

Low-variance estimation in the Plackett-Luce model via quasi-Monte Carlo sampling

Alexander Buchholz
buchhola@amazon.com
Amazon Music ML
Berlin, Germany

Yannik Stein
syannik@amazon.com
Amazon Music ML
Berlin, Germany

Jan Malte Lichtenberg
jlichten@amazon.com
Amazon Music ML
Berlin, Germany

Vito Bellini
vitob@amazon.com
Amazon Music ML
Berlin, Germany

Giuseppe Di Benedetto
bgiusep@amazon.com
Amazon Music ML
Berlin, Germany

Matteo Ruffini
ruffinim@amazon.com
Amazon Music ML
Berlin, Germany

ABSTRACT

The Plackett-Luce (PL) model is popular in learning-to-rank (LTR) because it provides a useful and intuitive probabilistic model for sampling ranked lists. Counterfactual offline evaluation and optimization of ranking metrics are pivotal for using LTR methods in production. When adopting the PL model as a ranking policy, both tasks require the computation of expectations with respect to the model. These are usually approximated via Monte-Carlo (MC) sampling, since the combinatorial scaling in the number of items to be ranked makes their analytical computation intractable. Despite recent advances in improving the computational efficiency of the sampling process via the Gumbel top-k trick [23], the MC estimates can suffer from high variance. We develop a novel approach to producing more sample-efficient estimators of expectations in the PL model by combining the Gumbel top-k trick with quasi-Monte Carlo (QMC) sampling, a well-established technique for variance reduction. We illustrate our findings both theoretically and empirically using real-world recommendation data from Amazon Music and the Yahoo learning-to-rank challenge.

KEYWORDS

Policy gradient, quasi-Monte Carlo, learning-to-rank, Plackett-Luce

1 INTRODUCTION

In streaming media services with enormous catalogs, it is paramount to provide customers with personalized content, so that they can readily access media they are most likely to engage with. Often content is displayed in the form of widgets, ranked from top to bottom in the home page, containing for instance songs from the same music genre or artist. Many approaches to solve these ranking problems are based on the Plackett-Luce model [28, 36], a probabilistic model for ranked lists, that given relevance scores, performs repeated sampling from a softmax distribution. Removing drawn items at each turn, the model constructs a list where items most likely to be sampled appear in top positions. As such it has been widely used in economics and statistics [5, 16, 21, 24] as well as in recommender systems and information retrieval [9, 13, 38]. Despite the intuitive way of sampling from the model, computing expectations with respect to the PL distribution is a challenging task. The computational burden comes from enumerating all possible combinations of list entries, and it leads to a complexity of

$O(k!)$ where k is the size of the list. Expectations with respect to the PL model are required, for example, when computing utilities of rankings and their gradients [31, 38] or propensities, needed for counterfactual offline policy evaluation [26].

Monte Carlo sampling allows to approximate resulting expectation but suffers from high variance of the resulting estimator. As a remedy to the mentioned problems we suggest the use of quasi-Monte Carlo (QMC) [10], a well known variance reduction technique. Approximating expectations with QMC can lead to mean squared error rates of up to $O(1/N^2)$ compared with the Monte Carlo (MC) rate of $O(1/N)$ where N is the number of samples.

In what follows we will illustrate how the reduced variance that comes from QMC sampling leads to better estimation of propensities as well as more precise utilities and gradients in ranking models that are based on the PL model. We will combine our approach with the Gumbel top-k trick [23, 31] to generate efficient, low variance samples from the PL distribution. Our contributions consist in

- Introducing QMC to the field of learning-to-rank (LTR) and highlighting its ease of use in a wide range of settings;
- Deriving theoretical guarantees that show trade-offs between query batch sizes and Monte Carlo sample sizes when using QMC for LTR;
- Showing practical gains through experiments on the Yahoo LTR data [11] and production logs from Amazon Music.

The rest of our work is structured as follows. In Section 2 we discuss how our contribution relates to recent work, Section 3 provides required background and sets the stage for our suggested approach in Section 4, Section 5 shows our experimental results and finally Section 6 concludes.

