Kernel Methods for Cooperative Contextual Bandits

Abhimanyu Dubey and Alex Pentland

Media Lab and Institute for Data Systems and Society (IDSS)
Massachusetts Institute of Technology

dubeya@mit.edu

ICML 2020

Motivation

- Distributed learning is an increasingly popular paradigm in ML: multiple parties collaborate to train a stronger joint model by sharing data.
- An alternative is to let data remain in a distributed setup, and have one ML algorithm (agent) for each data center, i.e., federated learning.
- Each agent can communicate with other agents to (securely) share relevant information, e.g., over a network.
- The group of all agents therefore collectively *cooperate* to solve their own learning problems.

Multi-Armed Bandits

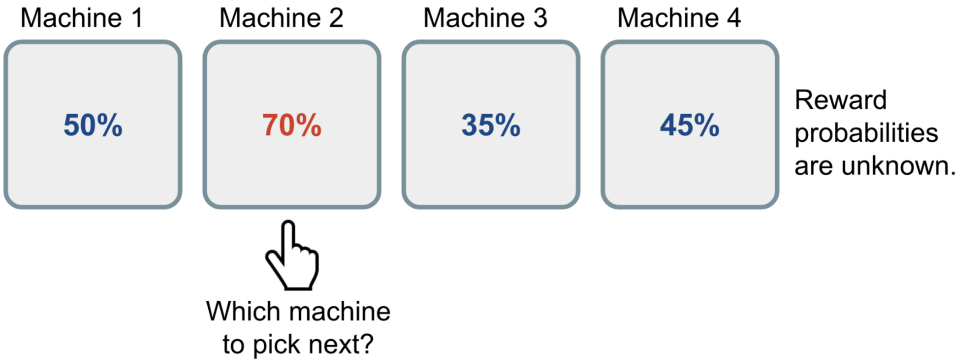

Figure: Multi-armed bandit (courtesy `lilianweng.github.io`).

The Upper Confidence Bound (UCB) Algorithm

- "Optimism in the face of uncertainty" strategy – i.e. to be optimistic about an arm when we are uncertain of its utility.
- For each of K arms, we compute

$$Q_k(t) = \underbrace{\frac{\sum_{i=1}^{n_k(t-1)} r_k^i}{n_k(t-1)}}_{\text{empirical mean}} + \underbrace{\sqrt{\frac{2\ln(t-1)}{n_k(t-1)}}}_{\text{"uncertainty"}}.$$

- Choose arm with largest $Q_k(t)$.
- This general family of algorithms has strong guarantees as well.

Basic Cooperation: UCB with Naive Averaging

- **Basic Idea**: Use observations from neighbors naively to construct Q-values.
- Assume 2 agents (A and B):

$$Q_k^A(t) = \underbrace{\frac{\sum_{i=1}^{n_k(t-1)} r_{k,A}^i + \sum_{i=1}^{n_k(t-1)} r_{k,B}^i}{n_k^A(t-1) + n_k^B(t-1)}}_{\text{mean of both agents' observations}} + \underbrace{\sqrt{\frac{2\ln(t-1)}{n_k^A(t-1) + n_k^B(t-1)}}}_{\text{smaller ``uncertainty''}}.$$

- Works well when each agent faces the **same** bandit problem.
- Can trivially be extended to other algorithms (e.g., Thompson Sampling)

Naive combinations aren't always useful

- Consider two agents A and B, each solving a 2-armed bandit problem.
  - For agent A, let the arms have mean payouts $(0.8, 0.2)$.
  - For agent B, let the arms have mean payouts $(0.2, 0.8)$.
- If each agent naively incorporated the other agents' observations, they will each have mean estimates of arms as $\approx (0.5, 0.5)$, leading to $O(T)$ regret.

Summary

- It is clear that instead of naively combining observations from neighbors, agents must intelligently "weigh" external behavior.
- Intuitively, this weighing factor would be a function of how "similar" the agents' problems are.
- For each agent, when rewards are drawn from arbitrary distributions, it is unclear how "similarity" can be measured between the distributions.
- **Summary**. In this work, we propose a framework based on Reproducing Kernel Hilbert Spaces (RKHS) to measure similarity between agent rewards, and several *near-optimal* algorithms for the cooperative contextual bandit problem using this framework.

The Contextual Bandit Problem

- At any trial $t = 1, 2, ...,$ each agent $v \in V$ is supplied a *decision set* $D_{v,t}$.
- They select an action $\boldsymbol{x}_{v,t} \in D_{v,t}$ and obtain a reward $y_{v,t}$.

$$y_{v,t} = f_v(\boldsymbol{x}_{v,t}) + \varepsilon_{v,t},$$

- The objective of the problem is to minimize the group regret:

$$R_{\mathcal{G}}(T) = \sum_{v \in V} \sum_{t=1}^{T} \left(f_v(\boldsymbol{x}^*_{v,t}) - f_v(\boldsymbol{x}_{v,t}) \right),$$

where, $\boldsymbol{x}^*_t = \arg\max_{\boldsymbol{x} \in D_{v,t}} f_v(\boldsymbol{x})$.

The Cooperative Contextual Bandit

- We assume the $|V|$ agents communicate via an undirected, connected graph $\mathcal{G} = (V, E)$, where $(i, j) \in E$ if agents i and j can communicate.
- Messages from any agent v are available to agent v' after $d(v, v') - 1$ trials of the bandit, where d is the distance between the agents in $\mathcal{G}$.
- Every trial, every agent sends the following message $\boldsymbol{m}_{v,t}$ to all its neighbors in $\mathcal{G}$:

$$\boldsymbol{m}_{v,t} = \langle t, v, \boldsymbol{x}_{v,t}, y_{v,t} \rangle$$

- This message is forwarded from agent to agent γ times (taking one trial of the bandit problem each between forwards), after which it is dropped.

