Adaptively Learning to Select-Rank

Jingyuan Wang

Stern School of Business, New York University

Joint work with Perry Dong, Ying Jin, Ruohan Zhan, Zhengyuan Zhou

Select-Rank in Online Platforms

Ranking algorithms are designed to organize vast quantities of information to enhance user satisfaction:

- Streaming Services: YouTube, Netflix, Disney Plus ...
- Online Retailers: Amazon, Walmart, Target ...
- Short Videos: TikTok, KuaiShou ...

Remark

Usually, in an industrial context, the ranking process is twofold:

1. the retrieval/select phase;
2. the ranking phase.

Select-Rank in Online Platforms

Ranking algorithms are designed to organize vast quantities of information to enhance user satisfaction:

- Streaming Services: YouTube, Netflix, Disney Plus ...
- Online Retailers: Amazon, Walmart, Target ...
- Short Videos: TikTok, KuaiShou ...

Remark

Usually, in an industrial context, the ranking process is twofold:

1. the retrieval/select phase;
2. the ranking phase.

Literature Review

- Learning to rank in the **bandit literature**
  1. User Click Models:
     [Chuklin et al.(2022)Chuklin, Markov, and De Rijke,
     Zhong et al.(2021)Zhong, Chueng, and Tan,
     Katariya et al.(2016)Katariya, Kveton, Szepesvari, and Wen]
  2. Position-Based Models:
     [Lagrée et al.(2016)Lagrée, Vernade, and Cappe,
     Komiyama et al.(2017)Komiyama, Honda, and Takeda,
     Lattimore et al.(2018)Lattimore, Kveton, Li, and Szepesvari]
- Large-scale ranking algorithms with **"explore-then-commit"**
  [Liu et al.(2009), Cao et al.(2007)Cao, Qin, Liu, Tsai, and Li,
  Lee and Lin(2014), Li et al.(2007)Li, Wu, and Burges,
  Li and Lin(2006), Burges(2010),
  Li et al.(2024)Li, Feng, and Chen]

Problem Setup in Bandit Setting

Consider an online platform that hosts N items, and displays ordered K items for each customer, for a total of T periods.

At every time period $t \in [T]$,

1. a user arrives with a context $X_t \in \mathbb{R}^d$
2. ranking agent chooses the retrieved K items $s_t(X_t)$: $s_t(X_t) = (q_t(1), \cdots, q_t(K))$
3. ranking agent decides and displays the ordered K items σ_t
4. ranking agent sees the user satisfaction $r(X_t, \sigma_t)$

Goals

1. Personalized Retrieval: choose K items from N items
2. Personalized Ranking: optimally rank the retrieved K items

Problem Setup in Bandit Setting

Consider an online platform that hosts N items, and displays ordered K items for each customer, for a total of T periods.

At every time period $t \in [T]$,

1. a user arrives with a context $X_t \in \mathbb{R}^d$
2. ranking agent chooses the retrieved K items $s_t(X_t)$: $s_t(X_t) = (q_t(1), \cdots, q_t(K))$
3. ranking agent decides and displays the ordered K items σ_t
4. ranking agent sees the user satisfaction $r(X_t, \sigma_t)$

Goals

1. Personalized Retrieval: choose K items from N items
2. Personalized Ranking: optimally rank the retrieved K items

Problem Setup in Bandit Setting

Consider an online platform that hosts N items, and displays ordered K items for each customer, for a total of T periods.