2 RELATED WORK

Plackett-Luce model and learning-to-rank. The PL model has seen growing attention in the machine learning community over recent years [12, 19]. Used inside ranking models, the PL model leads to robust performance in industrial settings due to the probabilistic nature of the model [6]. This is a quality that also leads to desirable exploration properties due to the implicit quantification of uncertainty [32, 33]. However, efficiently training the ranking objective is a computationally difficult task [45], in particular because policy gradients [44] can suffer from high variance, which consequently has led to improved gradient computation procedures [30, 31].

Quasi-Monte Carlo sampling for machine learning. Quasi-Monte Carlo (QMC) is a variance reduction technique that has seen wide adoption in finance [18] and found its way into the statistics [7, 25] and machine learning community [4, 8, 27, 43]. Its use can improve training of gradient based optimization methods that rely on sampling from a parametric distribution. To the best of our knowledge, we are not aware of any usage for learning-to-rank problems so far.

3 BACKGROUND

We recall here the building blocks for our suggested method.

3.1 Offline learning and evaluation using propensities

In learning-to-rank a common objective is to find the best policy π^* that maximizes expected utility in a policy class $\pi \in \Pi$

$$\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}_{q \sim Q} [U(\pi|q)], \quad (1)$$

where queries q (list of items to rank) are sampled from a query distribution Q . The utility of a policy given a query is defined as

$$U(\pi|q) = \mathbb{E}_{r \sim \pi(r|q)} [\Delta(r, \text{rel}^q)], \quad (2)$$

where r denotes a ranked list, rel^q denotes the relevance of the items in query q and $\Delta(\cdot, \cdot)$ is a ranking loss such as DCG or NDCG. See [38] for more details. In practical settings, we often only dispose of data coming from a different *logging policy* h and hence off-policy learning and evaluation methods are needed [26]. We rewrite (2) as

$$U(\pi|q) = \mathbb{E}_{r \sim h(r|q)} \left[\Delta(r, \text{rel}^q) \frac{\pi(r|q)}{h(r|q)} \right], \quad (3)$$

using an importance sampling identity, see [1, 22, 40] for more details. Depending on the assumptions made on the underlying click models we either use $h(r|q)$ directly or the propensity that item $i \in r$ is shown in position k , denoted by $h(i, k|q) = \mathbb{P}(\text{rank}(i) = k)$. If logged propensities are not available, they can be approximated using Monte Carlo using sampling from the policy h in order to get

$$\mathbb{P}(\text{rank}(i) = k) \approx \sum_{n=1}^N \mathbb{1}\{\text{rank}(i_n) = k\} / N. \quad (4)$$

Then, $\hat{h} \approx h$ is used inside (3), see also [3]. Policy evaluation or training can be done on these approximate utilities.

3.2 Plackett-Luce sampling and the policy-gradient ranking algorithm

A natural policy over rankings is the Plackett-Luce policy π_θ , defined as a product of softmax distributions with scores $s_i = h_\theta(x_i^q)$, parametrized by a scoring function h_θ (e.g. neural network), which takes query-item features x_i^q as input. Optimizing the policy is achieved by finding $\theta^* = \arg \max_{\theta} \mathbb{E}_{q \sim Q} [U(\pi_\theta|q)]$, typically by using gradient-based methods. However, differentiating the previous expression is difficult as the utility is itself an integral with respect to the policy π_θ . The log derivative trick provides a solution [44]. The gradient of the utility can be approximated as

$$\nabla_{\theta} \mathbb{E}_{q \sim Q} [U(\pi_\theta|q)] \approx \frac{1}{QN} \sum_{q=1}^Q \sum_{n=1}^N \nabla_{\theta} \log \pi_\theta(r_{n,q}|q) \Delta(r_{n,q}, \text{rel}^q), \quad (5)$$

where for every query q we draw N rankings from the Plackett-Luce policy $\pi_\theta(\cdot|q)$. Estimated gradients are then used inside well established stochastic gradient descent methods where we use query batches of size Q and N samples from the PL model per query.

This framework has two potential issues: first, the score function gradient typically has high variance, which can lead to slow training [37]. Second, sampling from the PL model can become costly for huge query sizes. We will show an efficient solution consisting in the combination of QMC with the Gumbel top-k trick [23].

