Almost Optimal Batch-Regret Tradeoff for linear contextual bandits

Zihan Zhang[1], Xiangyang Ji[2], Yuan Zhou[3]

1. Department of CSE, UW
2. Department of Authomation, Tsinghua
3. YMSC, Tsinghua

Motivation

- Issues in online learning
  - Policy deployment cost
  - Communication cost

Batch learning framework

- Hyperparameter: batch complexity M, number of rounds T
  - Decide $T_1, T_2, \ldots, T_M$ such that $T_1 + T_2 + \ldots + T_M = T$
  - For $i = 1,2,\ldots,M$
    - Decide π_i as the policy for the i-th batch
    - Run π_i for T_i rounds

Contextual linear bandit

- Hyperparameter: the reward kernel θ
- At each round $t = 1,2,...,T$
  - Receive the context $\{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,K}\}$ drawn from an unknown distribution D
  - Select $i_t \in [K]$ and receive the reward r_t such that $\mathbb{E}[r_t] = \mathbf{x}_{t,i_t}^\top \theta$

Regret $R_T = \sum_{t=1}^{T} (\max_{i \in K} \mathbf{x}_{t,i}^\top \theta - \mathbf{x}_{t,i_t}^\top \theta)$

Main result

Theorem.

For any contextual linear bandit problem with batch complexity as M, the minimax regret bound is (up to log factors)

$$\tilde{\Theta}\left(\min \left\{ T^{\frac{1}{2-2^{-M+2}}} d^{\frac{1-2^{-M+2}}{2-2^{-M+2}}}, T^{\frac{1}{2-2^{-M+1}}} d^{\frac{1-2^{-M+1}}{2-2^{-M+1}}} \min\{K, d\}^{\frac{2^{-M+1}}{2-2^{-M+1}}} \right\} \right)$$

,

where d is the dimension of feature, K is the number of arms and T is the number of rounds.

Main result

Corollay.

It suffices to use $O(\log\log(T))$ batches to reach the minimax optimal regret bound of $\tilde{O}(\sqrt{Td})$

Algorithm ingredients

- Elimination-based bandit learning
- Single-phase learning for exploration policy
  - Explore via reward-free LinUCB
    - Use empirical context to learn the optimal design of D
  - Scaled and clippled update rule
    - Adjust the weight of each feature vector to avoid over exploraiton

Technical ideas

- Hardness
  - Lack of knowledge of infrequent directions
  - A large burn-in time to identify the unknown distribution
- Solution
  - A two phase learning framework
  - Reward-free LinUCB with scaled and clippled update rule

Future direction

- Extend the results to linear MDP and linear mixture MDP
- Devise efficient design algorithm with single-phase learning

Thanks