Online Learning to Rank in Stochastic Click Models

Learning to Rank

- The goal of LTR is to present a ranked list of K documents out of L that maximizes the satisfaction of the user.
- This problem has been traditionally solved by training supervised learning models on manually annotated relevance judgments.
- Strong evidence suggests that the feedback of users, that is clicks, can lead to major improvements over supervised LTR methods.

Click Model

- The click model is a stochastic model of how the user examines and clicks on a list of documents.
- Different types of click models have been defined and studied for the purpose of solving LTR(Learning to Rank) problem.
- There are two prominent click models - Cascade Model and Position Based Model.

Problem with the existing approaches

- Many theoretically sound algorithms have been proposed and analyzed for finding the optimal ranked list in the cascade model (CM) and position based model(PBM).
- Recall that a click model is the manner of exploration followed by a user and we have no control over that manner.
- Problem is that if the user interacts with ranked lists using a different click model, the theoretical guarantees cease to hold.

Contribution of the paper

- Proposes stochastic click bandits, a learning framework for maximizing the expected number of clicks in online LTR in a broad class of click models, which includes both the PBM and CM.
- Proposes an algorithm called, BatchRank, that is guaranteed to learn the optimal solution in a diverse class of click models

Notation and Terminologies

- The universe of all documents is represented by ground set D = [L]
- The user is presented a ranked list, an ordered list of K documents out of L, denoted by R = (d1, d2, d3, …, dk)
- The user’s likeness for a particular document is represented by a attraction probability .
- The user expresses this likeness in the form of clicks.
- The crux of the LTR problem is the estimate these likeness values indirectly via the clicks obtained, in order to show better ranked documents.

Position-Based Model (PBM)

- In this model, given a list of documents, the user is assumed to first examine a document according to an examination probability and then click on a document based on the attraction probability.
- Thus, the expected number of clicks on list R is

$$r(\mathcal{R}) = \sum_{k=1}^{K} \chi(k)\alpha(d_k)\,.$$

Position Based Model(PBM)

- In practice, it is often observed that X(1) > X(2) > …> X (K).
- Under this assumption, the above function is maximized by the list of K most attractive items R* = (1,...,K),

Cascade Model (CM)

- In this model, given a list of documents, the user examines the list from top to bottom and depending on the attraction probability of the document clicks on it.
- Once the user has clicked a document, he stops scanning the list.
- Hence, the expected number of clicks, under this model, for a ranked list, is at-most 1.

$$\chi(\mathcal{R}, k) = \prod_{i=1}^{k-1} (1 - \alpha(d_i)) \, .$$

Cascade Model (CM)

- Like in PBM, the optimization function $$r(\mathcal{R}) = \sum_{k=1}^{K} \chi(k)\alpha(d_k)\,.$$

is maximized for getting the K most attractive documents.

$$\mathcal{R}^* = (1, \ldots, K)\,,$$

Online Learning to Rank in Click Models

- The PBM and CM are similar in many aspects.
- FIrst, both models are parameterized by L item-dependent attraction probabilities
- Secondly, the probability of clicking on the item is a product of its attraction probability and the examination probability of its position.
- Finally, the optimal solution in both models is the list of K most attractive items.

$$\mathcal{R}^* = (1, \dots, K)\,,$$

Online Learning to Rank in Click Models

- This suggests that the optimal solution in both models can be learned by a single learning algorithm, which does not know the underlying model.
- The authors approached this problem by modelling it as a multi armed bandit problem.

Stochastic Click Bandit

- This problem is defined by a four tuple (K, L, P_α, P_χ).
- Where, *K* is the number of positions, *L* is the number of items, P_α is a distribution over binary vectors $\{0, 1\}^L$, and P_χ is a distribution over binary matrices $\{0,1\}^{\Pi_K(\mathcal{D})\times K}$

Users Interaction with the Click Bandits

- Each user has an inherent attraction and examination probability distribution upon which he bases his clicks.

$$P_\alpha \otimes P_\chi$$

- Hence, the clicking and examining process is analogous to a drawing a binary attraction and examination probability vector from their probability distributions.

$$(\breve{\boldsymbol{A}}_t, \breve{\boldsymbol{X}}_t)_{t=1}^{T}$$

User interaction continued

- At each time t, the agent, i.e. the ranking program chooses a ranked list depending upon the past click history of the user.