3.3 Sampling from the PL-model and the Gumbel top-k trick

Sampling from the PL distribution can be achieved through repeated softmax sampling where the last selected item is removed from the list. This naive approach has complexity $O(n_q^2)$ due to the repeated normalization of the softmax. The Gumbel top-k trick allows to sample from the PL distribution with log-linear complexity and was popularized by [23] and used in the ranking literature by [6, 13, 31]. A sample r can be obtained by first sampling $u_1, \dots, u_{|q|} \sim \mathcal{U}[0, 1]$, followed by computing standard Gumbel random variables $g_i = -\log(-\log(u_i)) \forall i$ and finally sorting the Gumbel variables summed with the scores. We obtain $r = \arg \text{sort}_i (s_i + g_i)$ and thus the required $r \sim \pi(\cdot|q)$.

3.4 Quasi-Monte Carlo

Quasi-Monte Carlo [10, 18] is a numeric integration technique that has seen extensive use in statistics and machine learning over the last years. Analogously to Monte Carlo sampling the goal is to approximate the integral $\int_{[0,1]^d} f(u) du = I$. Standard Monte Carlo sampling achieves this by formulating the above problem as integral $I = \mathbb{E}[f(U)]$, where $U \sim \mathcal{U}[0, 1]^d$. By the law of large numbers $I_N = \sum_{n=1}^N f(u_n) / N$, where $u_n \sim \mathcal{U}[0, 1]^d$ for $n \in 1, \dots, N$, converges to I as $N \rightarrow \infty$. This convergence happens at a rate of $\text{Var}(I_N) = O(1/N)$. Going beyond the uniform distribution, the above approach can be applied to almost arbitrary distributions $X \sim \mathbb{P}$ by exploiting the fact that $X = \Gamma(U)$ and Γ is the inverse cdf of X and hence $I_N = \sum_{n=1}^N f(\Gamma(u_n)) / N$ approximates $\mathbb{E}[f(X)]$.

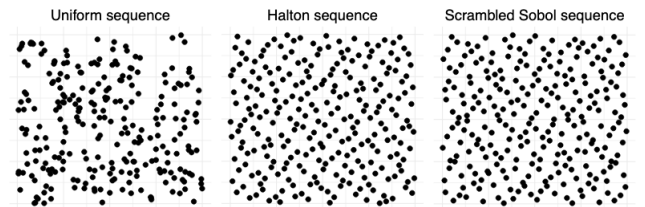


Figure 1: 256 samples generated by a MC sequence (left), and two popular QMC sequences (middle and right) over $[0, 1]^2$.

Quasi-Monte Carlo builds on the above by constructing low discrepancy sequences over the unit hypercube $u_1, \dots, u_N \in [0, 1]^d$. These sequences can be thought of as being more evenly distributed than random uniform sampling. Under smoothness conditions on the function of interest $f \circ \Gamma$, quasi-Monte Carlo then achieves better approximations of I_N : the error behaves as $\mathbb{E}[|I - I_N|^2] = o(1/N)$

and thus goes to 0 at a rate faster than standard MC, see for example [17]. This rate can be as fast as $O(1/N^2)$, see [17], and empirically almost always leads to a smaller error. We illustrate three different uniform sequences in Figure 1. The left sequence corresponds to a random uniform Monte Carlo sequence and shows some typical random clustering. The middle (called Halton sequence [20]) and right sequences (called scrambled Sobol sequence [14, 39]) are more evenly distributed, hence using them for integration will result in a reduced error. In practice we use randomized QMC sequences, such as the scrambled Sobol sequence, which reintroduce stochasticity in the sequence but keep the desirable properties. Thus, expectations are well defined, the estimates are unbiased and we can use our probabilistic toolbox to compute variances. We refer to [15, 18] for more details on construction of the sequences.

When not to use quasi-Monte Carlo instead of regular MC. Despite the highlighted advantages, there are minor caveats to consider. First, QMC can suffer from a curse of dimensionality. If we integrate over high dimensional subspaces, we need more points to construct evenly spread sequences. Second, the function of interest to integrate $f \circ \Gamma$ must be sufficiently smooth, i.e., the function must at least twice integrable (the variance must exist). Third, sample sizes have to be set to 2^k for $k \in \mathbb{N}$ in order to obtain theoretical guarantees, [34]. In practice, however, this is just a minor caveat.