At every time period $t \in [T]$,

1. a user arrives with a context $X_t \in \mathbb{R}^d$
2. ranking agent chooses the retrieved K items $s_t(X_t)$: $s_t(X_t) = (q_t(1), \cdots, q_t(K))$
3. ranking agent decides and displays the ordered K items σ_t
4. ranking agent sees the user satisfaction $r(X_t, \sigma_t)$

Goals

1. Personalized Retrieval: choose K items from N items
2. Personalized Ranking: optimally rank the retrieved K items

Challenges and Our Solution Framework

Challenges

1. Estimating user satisfaction in the face of uncertainty
2. Optimally choosing from a total of $\binom{N}{K}K!$ ranking options

Solution Framework

1. Reward estimation via exploration-based bandit algorithm
2. Optimal ranking via solution to matching problem

Challenges and Our Solution Framework

Challenges

1. Estimating user satisfaction in the face of uncertainty
2. Optimally choosing from a total of $\binom{N}{K}K!$ ranking options

Solution Framework

1. Reward estimation via exploration-based bandit algorithm
2. Optimal ranking via solution to matching problem

User Satisfaction Model as a Generalized Linear Model

At any time t, for each item j ranked in position k, let $Y_{t,j,k}$ be the potential outcome of the user satisfaction with this item.

$$\mathbb{P}(Y_{t,j,k}|X_t; j, k) = h(Y_{t,j,k}, \tau) \exp\left(\frac{Y_{t,j,k}(\alpha_j k + {\beta_j}^T X_t) - A(\alpha_j k + {\beta_j}^T X_t)}{d(\tau)}\right)$$

- $h(\cdot), d(\cdot), A(\cdot)$ are the known specified functions
- τ is the known scale parameter
- $\beta_j \in \mathbb{R}^d$ is the unknown embedding of item j
- $\alpha_j \in \mathbb{R}$ is the unknown position effect of item j

User Satisfaction Model as a Generalized Linear Model

At any time t, for each item j ranked in position k, let $Y_{t,j,k}$ be the potential outcome of the user satisfaction with this item.

$$\mathbb{P}(Y_{t,j,k}|X_t; j, k) = h(Y_{t,j,k}, \tau) \exp\left(\frac{Y_{t,j,k}(\alpha_j k + \beta_j{}^T X_t) - A(\alpha_j k + \beta_j{}^T X_t)}{d(\tau)}\right)$$

- $h(\cdot), d(\cdot), A(\cdot)$ are the known specified functions
- τ is the known scale parameter
- $\beta_j \in \mathbb{R}^d$ is the unknown embedding of item j
- $\alpha_j \in \mathbb{R}$ is the unknown position effect of item j

User Satisfaction Model as a Generalized Linear Model

At any time t, for each item j ranked in position k, let $Y_{t,j,k}$ be the potential outcome of the user satisfaction with this item.

$$\begin{aligned}&\mathbb{P}(Y_{t,j,k}|X_t; j,k)\\ =&h(Y_{t,j,k},\tau)\exp\left(\frac{Y_{t,j,k}(\alpha_j k+\beta_j{}^T X_t)-A(\alpha_j k+\beta_j{}^T X_t)}{d(\tau)}\right)\end{aligned}$$

Remark

For the learning purpose, we are interested in *estimating the item-specific parameters* β_j and α_j:

$$\mu_j(X_t,k):=\mathbb{E}[Y_{t,j,k}|X_t,j,k]=A'(\alpha_j k+\beta_j{}^T X_t).$$

User Satisfaction Model as a Neural Network

At any time t, for each item j ranked in position k, let $Y_{t,j,k}$ be the potential outcome of the user satisfaction with this item.

$$\mathbb{P}(Y_{t,j,k}|X_t, j, k) = Sigmoid\Big(f^{(k)}(X_t; \theta_j)\Big)$$

- $f^{(k)}$ is the logit of reward probability at position k:

$$f^{(k)}(X_t; \theta) = \sqrt{m} W_L \Sigma\Big(W_{L-1}\Sigma\big(\ldots\Sigma(W_1 X_t)\big)\Big)$$

- X_t is the context information
- θ_j is the true parameters in the reward function of item j:

$$\theta_j = [vec(W_1^{(j)}), \ldots, vec(W_L^{(j)})]$$

User Satisfaction Model as a Neural Network

At any time t, for each item j ranked in position k, let $Y_{t,j,k}$ be the potential outcome of the user satisfaction with this item.