$$\mathcal{R}_t = (\boldsymbol{d}_1^t, \ldots, \boldsymbol{d}_K^t) \in \Pi_K(\mathcal{D})$$

- The user, click on those documents for which it's click indicator, defined as the product of attraction and examination indicator is one.

$$\boldsymbol{c}_t(k) = \boldsymbol{X}_t(\mathcal{R}_t, k)\boldsymbol{A}_t(\boldsymbol{d}_k^t)$$

Reward function

- The goal of the learning agent is to maximize the number of clicks.
- Therefore, the number of clicks at time t is the reward of the agent at time t.

$$\boldsymbol{r}_t = \sum_{k=1}^{K} \boldsymbol{c}_t(k) = r(\boldsymbol{\mathcal{R}}_t, \boldsymbol{A}_t, \boldsymbol{X}_t)$$

Regret

- We evaluate the performance of a learning agent by its expected cumulative regret.

Expected Reward: $$\mathbb{E}\left[r(\mathcal{R}, \boldsymbol{A}_t, \boldsymbol{X}_t)\right] = \sum_{k=1}^{K} \chi(\mathcal{R}, k)\alpha(d_k) = r(\mathcal{R}, \alpha, \chi)\,.$$

Expected cumulative regret: $$R(T) = \mathbb{E}\left[\sum_{t=1}^{T} R(\boldsymbol{\mathcal{R}}_t, \boldsymbol{A}_t, \boldsymbol{X}_t)\right]$$

Where $R(\boldsymbol{\mathcal{R}}_t, \boldsymbol{A}_t, \boldsymbol{X}_t) = r(\mathcal{R}^*, \boldsymbol{A}_t, \boldsymbol{X}_t) - r(\boldsymbol{\mathcal{R}}_t, \boldsymbol{A}_t, \boldsymbol{X}_t)$

and $\mathcal{R}^* = \arg\max_{\mathcal{R} \in \Pi_K(\mathcal{D})} r(\mathcal{R}, \alpha, \chi)$

Algorithm: Batch Rank

Algorithm 1 `BatchRank`

// Initialization
for $b = 1, \ldots, 2K$ **do**
    for $\ell = 0, \ldots, T-1$ **do**
        for all $d \in \mathcal{D}$ **do**
            $\boldsymbol{c}_{b,\ell}(d) \leftarrow 0, \; \boldsymbol{n}_{b,\ell}(d) \leftarrow 0$

$\mathcal{A} \leftarrow \{1\}, \; b_{\max} \leftarrow 1$
$\boldsymbol{I}_1 \leftarrow (1, K), \; \boldsymbol{B}_{1,0} \leftarrow \mathcal{D}, \; \ell_1 \leftarrow 0$
for $t = 1, \ldots, T$ **do**
    for all $b \in \mathcal{A}$ **do**
        `DisplayBatch`(b, t)
    for all $b \in \mathcal{A}$ **do**
        `CollectClicks`(b, t)
    for all $b \in \mathcal{A}$ **do**
        `UpdateBatch`(b, t)

Sub-Algorithms: DisplayBatch and CollectClicks

Algorithm 2 `DisplayBatch`

Input: batch index b, time t

$\ell \leftarrow \ell_b,\ n_{\min} \leftarrow \min_{d \in \boldsymbol{B}_{b,\ell}} \boldsymbol{n}_{b,\ell}(d)$

Let $d_1, \ldots, d_{|\boldsymbol{B}_{b,\ell}|}$ be a random permutation of items $\boldsymbol{B}_{b,\ell}$ such that $\boldsymbol{n}_{b,\ell}(d_1) \leq \ldots \leq \boldsymbol{n}_{b,\ell}(d_{|\boldsymbol{B}_{b,\ell}|})$