4 QMC FOR PL SAMPLING

We suggest to leverage the variance reduction from QMC for two purposes: (i) for obtaining low variance propensity estimates; (ii) for obtaining more precise gradients for our ranking objective in (1). The combination of these ideas leads to a straightforward, easy-to-implement (see Listing 1) variance reduction as QMC generators are available in widely used libraries such as `scipy` [41] and `pytorch` [35]. In summary we advocate to use the Gumbel top-k trick to sample N rankings for every query q by generating a QMC sequence of length N and dimension $|q|$. Estimators of the utility in (2) will thus be of reduced variance and consequently the estimated gradients in (5) will also be more precise. Our approach can also be used for estimating propensities that are required for counterfactual offline evaluation of ranking algorithms [26]. Here is a code example to show how straightforward the implementation is:

```
import numpy as np
from torch.quasirandom import SobolEngine

def mc_rank_sampling(mc_type, mc_samples, n_actions, scores):
    if mc_type == "MC":
        u = np.random.uniform(size=(mc_samples, n_actions))
    elif mc_type == "QMC":
        s = SobolEngine(dimension=n_actions, scramble=True)
        u = s.draw(mc_samples).T.numpy()
    gumbels = -np.log(-np.log(u))
    return np.argsort(scores + gumbels, axis=1)
```

Listing 1: Sample rankings from PL policy with MC and QMC

4.1 Theoretical considerations

As gets evident from (5), the gradient is an estimator that suffers from two sources of variability: the variance from the mini batch size of the queries (using Q both for the number of queries and the

query distribution) and the variance from sampling from the PL-model (with N samples per query). As such, both terms will impact the precision of the estimators, a situation that has been studied in a reverse setting by [7]. Using a similar variance decomposition we obtain the following result.

THEOREM 4.1. *Let $\hat{F}_{Q,N} = \frac{1}{QN} \sum_{q=1}^Q \sum_{n=1}^N \Delta(r_{n,q}, \text{rel}^q)$ be an estimator of $\mathbb{E}_{q \sim Q}[U(\pi_\theta|q)]$. If $r_{n,q}$ is based on samples of a random MC distribution, then*

$$\text{Var}_Q[\mathbb{E}_N[\hat{F}_{Q,N}|Q]] = O\left(\frac{1}{Q}\right), \quad (6)$$

$$\mathbb{E}_Q[\text{Var}_N[\hat{F}_{Q,N}|Q]] = O\left(\frac{1}{QN}\right). \quad (7)$$

If $r_{n,q}$ is based on samples of a transformed QMC sequence (in particular a scrambled Sobol sequence) then

$$\mathbb{E}_Q[\text{Var}_N[\hat{F}_{Q,N}|Q]] = O\left(\frac{1}{QN^2}\right), \quad (8)$$

and the first term (6) stays the same.

PROOF. The proof is based on a decomposition of variance (overloading notation of Q and N) using the law of total variance

$$\text{Var}[\hat{F}_{Q,N}] = \mathbb{E}_Q[\text{Var}_N[\hat{F}_{Q,N}|Q]] + \text{Var}_Q[\mathbb{E}_N[\hat{F}_{Q,N}|Q]].$$

We exploit that both randomized QMC and MC yield unbiased estimators per query. Then, we check that the transformation mapping u_n to $\Delta(r_{n,q}, \text{rel}^q)$ is square-integrable. This is true as the transformation is bounded due to the Gumbel top-k trick. $\square$

A similar result can be obtained for the gradient in (5). Define

$$\hat{\nabla} F_{Q,N} = \frac{1}{QN} \sum_{q=1}^Q \sum_{n=1}^N \nabla_\theta \log \pi_\theta(r_{n,q}|q) \Delta(r_{n,q}, \text{rel}^q).$$

The trace of the variance $\text{tr Var} \hat{\nabla} F_{Q,N}$ can also be decomposed using the law of total variance and we obtain equivalent statements for (6), (7) and (8). We will provide a more thorough theoretical analysis in an extended version of this paper.

As shown in [8] (without the mini-batch perspective), the resulting stochastic gradient procedure results in faster convergence of the loss function optimization. These results directly translate to the above setting.

Our result highlights two points: first, QMC will almost always result in more precise estimators due to the improved rate in (8) compared to (7). However, second, a natural limit exists as the dominating term in the decomposition is the batch size in (6). The batch size variance contribution remains even if all variance from the PL sampling is removed. In our experiment section we will highlight these two points.

