.i.d. with probability proportional to the ridge leverage score of each item, and then runs a conventional DPP sampling algorithm on this thinned or downsampled set of items, whose cardinality is much smaller than M. We use a differentiable version of the Cholesky-based DPP sampling algorithm [20, 28] on this thinned set of items. When downsampling the items, the first step is to select the number of items that will be kept using a Poisson sampling step, followed by a multinomial sampling step that selects the items that will be included in the downsampled set. Finally, a Bernoulli sampling step is used to perform rejection sampling, in order to ensure that the final exact DPP sample will be contained within the downsampled set.

Our differentiable DPP-VFX sampling algorithm is presented in Algorithm 2, where β is a kernel rescaling parameter that ensures that the Poisson parameter $(s * \boldsymbol{e}^{s/q})$ is equal to the catalog size M. Differentiable versions of the Poisson, multinomial, Bernoulli, and Cholesky-based samplers invoked by Algorithm 2 are shown in Algorithm 3, Algorithm 4, Algorithm 5, and Algorithm 6, respectively.

Algorithm 2 Differentiable DPP-VFX sublinear sampling S $\sim$ DPP($\boldsymbol{L}$)

```
Input: $\boldsymbol{L} \in \mathbb{R}^{M \times M}$, $\beta > 0$
Initialization: $\boldsymbol{L} \leftarrow \beta * \boldsymbol{L}$, $\boldsymbol{K} \leftarrow \boldsymbol{I} - (\boldsymbol{L} + \boldsymbol{I})^{-1}$
$l_i \leftarrow K_{ii} \approx Pr(i \in S)$, $s \leftarrow \sum_i l_i$
if $s > 1$ then $q \leftarrow s^2$ else $q \leftarrow s$
$\widetilde{\boldsymbol{L}} \leftarrow \frac{s}{q}[\frac{1}{\sqrt{l_i l_j}} L_{i,j}]_{i,j}$
Downsampling: $Acc \leftarrow$ False
while not$Acc$ do
    $t \sim$ SST-Poisson($s * \boldsymbol{e}^{s/q}$) (Algorithm 3)
    $\sigma_1, ..., \sigma_t \overset{i.i.d}{\sim}$ SST-Multinomial$\left(\frac{l_1}{s}, ..., \frac{l_n}{s}\right)$ (Algorithm 4)
    $Acc \sim$ SST-Bernoulli $\left(\frac{\boldsymbol{e}^s \det(\boldsymbol{I}+\widetilde{\boldsymbol{L}}_\sigma)}{\boldsymbol{e}^{ts/q} \det(\boldsymbol{I}+\widetilde{\boldsymbol{L}})}\right)$ (Algorithm 5)
end while
{Sample from thinned item catalog:} $\widetilde{S} \sim$ DPP($\widetilde{\boldsymbol{L}}_\sigma$) (Algorithm 6)
return $\boldsymbol{S} = \{\sigma_i : i \in \widetilde{S}\}$
```

Algorithm 3 SST-Poisson sampling

```
Input: $\lambda$, temperature $\tau$
STEP 1: {Truncate the total support and compute the probabilities of the integers from 1 to $2 * \lambda$}
massLogProb $\leftarrow$ $\log(\text{Poisson}_\lambda(i))$ for i in $[0, ..., 2 * \lambda]$
STEP 2: {Differentiable sampling using the Gumbel Softmax trick over the massLogProb log mass probability distribution}
oneHotSample $\sim$ GumbelSoftmax$_\tau$(massLogProb)
STEP 3: {Rearrange the one-hot-vector sample into the desired output format using matrix operations}
sstPoissonSample $\leftarrow$ $\boldsymbol{I}\cdot$ oneHotSample
return sstPoissonSample
```

A.1 Gumbel-Softmax trick

Our differentiable DPP sampling approach relies on the Gumbel-Softmax reparameterization trick [13], which is an efficient gradient estimator that replaces the non-differentiable sample from a discrete distribution with a differentiable sample from a Gumbel-Softmax distribution. The Gumbel-Max trick provides a simple and efficient way to draw samples $\boldsymbol{z}$ from a discrete distribution with

Algorithm 4 SST-Multinomial sampling

```
**Input:** masslogprob, nbsample, upperbound, temperature $\tau$
**STEP 1:** {Sample upperbound differentiable multinomial samples}
allSamples $\leftarrow$ [GumbelSoftmax$_\tau$(masslogprob)] **for** i **in** [0,...,upperbound]
**STEP 2:** {Select nbsample unique samples from allSamples}
uniqueMultinomialSamples $\leftarrow$ unique(allSamples)
**return** uniqueMultinomialSamples
```

