Combinatorial Cascading Bandits

Branislav Kveton, Zheng Wen, Azin Ashkan, Csaba Szepesvari

NIPS 2015

Combinatorial Multi-armed Bandit

- The player selects a subset of arms (a super arm) to pull in each round
  - Each pulled arm generates a random reward following an unknown distribution
  - Collectively provides a random reward to the player
- Semi-bandit feedback

Combinatorial Multi-armed Bandit

- The action unit is a combinatorial object
- Goal
  - Collect cumulative reward over multiple rounds as much as possible
- Regret
  - Difference of cumulative reward of optimal solution and the cumulative reward of the bandit strategy

Combinatorial Optimization

- A binary objective function
  - Return one if all weights of chosen items are one
  - Weights: binary, stochastic, independent
- Non-linear objective function
- Only observe the index of the first chosen items with a zero weight
- Cascading:
  - Examine the selected list from the first item to the last
  - May stop before the last item
  - Not semi-bandit feedback

Application: Network routing

- Choose a routing path to maximize the probability that all links are up
- The weight of each link
  - A probability of being up
  - Independent Bernoulli random variables
- Only observe the first link that is down
- In e-mail delivery
  - Get information from SMTP

Application: Recommendation

- Choose a list of items to minimize the probability that none of them are attractive
- The weight of an item
  - This item attracts the user
  - Bernoulli distribution
- Only observe the index of the first attractive items in the list

Previous work on Combinatorial Bandit

- Semi-bandit feedback in combinatorial bandits
  - Get the feedback of all chosen items
  - More informative
- Cascading bandits
  - Choose largest sum
  - Uniform matroid
  - Any list of K items out of L is feasible
  - Exchangeable items in one solution

Problem Formulation

- Ground items $E = \{1, \ldots, L\}$
  - Individual arms
- A distribution P over a binary hypercube $\{0,1\}^E$
- Feasible set Θ
  - A set of distinct items
- Weights $(\boldsymbol{w}_t)_{t=1}^n$
  - $\boldsymbol{w}_t \in \{0,1\}^E$
  - Drawn from distribution P
  - $i.i.d.$ sequence
- Decision $A_t \in \Theta$

Problem Formulation

- A binary reward
  - $r_t = \min_{e \in A_t} w_t(e) = \wedge_{e \in A_t} w_t(e)$
  - Reward is 1 if all items in A_t are 1
- Reward function
  - $f(A, w) = \Pi_{e \in A} w(e)$
- Feedback
  - Observe the index of the first item in A_t whose weight is zero
  - $O_t = \min\{1 \leq k \leq |A_t| : w_t(a_t^k) = 0\}$

Goal

- Maximize expected cumulative reward
- Minimize the expected cumulative regret
- $R(n) = \mathbb{E}[\sum_{t=1}^{n}\{f(A^*, w_t) - f(A_t, w_t)\}]$

Assumption

- The distribution P is factored
  - $P(w) = \Pi_{e \in E} P_e\big(w(e)\big)$
  - P_e is a Bernoulli distribution with mean $\overline{w}(e)$
- $\mathbb{E}[f(A, w)] = f(A, \overline{w})$
  - Depends only on the expected weights of each individual items
- $A^* = \text{argmax}_{A \in \Theta} f(A, \overline{w})$

Algorithm

- A family of UCB algorithms
  - Rule of optimality
- Step 1: computes the upper confidence bound on the expected weights

The average of observed weights

$$\mathbf{U}_t(e) = \min\left\{\hat{\mathbf{w}}_{\mathbf{T}_{t-1}(e)}(e) + c_{t-1,\mathbf{T}_{t-1}(e)}, 1\right\}$$

Number of observed times

Radius of confidence interval

- $\overline{w}(e) \in [\widehat{w}_s(e) - c_{t,s}, \widehat{w}_s(e) + c_{t,s}]$ holds with high probability

Algorithm

- Step 2: choose the optimal solution with respect to these UCBs
  - $A_t = \mathrm{argmax}_{A \in \Theta} f(A, U_t)$
- Step 3: observe and update
  - Observe all items a_k^t such that $k \leq O_t$
  - Update based on the observed weights

Algorithm

Algorithm 1 `CombCascade` for combinatorial cascading bandits.