5 EXPERIMENTS

We illustrate how the obtained variance reduction from QMC bring improvements in two use cases. First, increased precision when computing propensities useful for counterfactual offline evaluation, and second, we will show performance improvement when training PG-rank due to reduced gradient variance using industrial production logs from a ranking use case in Amazon Music, as well as data from the Yahoo LTR challenge.

5.1 Propensity estimation

We illustrate the improved estimation of propensities in (4) using QMC for the PL model. We fix lists of size $[5, 25, 50]$ with item scores generated according to a standard normal distribution $\mathcal{N}(0, 1)$ with a fixed seed. We simulate draws from the PL model for sample sizes going from 2^2 to 2^{10} to compute estimators of the propensities. We repeat this 200 times to assess the variance of the estimator in (4). The propensities are computed for a single query (batch size of 1). Thus, we isolate the effect that comes from the Monte Carlo estimator of the propensity. We illustrate the reduced error in estimating the propensity in Figure 2. As shown, the QMC based estimator is consistently more precise (lower mean squared error) for different list sizes (number of items) and the error decreases faster than the equivalent estimator based on MC. The more precise estimation of the propensity with larger sample sizes comes from the faster rate of QMC sampling (i.e., $1/N^2$ compares to $1/N$ for regular Monte Carlo). As the list sizes increases from 5 to 50 the achieved MSE reduction decreases. This is due to the aforementioned curse of dimensionality. As the space to cover increases in dimension, we need more QMC-points to cover the space well.

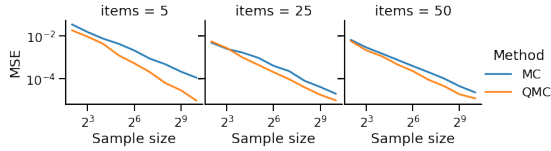


Figure 2: Propensity estimation for the PL model: MSE of MC and QMC estimates for different sample and list sizes.

5.2 Training of PG-rank

We illustrate improved training of PG rank using data from the YLTR challenge [11] and logs of a production policy used for a ranking use case in Amazon Music.

YLTR. We simulate an online learning framework by using the approach described in [2, 29] using the position bias model for simulating if a relevant item was seen. We use 100 different items to rank and set the position bias curve to $1/k$ where k is the item position. We run over a single epoch of the dataset only, using a batch size of 1000. The metrics are computed out-of-sample, meaning for every batch we computed loss, DCG and CTR before updating the model with the data from the batch, mimicking the production scenario where the model is trained online with batch updates. We use a neural network with 64 hidden units inside PG rank. The algorithms runs over one epoch of data using SGD with a learning rate of 0.1. We repeat the run 10 times to obtain confidence intervals. As illustrates Figure 3, the use of as little as 8 quasi-Monte Carlo samples speeds up learning substantially. We see an increase of the DCG, the CTR and a reduction in the loss as learning progresses.

Real world production logs. Finally, we illustrate the applicability of our approach on production data from Amazon Music. The dataset contains roughly 10^6 data points that were logged from a deterministic ranking policy. Our offline training approach uses a

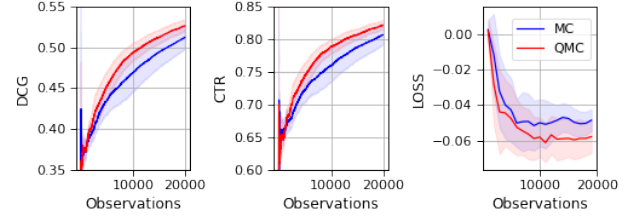


Figure 3: DCG, CTR, and loss when online training PG-rank on the Yahoo Learning to Rank challenge data.

position bias model to correct for the fact that implicit feedback of the user depends on the visibility of the content. The position bias curve was estimated using the approach described in [2]. We use the offline evaluation methodology as outlined in [26]. We train PG-rank algorithms using SGD with different step sizes using a single layer neural network of size 64 and with a learning rate of 0.001. We vary query batch sizes and Monte Carlo sampling sizes in order to disentangle their respective effect.

To avoid disclosing sensitive business information on the target metric we report relative improvements (lift in % of the target metrics) of QMC with respect to the MC version of the algorithm only. We repeat estimations 10 times in order to obtain confidence intervals.