Let $\pi \in \Pi_{\mathrm{len}(b)}([\mathrm{len}(b)])$ be a random permutation of position assignments

for $k = \boldsymbol{I}_b(1), \ldots, \boldsymbol{I}_b(2)$ **do**

    $\boldsymbol{d}_k^t \leftarrow d_{\pi(k - \boldsymbol{I}_b(1) + 1)}$

Algorithm 3 `CollectClicks`

Input: batch index b, time t

$\ell \leftarrow \ell_b,\ n_{\min} \leftarrow \min_{d \in \boldsymbol{B}_{b,\ell}} \boldsymbol{n}_{b,\ell}(d)$

for $k = \boldsymbol{I}_b(1), \ldots, \boldsymbol{I}_b(2)$ **do**

    if $\boldsymbol{n}_{b,\ell}(\boldsymbol{d}_k^t) = n_{\min}$ **then**

        $\boldsymbol{c}_{b,\ell}(\boldsymbol{d}_k^t) \leftarrow \boldsymbol{c}_{b,\ell}(\boldsymbol{d}_k^t) + \boldsymbol{c}_t(k)$

        $\boldsymbol{n}_{b,\ell}(\boldsymbol{d}_k^t) \leftarrow \boldsymbol{n}_{b,\ell}(\boldsymbol{d}_k^t) + 1$

Sub Algorithm : UpdateBatch

Algorithm 4 `UpdateBatch`

Input: batch index b, time t

// End-of-stage elimination

$\ell \leftarrow \ell_b$

if $\min_{d \in \boldsymbol{B}_{b,\ell}} \boldsymbol{n}_{b,\ell}(d) = n_\ell$ **then**

  for all $d \in \boldsymbol{B}_{b,\ell}$ **do**

    Compute $\boldsymbol{U}_{b,\ell}(d)$ and $\boldsymbol{L}_{b,\ell}(d)$

  Let $d_1, \ldots, d_{|\boldsymbol{B}_{b,\ell}|}$ be any permutation of items $\boldsymbol{B}_{b,\ell}$ such that $\boldsymbol{L}_{b,\ell}(d_1) \geq \ldots \geq \boldsymbol{L}_{b,\ell}(d_{|\boldsymbol{B}_{b,\ell}|})$

UpdateBatch

for $k = 1, \ldots, \mathrm{len}(b)$ **do**

  $B_k^+ \leftarrow \{d_1, \ldots, d_k\}$

  $B_k^- \leftarrow \boldsymbol{B}_{b,\ell} \setminus B_k^+$

// Find a split at the position with the highest index

$s \leftarrow 0$

for $k = 1, \ldots, \mathrm{len}(b) - 1$ **do**

  if $\boldsymbol{L}_{b,\ell}(d_k) > \max_{d \in B_k^-} \boldsymbol{U}_{b,\ell}(d)$ **then**

    $s \leftarrow k$

if $(s = 0)$ and $(|\boldsymbol{B}_{b,\ell}| > \mathrm{len}(b))$ **then**

  // Next elimination stage

  $\boldsymbol{B}_{b,\ell+1} \leftarrow \left\{ d \in \boldsymbol{B}_{b,\ell} : \boldsymbol{U}_{b,\ell}(d) \geq \boldsymbol{L}_{b,\ell}(d_{\mathrm{len}(b)}) \right\}$

UpdateBatch

// Next elimination stage
$\boldsymbol{B}_{b,\ell+1} \leftarrow \{d \in \boldsymbol{B}_{b,\ell} : \boldsymbol{U}_{b,\ell}(d) \geq \boldsymbol{L}_{b,\ell}(d_{\mathrm{len}(b)})\}$
$\ell_b \leftarrow \ell_b + 1$
else if $s > 0$ **then**
// Split
$\mathcal{A} \leftarrow \mathcal{A} \cup \{b_{\max} + 1, b_{\max} + 2\} \setminus \{b\}$
$\boldsymbol{I}_{b_{\max}+1} \leftarrow (\boldsymbol{I}_b(1), \boldsymbol{I}_b(1) + s - 1)$
$\boldsymbol{B}_{b_{\max}+1,0} \leftarrow B_s^+,\ \ell_{b_{\max}+1} \leftarrow 0$
$\boldsymbol{I}_{b_{\max}+2} \leftarrow (\boldsymbol{I}_b(1) + s, \boldsymbol{I}_b(2))$
$\boldsymbol{B}_{b_{\max}+2,0} \leftarrow B_s^-,\ \ell_{b_{\max}+2} \leftarrow 0$
$b_{\max} \leftarrow b_{\max} + 2$