$$\mathbb{P}(Y_{t,j,k}|X_t, j, k) = Sigmoid\big(f^{(k)}(X_t;\theta_j)\big)$$

- $f^{(k)}$ is the logit of reward probability at position k:

$$f^{(k)}(X_t;\theta) = \sqrt{m}W_L\Sigma\Big(W_{L-1}\Sigma\big(\dots\Sigma(W_1X_t)\big)\Big)$$

- X_t is the context information
- θ_j is the true parameters in the reward function of item j:

$$\theta_j = [vec(W_1^{(j)}), \dots, vec(W_L^{(j)})]$$

User Satisfaction Model as a Neural Network

At any time t, for each item j ranked in position k, let $Y_{t,j,k}$ be the potential outcome of the user satisfaction with this item.

$$\mathbb{P}(Y_{t,j,k}|X_t, j, k) = Sigmoid\big(f^{(k)}(X_t; \theta_j)\big)$$

Remark

For the learning purpose, we are interested in *estimating the item-specific network parameters* θ_j.

Additive Reward Structure

Given a ranking $\sigma_t = (\sigma_t(1), \ldots, \sigma_t(K))$, we assume the expected user satisfaction of the ranked list is additive:

$$r(X_t, \sigma_t) = \sum_{k=1}^{K} \mu_{q_t(k)}(X_t, \sigma_t(k)).$$

Example (Watchtime).

Streaming services optimize total user watchtime.

Example (Revenue).

Online retailers maximize total revenue of the displayed items.

Additive Reward Structure

Given a ranking $\sigma_t = (\sigma_t(1), \ldots, \sigma_t(K))$, we assume the expected user satisfaction of the ranked list is additive:

$$r(X_t, \sigma_t) = \sum_{k=1}^{K} \mu_{q_t(k)}(X_t, \sigma_t(k)).$$

Example (Watchtime).

Streaming services optimize total user watchtime.

Example (Revenue).

Online retailers maximize total revenue of the displayed items.

Additive Reward Structure

Given a ranking $\sigma_t = (\sigma_t(1), \ldots, \sigma_t(K))$, we assume the expected user satisfaction of the ranked list is additive:

$$r(X_t, \sigma_t) = \sum_{k=1}^{K} \mu_{q_t(k)}(X_t, \sigma_t(k)).$$

Example (Watchtime).

Streaming services optimize total user watchtime.

Example (Revenue).

Online retailers maximize total revenue of the displayed items.

Upper Confidence Ranking (UCR)

To adaptively learn to rank in the bandit setting, we follow the principle of "optimism in the face of uncertainty".

Specifically, at any time period $t \in [T]$, the ranking agent

1. estimates the upper confidence bound $U_t(X_t, \sigma)$ of the expected user satisfaction $r(X_t, \sigma)$ for each possible ranking;
2. selects the optimal ranking σ_t:

$$\sigma_t = \arg\max_{\sigma} \left\{ U_t(X_t, \sigma) \right\}.$$

Upper Confidence Ranking (UCR)

To adaptively learn to rank in the bandit setting, we follow the principle of "optimism in the face of uncertainty".
Specifically, at any time period $t \in [T]$, the ranking agent

1. estimates the upper confidence bound $U_t(X_t, \sigma)$ of the expected user satisfaction $r(X_t, \sigma)$ for each possible ranking;
2. selects the optimal ranking σ_t:

$$\sigma_t = \arg\max_{\sigma} \left\{ U_t(X_t, \sigma) \right\}.$$

Constructing Upper Confidence Bounds [Li et al.(2017)Li, Lu, and Zhou]

Maximum Likelihood Estimation (MLE) $\hat{\theta}_{t,j} := (\hat{\alpha}_{t,j}, \hat{\beta}_{t,j})$
Action Vector $z_{t,q_t^{-1}(j)} := (\sigma_t(q_t^{-1}(j)), X_t)$
Covariance Matrix $V_j^{(t)} := \sum_{\tau=1}^{t} \mathbb{1}\{j \in s(X_\tau)\} \cdot z_{\tau,j} z_{\tau,j}^\top$