As illustrated in Figure 4, the use of QMC instead of MC sampling leads to improvements in the training objective of up to 4%. At worst, there is no notable difference between the two algorithms. The gain decreases when we allow for more Monte-Carlo samples, which in turn increases the computational cost of the model updates. Importantly, there was no notable difference in computational time between training using MC and QMC.

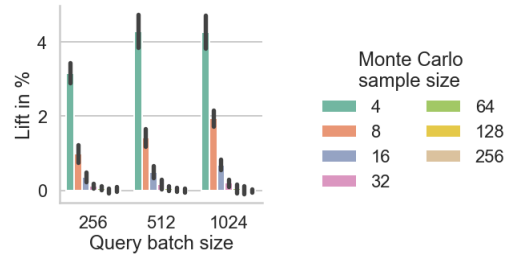


Figure 4: Lift in % of our target metric when using QMC instead of MC for training the PG-rank algorithm on production logs of Amazon Music.

6 CONCLUSION

We showed that QMC is a straightforward approach for sampling from the PL model. QMC yields low-variance, unbiased estimators of propensities and speeds up training of learning-to-rank methods as highlighted with PG-rank. This provides a strong case for using QMC consistently in production models as the implementation is easy and comes with guarantees. Potential extensions are the comparison and generalization to other ranking algorithms that use sampling from the PL model to approximate loss functions or their

gradients such as [42] or [31] and a more thorough investigation of the offline learning setting. We conjecture that in these models QMC will also lead to convergence improvements.