Algorithm 5 SST-Bernoulli sampling

```
**Input:** value, temperature $\tau$
**STEP 1:** Sample a uniform value and build a massLogProb out of the two values
randValue $\leftarrow$ uniform(0, 1)
massLogProb $\leftarrow$ $[\log(\text{value}), \log(\text{randValue})]$, {massLogProb $\in \mathbb{R}^2$}
**STEP 2:** {Differentiable sampling using the Gumbel Softmax trick over the massLogProb log mass probability distribution}
oneHotSample $\leftarrow$ GumbelSoftmax$_\tau$(massLogProb)
**return** oneHotSample[0]
```

Algorithm 6 Differentiable DPP Cholesky linear sampling S $\sim$ DPP($\boldsymbol{L}$)

```
**Input:** $\boldsymbol{L} \in \mathbb{R}^{M \times M}$, temperature $\tau$
$\boldsymbol{K} \leftarrow \boldsymbol{I} - (\boldsymbol{L} + \boldsymbol{I})^{-1}$
$S \leftarrow []$
**for each** item $i$ **in** catalog **do**
    itemValue $\sim$Differentiable-Bernoulli($K_{\text{i,i}}$) (Algorithm 5)
    {Add 0 or soft-value to S:}
    S $\leftarrow$ S + binary(itemValue) $*$ sigmoid(itemValue$/\tau$)
    {Update the kernel according to the item sample:}
    $K_{\text{i,i}} \leftarrow K_{\text{i,i}} - (1 - \text{binary(itemValue)})$
    $K_{\text{[i+1:M],i}} \leftarrow K_{\text{[i+1:M],i}} / K_{\text{i,i}}$
    $K_{\text{[i+1:M],[i+1:M]}} \leftarrow K_{\text{[i+1:M],[i+1:M]}} - K_{\text{[i+1:M],i}} \otimes K_{\text{i,[i+1:M]}}$
**end for**
**return** $S$
```

class probabilities π_i :

$$\boldsymbol{z} = \text{oneHot}\left(\underset{i}{\text{argmax}}[g_i + \log(\pi_i)]\right) \tag{3}$$

where $g_1 ... g_k$ are i.i.d samples drawn from Gumbel(0, 1). The softmax is used as a continuous, differentiable approximation to argmax, and generates k-dimensional sample vectors $\boldsymbol{y} \in \Delta^{k-1}$, where each component y_i is:

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau} \text{ for } i = 1, ..., k, \tag{4}$$

where τ is the temperature hyperparameter. As the softmax temperature τ approaches 0, samples from the Gumbel-Softmax distribution become one-hot and the Gumbel-Softmax distribution becomes identical to the categorical distribution $p(z)$.

B Hyperparameters for experiments in Table 1

We perform a grid search using a held-out validation set to select the best performing hyperparameters for each model and dataset. The hyperparameter settings used for each model and dataset are described below.

Baseline MLE SDPP [7]. For this model, we use K for the number of item feature dimensions for the symmetric component $\boldsymbol{V}$, and α for the regularization hyperparameter for $\boldsymbol{V}$. We use the following hyperparameter settings:

- All datasets: $K = 30, \alpha = 0$, batch-size = 200.

Baseline MLE NDPP [9]. For this model, to ensure consistency with the notation used in [9], we use D to denote the number of item feature dimensions for the symmetric component $\boldsymbol{V}$, and D' to denote the number of item feature dimensions for the nonsymmetric components, $\boldsymbol{B}$ and $\boldsymbol{C}$. As described in [9], α is the regularization hyperparameter for the $\boldsymbol{V}$, while β and γ are the regularization hyperparameters for $\boldsymbol{B}$ and $\boldsymbol{C}$, respectively. We use the following hyperparameter settings:

- All datasets: $D = D' = 30, \alpha = \beta = \gamma = 0$, batch-size = 200.

WDPP (ours). We use K to denote the number of item feature dimensions for $\boldsymbol{V}$. α is the regularization hyperparameter. τ_C, τ_P, τ_M and τ_B are the temperature hyperparameters for Cholesky-based DPP sampling, stochastic softmax trick (SST) Poisson sampling, SST multinomial sampling, and the SST Bernoulli sampling, respectively. We use the following hyperparameter settings:

- All datasets: $K = 30$, $\alpha = 0.01$, $\tau_C = \tau_P = 0.1$, $\tau_M = 1$, $\tau_B = 10^{-8}$, batch-size = 400.

During WDPP training, we anneal both the learning rate and α.

C Additional Experimental Results

Fig. 2 shows a plot of the kernels learned by the MLE SDPP and WDPP models for the Amazon feeding dataset. We see more apparent structure in the WDPP kernel, suggesting that our Wasserstein learning approach allows the DPP to capture more structure from the data than when trained using MLE. In Fig. 3 we compare a portion of the empirical marginal item distribution with the marginals captured by the MLE SDPP and WDPP models when trained on the Amazon diaper dataset. We see that WDPP appears to learn a better approximation of the true marginal distribution of the items in the data. Finally, Table 2 shows a collection of some of the most common non-singleton subsets (modes) from the test set, and samples generated by the WDPP, SDPP, and NDPP models, for the Amazon apparel dataset. Compared to the DPPs trained by MLE, we see that our WDPP model generates subsets that are much closer to subsets found in the empirical test set.