- Upper Confidence Bound of $\mu_j(X_t, k)$, i.e. $\sigma_t(q_t^{-1}(j)) = k$:

$$\hat{\mu}^U_{t,j}(X_t, k) := A'(\underbrace{\hat{\theta}_{t,j}^T z_{t,k}}_{\text{reward estimation}} + \underbrace{\xi \|z_{t,k}\|_{(V_j^{(t)})^{-1}}}_{\text{exploration term}})$$

- Upper Confidence Bound of $r(x_t, \sigma_t)$:

$$\hat{U}_t(X_t, \sigma_t) := \sum_{k=1}^{K} \hat{\mu}^U_{t,q_t(k)}(X_t, \sigma_t(k)).$$

Constructing Upper Confidence Bounds [Li et al.(2017)Li, Lu, and Zhou]

Maximum Likelihood Estimation (MLE) $\hat{\theta}_{t,j} := (\hat{\alpha}_{t,j}, \hat{\beta}_{t,j})$
Action Vector $z_{t,q_t^{-1}(j)} := (\sigma_t(q_t^{-1}(j)), X_t)$
Covariance Matrix $V_j^{(t)} := \sum_{\tau=1}^{t} \mathbb{1}\{j \in s(X_\tau)\} \cdot z_{\tau,j} z_{\tau,j}^\top$

- Upper Confidence Bound of $\mu_j(X_t, k)$, i.e. $\sigma_t(q_t^{-1}(j)) = k$:

$$\hat{\mu}_{t,j}^U(X_t, k) := A'\Big(\underbrace{\hat{\theta}_{t,j}^T z_{t,k}}_{\text{reward estimation}} + \underbrace{\xi \|z_{t,k}\|_{(V_j^{(t)})^{-1}}}_{\text{exploration term}}\Big)$$

- Upper Confidence Bound of $r(x_t, \sigma_t)$:

$$\hat{U}_t(X_t, \sigma_t) := \sum_{k=1}^{K} \hat{\mu}_{t,q_t(k)}^U(X_t, \sigma_t(k)).$$

Constructing Upper Confidence Bounds [Li et al.(2017)Li, Lu, and Zhou]

Maximum Likelihood Estimation (MLE) $\hat{\theta}_{t,j} := (\hat{\alpha}_{t,j}, \hat{\beta}_{t,j})$
Action Vector $z_{t,q_t^{-1}(j)} := (\sigma_t(q_t^{-1}(j)), X_t)$
Covariance Matrix $V_j^{(t)} := \sum_{\tau=1}^{t} \mathbb{1}\{j \in s(X_\tau)\} \cdot z_{\tau,j} z_{\tau,j}^\top$

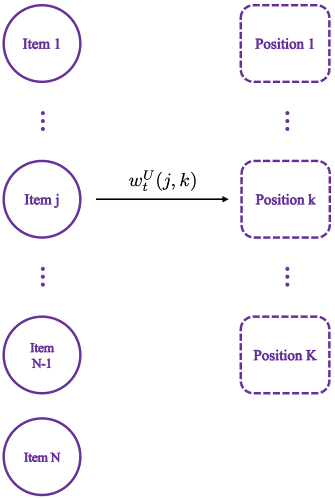
- Upper Confidence Bound of $\mu_j(X_t, k)$, i.e. $\sigma_t(q_t^{-1}(j)) = k$:

$$\hat{\mu}_{t,j}^{U}(X_t, k) := A'(\underbrace{\hat{\theta}_{t,j}^{T} z_{t,k}}_{\text{reward estimation}} + \underbrace{\xi \|z_{t,k}\|_{(V_j^{(t)})^{-1}}}_{\text{exploration term}})$$