REFERENCES

- [1] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A general framework for counterfactual learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 5–14.
- [2] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating position bias without intrusive interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 474–482.
- [3] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W. Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. *41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018* (2018), 385–394. <https://doi.org/10.1145/3209978.3209986>
- [4] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. 2020. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in neural information processing systems* 33 (2020).
- [5] S Beggs, S Cardell, and J Hausman. 1981. Assessing the potential demand for electric cars. *Journal of Econometrics* 17, 1 (1981), 1–19. [https://doi.org/10.1016/0304-4076\(81\)90056-7](https://doi.org/10.1016/0304-4076(81)90056-7)
- [6] Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. 2020. A stochastic treatment of learning to rank scoring functions. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 61–69.
- [7] Alexander Buchholz and Nicolas Chopin. 2019. Improving Approximate Bayesian Computation via Quasi-Monte Carlo. *Journal of Computational and Graphical Statistics* 28 (2019), 205–219. Issue 1. <https://doi.org/10.1080/10618600.2018.1497511>
- [8] Alexander Buchholz, Florian Wenzel, and Stephan Mandt. 2018. Quasi-monte carlo variational inference. *35th International Conference on Machine Learning, ICLR 2018* 2 (2018), 1055–1068.
- [9] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23–581 (2010), 81.
- [10] Russel E Cafilisch. 1998. Monte carlo and quasi-monte carlo methods. *Acta numerica* 7 (1998), 1–49.
- [11] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*. PMLR, 1–24.
- [12] Weiwei Cheng, Krzysztof Dembczynski, and Eyke Hüllermeier. 2010. Label ranking methods based on the Plackett-Luce model. In *ICML*.
- [13] Fernando Diaz, Bhaskar Mitra, Michael D Ekstrand, Asia J Biega, and Ben Carterette. 2020. Evaluating stochastic rankings with expected exposure. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 275–284.
- [14] Josef Dick. 2011. Higher order scrambled digital nets achieve the optimal rate of the root mean square error for smooth integrands. *The Annals of Statistics* 39, 3 (2011), 1372–1398.
- [15] Josef Dick and Friedrich Pillichshammer. 2010. *Digital nets and sequences: discrepancy theory and quasi-Monte Carlo integration*. Cambridge University Press.
- [16] Dennis Fok, Richard Paap, and Bram Van Dijk. 2012. A rank-ordered logit model with unobserved heterogeneity in ranking capabilities. *Journal of applied econometrics* 27, 5 (2012), 831–846.
- [17] Mathieu Gerber. 2015. On integration methods based on scrambled nets of arbitrary size. *Journal of Complexity* 31, 6 (2015), 798–816.
- [18] Paul Glasserman. 2004. *Monte Carlo methods in financial engineering*. Vol. 53. Springer.
- [19] John Guiver and Edward Snelson. 2009. Bayesian inference for Plackett-Luce ranking models. In *proceedings of the 26th annual international conference on machine learning*. 377–384.
- [20] John H Halton. 1964. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM* 7, 12 (1964), 701–702.
- [21] Jerry A Hausman and Paul A Ruud. 1987. Specifying and testing econometric models for rank-ordered data. *Journal of econometrics* 34, 1–2 (1987), 83–104.
- [22] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2018. Unbiased learning-to-rank with biased feedback. *IJCAI International Joint Conference on Artificial Intelligence* 2018-July (2018), 5284–5288. <https://doi.org/10.24963/ijcai.2018/738>
- [23] Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The Gumbel-Top-k trick for sampling sequences without replacement. *36th International Conference on Machine Learning, ICLR 2019* 2019-June (2019), 6133–6145.
- [24] Gary Koop and Dale J Poirier. 1994. Rank-ordered logit models: An empirical analysis of Ontario voter preferences. *Journal of Applied Econometrics* 9, 4 (1994), 369–388.
- [25] Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. 2019. Constrained Bayesian optimization with noisy experiments. *Bayesian Analysis* 14, 2 (2019), 495–519.
- [26] Shuai Li, Yasin Abbasi-Yadkori, Branislav Kveton, S Muthukrishnan, Vishwa Vinay, and Zheng Wen. 2018. Offline evaluation of ranking policies with click models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1685–1694.
- [27] Sifan Liu and Art B Owen. 2021. Quasi-Monte Carlo Quasi-Newton in Variational Bayes. *Journal of Machine Learning Research* 22, 243 (2021), 1–23.
- [28] R Duncan Luce. 1959. Individual choice behavior. (1959).
- [29] Oriol Barbany Mayor, Vito Bellini, Alexander Buchholz, Giuseppe Di Benedetto, Diego Marco Granzio, Matteo Ruffini, and Yannik Stein. 2021. Ranker-agnostic Contextual Position Bias Estimation. *arXiv preprint arXiv:2107.13327* (2021).
- [30] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. 2020. Monte Carlo Gradient Estimation in Machine Learning. *J. Mach. Learn. Res.* 21, 132 (2020), 1–62.
- [31] Harrie Oosterhuis. 2021. Computationally Efficient Optimization of Plackett-Luce Ranking Models for Relevance and Fairness. *arXiv preprint arXiv:2105.00855* (2021).
- [32] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable unbiased online learning to rank. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1293–1302.
- [33] Harrie Oosterhuis and Maarten de Rijke. 2021. Unifying Online and Counterfactual Learning to Rank: A Novel Counterfactual Estimator that Effectively Utilizes Online Interventions. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 463–471.
- [34] Art B Owen. 2020. On dropping the first Sobol’ point. *arXiv preprint arXiv:2008.08051* (2020).
- [35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems* 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- [36] Robin L Plackett. 1975. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 24, 2 (1975), 193–202.
- [37] Rajesh Ranganath, Sean Gerrish, and David Blei. 2014. Black box variational inference. In *Artificial intelligence and statistics*. PMLR, 814–822.
- [38] Ashudeep Singh and Thorsten Joachims. 2019. Policy learning for fairness in ranking. *arXiv preprint arXiv:1902.04056* (2019).
- [39] Il’ya Meerovich Sobol’. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel’noi Matematiki i Matematicheskoi Fiziki* 7, 4 (1967), 784–802.
- [40] Adith Swaminathan and Thorsten Joachims. 2015. The self-normalized estimator for counterfactual learning. *Advances in Neural Information Processing Systems* 2015-Janua (2015), 3231–3239.
- [41] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- [42] Xuanhui Wang, Cheng Li, Nadav Golbandi, Mike Bendersky, and Marc Najork. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *Proceedings of The 27th ACM International Conference on Information and Knowledge Management (CIKM ’18)*. 1313–1322.
- [43] Florian Wenzel, Alexander Buchholz, and Stephan Mandt. 2018. Quasi-Monte Carlo Flows. *Proceedings of the 3rd Workshop on Bayesian Deep Learning, Neurips*.
- [44] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.
- [45] Hai-Tao Yu, Degen Huang, Fuji Ren, and Lishuang Li. 2022. Diagnostic Evaluation of Policy-Gradient-Based Ranking. *Electronics* 11, 1 (2022). <https://doi.org/10.3390/electronics11010037>