Sub-Algorithm: UpdateBatch

Algorithm 4 `UpdateBatch`

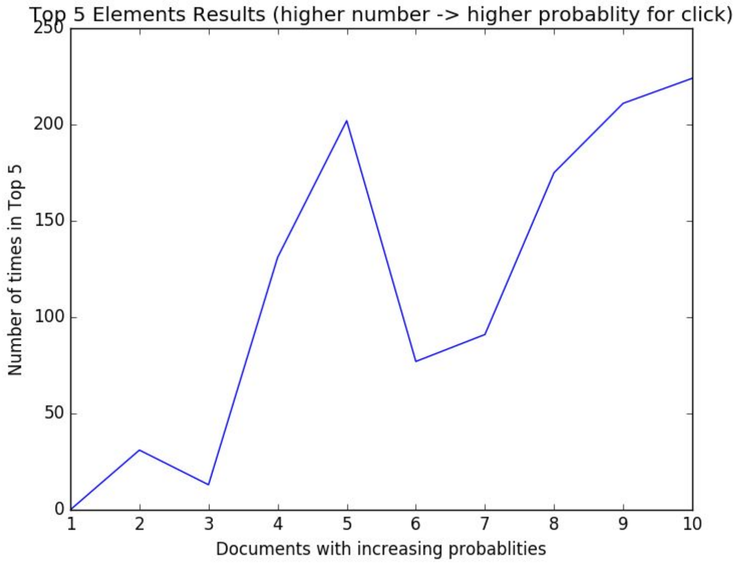
Input: batch index b, time t

// End-of-stage elimination
$\ell \leftarrow \ell_b$
if $\min_{d \in \boldsymbol{B}_{b,\ell}} \boldsymbol{n}_{b,\ell}(d) = n_\ell$ **then**
  for all $d \in \boldsymbol{B}_{b,\ell}$ **do**
    Compute $\boldsymbol{U}_{b,\ell}(d)$ and $\boldsymbol{L}_{b,\ell}(d)$
  Let $d_1, \ldots, d_{|\boldsymbol{B}_{b,\ell}|}$ be any permutation of items $\boldsymbol{B}_{b,\ell}$ such that $\boldsymbol{L}_{b,\ell}(d_1) \geq \ldots \geq \boldsymbol{L}_{b,\ell}(d_{|\boldsymbol{B}_{b,\ell}|})$
  for $k = 1, \ldots, \mathrm{len}(b)$ **do**
    $B_k^+ \leftarrow \{d_1, \ldots, d_k\}$
    $B_k^- \leftarrow \boldsymbol{B}_{b,\ell} \setminus B_k^+$

  // Find a split at the position with the highest index
  $s \leftarrow 0$
  for $k = 1, \ldots, \mathrm{len}(b) - 1$ **do**
    if $\boldsymbol{L}_{b,\ell}(d_k) > \max_{d \in B_k^-} \boldsymbol{U}_{b,\ell}(d)$ **then**
      $s \leftarrow k$
  if $(s = 0)$ and $(|\boldsymbol{B}_{b,\ell}| > \mathrm{len}(b))$ **then**
    // Next elimination stage
    $\boldsymbol{B}_{b,\ell+1} \leftarrow \left\{d \in \boldsymbol{B}_{b,\ell} : \boldsymbol{U}_{b,\ell}(d) \geq \boldsymbol{L}_{b,\ell}(d_{\mathrm{len}(b)})\right\}$
    $\ell_b \leftarrow \ell_b + 1$
  else if $s > 0$ **then**
    // Split
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{b_{\max} + 1, b_{\max} + 2\} \setminus \{b\}$
    $\boldsymbol{I}_{b_{\max}+1} \leftarrow (\boldsymbol{I}_b(1), \boldsymbol{I}_b(1) + s - 1)$
    $\boldsymbol{B}_{b_{\max}+1,0} \leftarrow B_s^+, \ \ell_{b_{\max}+1} \leftarrow 0$
    $\boldsymbol{I}_{b_{\max}+2} \leftarrow (\boldsymbol{I}_b(1) + s, \boldsymbol{I}_b(2))$
    $\boldsymbol{B}_{b_{\max}+2,0} \leftarrow B_s^-, \ \ell_{b_{\max}+2} \leftarrow 0$
    $b_{\max} \leftarrow b_{\max} + 2$

Experiments

- We experimented with documents [1,2,...,10] with probabilities [0,0.1,0.2,...,0.9] and estimated the top 5 documents after many runs. The result is the graph below.



Thanks!

Team RNA