- Upper Confidence Bound of $r(x_t, \sigma_t)$:

$$\hat{U}_t(X_t, \sigma_t) := \sum_{k=1}^{K} \hat{\mu}_{t,q_t(k)}^{U}(X_t, \sigma_t(k)).$$

Ranking via Maximum Weighted Bipartite Matching



Maximum Weighted Bipartite Matching

$$\max_{m_t} \quad \sum_{j\in[N],k\in[K]} w_t^U(j,k)m_t(j,k)$$

$$\textbf{s.t.} \quad \sum_{j\in[N]} m_t(j,k) = 1, \quad \forall k \in [K]$$

$$\sum_{k\in[K]} m_t(j,k) \le 1, \quad \forall j \in [N]$$

$$m_t(j,k) \in \{0,1\}, \quad \forall j \in [N], \forall k \in [K],$$

$$\sigma_t(j) = k \quad \Leftrightarrow \quad m_t(j,k) = 1,$$

$$s_t(X_t) = \{j \in [N] : \sum_{k\in[K]} m_t(j,k) = 1\}.$$

Maximum Weighted Bipartite Matching

$$\max_{m_t} \sum_{j\in[N],k\in[K]} w_t^U(j,k)m_t(j,k)$$

$$\textbf{s.t.} \quad \sum_{j\in[N]} m_t(j,k) = 1, \quad \forall k \in [K]$$

$$\sum_{k\in[K]} m_t(j,k) \leq 1, \quad \forall j \in [N]$$

$$m_t(j,k) \in \{0,1\}, \quad \forall j \in [N], \forall k \in [K],$$

$$\sigma_t(j) = k \quad \Leftrightarrow \quad m_t(j,k) = 1,$$

$$s_t(X_t) = \{j \in [N] : \sum_{k\in[K]} m_t(j,k) = 1\}.$$

Adaptively Learning to Rank

Algorithm 1 Upper Confidence Ranking (UCR)

Require: Environment $\mathcal{E}$, context sampling function $\mathcal{A}_X$, reward generating function $\mathcal{A}_R$, number of positions K, tuning parameter ξ, horizon T, randomization horizon T_0

```
    // Random initialization
```
for $t = 1, 2, \dots, T_0 - 1$ **do**
  Observe context $X_t \sim \mathcal{A}_X(\mathcal{E})$ and then randomly choose K items
  Sample $\sigma_t \sim \text{Unif}(S_K)$;
  Take ranking σ_t and observe outcomes $\{Y_{t,q_t(k),\sigma_t(k)}\}_{k\in[K]} \sim \mathcal{A}_R(\mathcal{E}, X_t, \sigma_t)$
end for

```
    // Upper Confidence Ranking
```
for $t = T_0, \dots, T$ **do**
  Observe context $X_t \sim \mathcal{A}_X(\mathcal{E})$
  for $j = 1, \dots, N$ **do**
    Compute $\hat{\theta}_{t,j} = (\hat{\alpha}_{t,j}, \hat{\beta}_{t,j})$ via MLE
    Compute covariance matrix $V_j^{(t)}$
    for $k = 1, \cdots, K$ **do**
      Compute $w_t^U(j,k) := g_k(A'(\hat{\theta}_{t,j}^T z_k + \xi \cdot \|z_k\|_{(V_j^t)^{-1}}))$
    end for
  end for
  Obtain $\sigma_t, s_t(X_t) \sim$ via Hungarian algorithm
  Take ranking σ_t and observe outcomes $\{Y_{t,q_t(k),\sigma_t(k)}\}_{k\in[K]} \sim \mathcal{A}_R(\mathcal{E}, X_t, \sigma_t)$
end for

Ensure: $\{(X_t, s_t(X_t), \sigma_t, Y_{t,q_t(1),\sigma_t(1)}, \dots, Y_{t,q_t(K),\sigma_t(K)})\}_{t\in[T]}$