// Initialization
Observe $\mathbf{w}_0 \sim P$
$\forall e \in E : \mathbf{T}_0(e) \leftarrow 1$
$\forall e \in E : \hat{\mathbf{w}}_1(e) \leftarrow \mathbf{w}_0(e)$

Assume initialized by one sample

for all $t = 1, \ldots, n$ **do**
// Compute UCBs
$\forall e \in E : \mathbf{U}_t(e) = \min\left\{\hat{\mathbf{w}}_{\mathbf{T}_{t-1}(e)}(e) + c_{t-1,\mathbf{T}_{t-1}(e)}, 1\right\}$

// Solve the optimization problem and get feedback
$\mathbf{A}_t \leftarrow \arg\max_{A \in \Theta} f(A, \mathbf{U}_t)$
Observe $\mathbf{O}_t \in \{1, \ldots, |\mathbf{A}_t|, +\infty\}$

Observe as much items as possible

// Update statistics
$\forall e \in E : \mathbf{T}_t(e) \leftarrow \mathbf{T}_{t-1}(e)$

Stopping point

for all $k = 1, \ldots, \min\{\mathbf{O}_t, |\mathbf{A}_t|\}$ **do**
$e \leftarrow \mathbf{a}_k^t$
$\mathbf{T}_t(e) \leftarrow \mathbf{T}_t(e) + 1$

Update observed times

$$\hat{\mathbf{w}}_{\mathbf{T}_t(e)}(e) \leftarrow \frac{\mathbf{T}_{t-1}(e)\hat{\mathbf{w}}_{\mathbf{T}_{t-1}(e)}(e) + \mathbb{1}\{k < \mathbf{O}_t\}}{\mathbf{T}_t(e)}$$

Update average value

Properties

- Computationally efficient
  - Equivalent to find $A_t = \mathrm{argmax}_{A \in \Theta} \sum_{e \in A} \log\ U_t(e)$
  - Maximizing a linear function over feasible sets
  - Matroids, matchings, paths
- Sample efficient
  - All items are estimated separately
  - The regret does not depend on the solution space
  - Due to assumption

Disjunctive Objective

- Previous one is conjunctive
- Disjunctive model
  - Reward is one if the weight of any chosen items is one
  - $r_t = \max_{e \in A_t} w_t(e) = \vee_{e \in A_t} w_t(e)$
  - Observe the index of the first item whose weight is one
- Reward function
  - $f_\vee(A, w) = 1 - \Pi_{e \in A}(1 - w(e))$ to maximize
  - $f(A, w) = \Pi_{e \in A}\big(1 - w(e)\big)$ to minimize
- Compute the lower confidence bound

$$\mathbf{L}_t(e) = \max\left\{1 - \hat{\mathbf{w}}_{\mathbf{T}_{t-1}(e)}(e) - c_{t-1,\mathbf{T}_{t-1}(e)}, 0\right\}$$

Analysis

- Reduce to stochastic combinatorial semi-bandit problem
- The regret is divided into two parts
  - High probability confidence intervals do not hold
  - High probability confidence intervals hold

Prefix

- When $f(A, w) \ll f^*$
  - Can distinguish A from A^* without learning all items in A
  - f becomes smaller with more items
- Prefixes of suboptimal solution
  - $A = \left(a_1, \dots, a_{|A|}\right)$
  - $B = (a_1, \dots, a_k)$
- The probability of observing all items in prefix is close to opt
- The gaps are close to original solutions
  - Must exist

Prefix

- Treat prefixes as feasible solutions to original problem
- Count the number of times that the prefix can be chosen when all items in prefix can be observed
  - Instead of optimal set
- Bound non-linear by the property of product
  - Convert to summation

Proof Sketch

- Bound the regret when confidence intervals fail
  - By Chernoff bound
- Change the counted events
  - From partially-observed solutions to fully-observed prefixes
  - Condition on the history up to current round
  - Use tower rule to remove conditional expectation
  - $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$

Proof Sketch

- Counting suboptimal prefixes
  - Event that suboptimal prefix B is hard to distinguish from opt
  - To bound the second term in regret by this event
  - $f(B, U_t) \geq f(A, U_t)$ if B is a prefix of A
  - $f(A_t, U_t) \geq f(A^*, U_t)$ if the agent selects A_t
- Apply to stochastic combinatorial semi-bandits
  - Introduce infinitely-many mutually-exclusive events
  - Suboptimal prefix are not observed sufficiently often

Gap-free Upper Bound

- Previous
  - Gap-dependent upper bound
  - Between suboptimal solution and opt
- Decompose regret into two parts
- The gaps exceed threshold
  - The gaps appear in the distribution-dependent bound
- Set the threshold to make the bound sublinear

Summary

- Feedback is less informative
  - A partial monitoring problem where some of chosen items are unobserved
- Proof is novel
  - Find an intermediate variable to reduce to an existing problem

Thank you!