Table 2: Most common subsets in the empirical test set and samples generated by the WDPP, MLE SDPP, and MLE NDPP models, for the Amazon apparel dataset.

Most represented Test subsets	Most represented sampled subsets for WDPP	Most represented sampled subsets for SDPP	Most represented sampled subsets for NDPP
(1, 12)	**(1, 12)**	(1, 9)	(1, 9)
(12, 23)	**(11, 12)**	(9, 20)	(9, 20)
(12, 22)	**(12, 26)**	(9, 21)	(9, 64)
(12, 50)	**(2, 12)**	(9, 64)	(9, 28, 78)
(2, 12)	**(12, 23)**	(9, 88)	(9, 37)
(12, 57)	**(12, 22)**	(9, 28)	(9, 43)
(12, 26)	**(12, 57)**	(1, 8)	(9, 19)
(11, 12)	(12, 39)	(9, 95)	(9, 49)
(4, 12)	(3, 12)	(9, 66)	(9, 24)
(31, 82)	(1, 22)	(9, 54)	(17, 28)

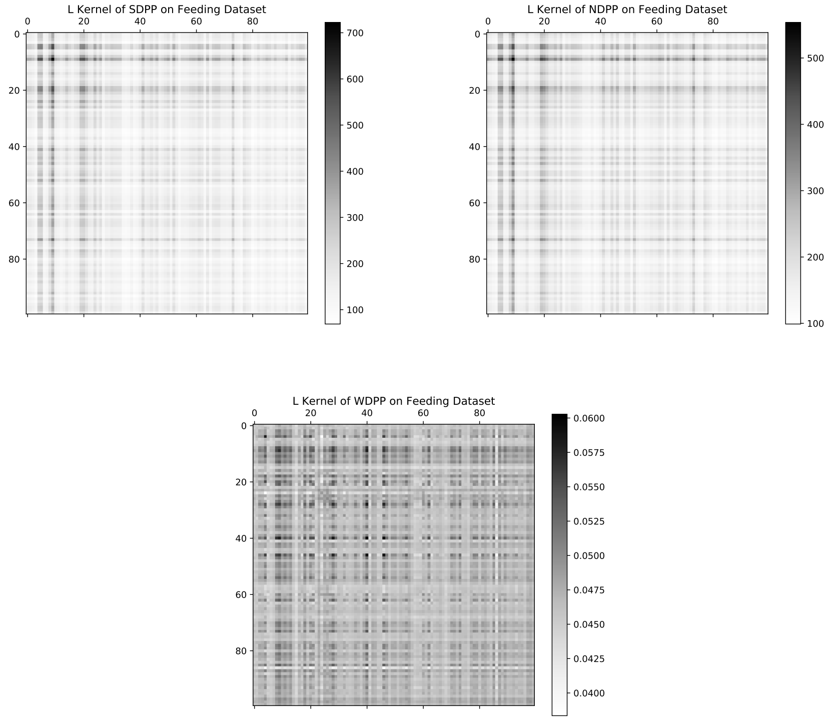

Figure 2: Comparison of the learned DPP kernels for the MLE SDPP, MLE NDPP, and WDPP models, for the Amazon feeding dataset.

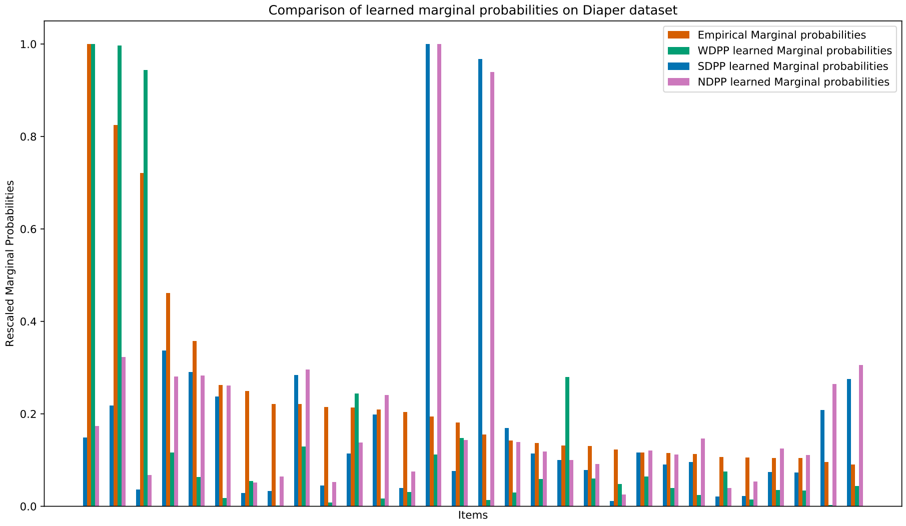

Figure 3: Comparison of the empirical marginal probabilities to the learned marginal probabilities captured by the SDPP and WDPP models, for the Amazon diaper dataset.