Main Result on Cumulative Regret

Proposition

For any $\delta \in (0,1)$. If $T_0 = \max\left\{O(\frac{(K+N)^2}{N^2}\log\frac{d}{\delta}), O(d\log\frac{T}{d})\right\}$, then with probability at least $1-\delta$, for all $t \in [T_0, T]$ and all $j \in [N]$, it holds that

$$\|\hat{\theta}_{t,j} - \theta_j\|_{V_j^{(t)}} = O(\sqrt{d\log(T/d) + \log(1/\delta)}).$$

Theorem

With probability at least $1-\delta$ and proper choice of T_0, the regret

$$R_T = \tilde{O}\left((K+N)^2 + d + d\sqrt{NKT}\right).$$

Main Result on Cumulative Regret

Proposition

For any $\delta \in (0,1)$. If $T_0 = \max\left\{O(\frac{(K+N)^2}{N^2}\log\frac{d}{\delta}), O(d\log\frac{T}{d})\right\}$, then with probability at least $1-\delta$, for all $t \in [T_0, T]$ and all $j \in [N]$, it holds that

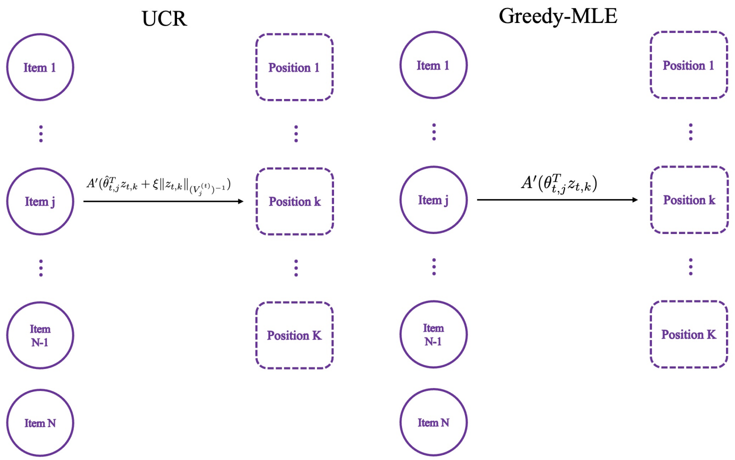
$$\|\hat{\theta}_{t,j} - \theta_j\|_{V_j^{(t)}} = O(\sqrt{d\log(T/d) + \log(1/\delta)}).$$

Theorem

With probability at least $1-\delta$ and proper choice of T_0, the regret

$$R_T = \tilde{O}\left((K+N)^2 + d + d\sqrt{NKT}\right).$$

Empirical Study and Benchmark



Experiment Results with Simulated Dateset

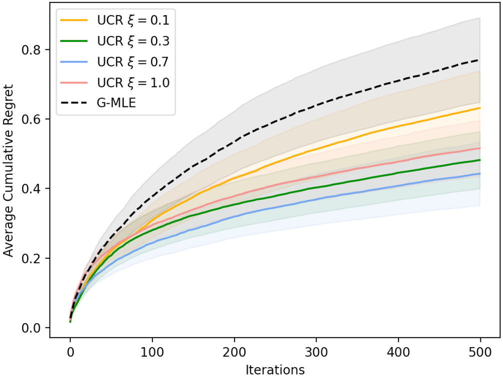


Figure 1: The average cumulative regret (with standard variation interval) of UCR and G-MLE in the simulated environment, with $N = K = 5$.

Experiment Results with Simulated Dateset

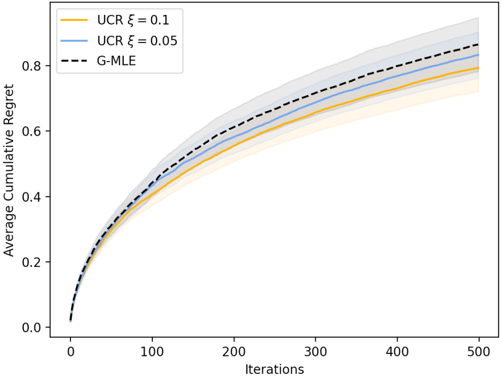


Figure 2: The average cumulative regret (with standard variation interval) of UCR and G-MLE in the simulated environment, with $N = 10, K = 5$.

Experiment Results with Real-World Dataset

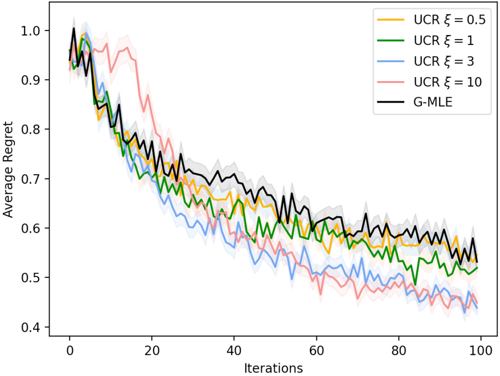


Figure 3: Average relative regret (with standard variation interval) of UCR and G-MLE on the real-world dataset, with $N = 114, K = 3$

Adaptively Learning to Select-Rank in Online Platforms

Publication:
Jingyuan, Wang, Perry Dong, Ying Jin, Ruohan Zhan, Zhengyuan Zhou. "Adaptively Learning to Select-Rank in Online Platforms." International conference on machine learning. PMLR, 2024.

Python Codes: https://github.com/arena-tools/ranking-agent

Christopher JC Burges.
From ranknet to lambdarank to lambdamart: An overview.
Learning, 11(23-581):81, 2010.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li.
Learning to rank: from pairwise approach to listwise approach.
In *Proceedings of the 24th international conference on Machine learning*, pages 129–136, 2007.

Aleksandr Chuklin, Ilya Markov, and Maarten De Rijke.
Click models for web search.
Springer Nature, 2022.

Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, and Zheng Wen.
Dcm bandits: Learning to rank with multiple clicks.
In *International Conference on Machine Learning*, pages 1215–1224. PMLR, 2016.

Junpei Komiyama, Junya Honda, and Akiko Takeda.
Position-based multiple-play bandit problem with unknown position bias.
Advances in Neural Information Processing Systems, 30, 2017.

Paul Lagrée, Claire Vernade, and Olivier Cappe.
Multiple-play bandits in the position-based model.
Advances in Neural Information Processing Systems, 29, 2016.

Tor Lattimore, Branislav Kveton, Shuai Li, and Csaba Szepesvari.
Toprank: A practical algorithm for online stochastic ranking.
Advances in Neural Information Processing Systems, 31, 2018.

Ching-Pei Lee and Chih-Jen Lin.
Large-scale linear ranksvm.
Neural computation, 26(4):781–817, 2014.

Lihong Li, Yu Lu, and Dengyong Zhou.

Provably optimal algorithms for generalized linear contextual bandits.
In *International Conference on Machine Learning*, pages 2071–2080. PMLR, 2017.

Ling Li and Hsuan-Tien Lin.
Ordinal regression by extended binary classification.
Advances in neural information processing systems, 19, 2006.

Ping Li, Qiang Wu, and Christopher Burges.
Mcrank: Learning to rank using multiple classification and gradient boosting.
Advances in neural information processing systems, 20, 2007.

Qinzhen Li, Yifan Feng, and Hongfan Kevin Chen.
Learning to rank under strategic.
Available at SSRN 4854583, 2024.

Tie-Yan Liu et al.
Learning to rank for information retrieval.

Foundations and Trends® in Information Retrieval, 3(3): 225–331, 2009.

Zixin Zhong, Wang Chi Chueng, and Vincent YF Tan.
Thompson sampling algorithms for cascading bandits.
The Journal of Machine Learning Research, 22(1):9915–9980, 2021.