Parameteric Network Contexts

- We assume that each agent v has an underlying *network context*, denoted by $\boldsymbol{z}_v$, and the reward function $f_v(\cdot)$ is parameterized by $\boldsymbol{z}_v$, i.e., for some unknown but fixed function F,

$$f_v(\boldsymbol{x}) = F(\boldsymbol{x}, \boldsymbol{z}_v) \; \forall x \in \mathcal{X}, \boldsymbol{z}_v \in \mathcal{Z}.$$

- We denote $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{z}_v)$. Furthermore, we assume that F has a bounded norm in some RKHS $\mathcal{H}$ with kernel $\widetilde{K}$ and feature $\phi(\cdot)$.
- For a given $\phi : (\mathcal{X} \times \mathcal{Z}) \to \mathbb{R}^d$ and unknown (but fixed) vector $\boldsymbol{\theta} \in \mathbb{R}^d$,

$$F(\tilde{\boldsymbol{x}}) = \phi(\tilde{\boldsymbol{x}})^T \boldsymbol{\theta}.$$

- This implies that in some higher-order feature space (kernel space), F is a linear function, and ϕ can be thought of as a "feature extractor".

Kernel Assumption

- Now, the *kernel function* $\widetilde{K}(\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2) = \phi(\tilde{\boldsymbol{x}}_1)^\top \phi(\tilde{\boldsymbol{x}}_2)$. We assume that this kernel is a composition of two separate kernels (where $\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i, \boldsymbol{z}_i)$):

$$\widetilde{K}(\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2) = \underbrace{K_z(\boldsymbol{z}_1, \boldsymbol{z}_2)}_{\text{network kernel}} \cdot \underbrace{K_x(\boldsymbol{x}_1, \boldsymbol{x}_2)}_{\text{action kernel}}.$$

- K_z provides us with a generic framework to measure similarity between agent functions, and can be learnt online when it is unknown.
- K_x can be any PSD kernel, e.g., Gaussian (RBF), Linear, deep neural network features, etc. K_z can be derived from geographical or demographic constraints (e.g., social networks).

Algorithm

- We face three challenges in algorithm design:
  - Non-identical reward functions f_v.
  - Communication delays between agents.
  - Heterogeneity (agents possess different information at all times).
- We modify the KERNEL-UCB [Valko13] algorithm as follows:
  - **Non-identical rewards**: We augment contexts $\boldsymbol{x}$ with network contexts $\boldsymbol{z}$, and use the augmented kernel $\tilde{K}$ to create the UCB.
  - **Delays**: We use a subsampling technique similar to [Weinberger02], i.e., each agent runs γ UCB instances in parallel.
  - **Heterogeneity**: We partition $\mathcal{G}$ carefully in terms of cliques to bound heterogeneity, i.e., each agent only accepts messages from a subset of nodes.

Estimating K_z

Typically, K_z may be derived from orthogonal information about the problem (e.g., social network structure). However, we may need to estimate K_z directly from samplies.

- We assume each context $\boldsymbol{x} \in \mathcal{D}_{v,t}$ is sampled from distribution $\mathcal{P}_v$.
- We define $\boldsymbol{z}_v = \Psi(\mathcal{P}_v)$, i.e., the kernel mean embedding of $\mathcal{P}_v$ under K_x.
- We then define $K_z(\boldsymbol{z}_1, \boldsymbol{z}_2)$ to be the RBF kernel with variance σ, i.e.,

$$K_z(\boldsymbol{z}_1, \boldsymbol{z}_2) = \exp\left(-\|\Psi(\mathcal{P}_1) - \Psi(\mathcal{P}_2)\|_2^2 / 2\sigma^2\right)$$

- However, $\mathcal{P}_v$ is unknown, so we replace it with the empirical kernel mean embedding $\widehat{\Psi}_t(\mathcal{P}_v)$:

$$\widehat{\Psi}_t(\mathcal{P}_v) = \sum_{\tau=1}^{t} K_x(\cdot, \boldsymbol{x}_{v,\tau}).$$

Regret Guarantees

- KERNEL-UCB for MT arms obtains pseudoregret:

$$R(T) = O\left(\sqrt{MT \log\left(\frac{\det\left(\widetilde{\boldsymbol{K}}_{MT+1} + \lambda \boldsymbol{I}\right)}{\lambda^{MT+1}}\right)}\right)$$

- Our algorithm COOP-KERNEL-UCB obtains regret (with known K_z):

$$R_{\mathcal{G}}(T) = O\left(\sqrt{MT \cdot \underbrace{\left(\bar{\chi}(\mathcal{G}_\gamma) \cdot \gamma\right)}_{\text{network overhead}} \cdot \underbrace{\rho_{\boldsymbol{z}}}_{\text{task overhead}} \cdot \log\left(\frac{\det\left(\widetilde{\boldsymbol{K}}_{MT+1} + \lambda \boldsymbol{I}\right)}{\lambda^{MT+1}}\right)}\right)$$

- $\bar{\chi}(\mathcal{G}_\gamma)$ is the clique number of the γ^{th} graph power of $\mathcal{G}$, and $\rho_{\boldsymbol{z}} = \text{rank}(K_z)$.

Regret Guarantees

- When all agents have identical f_v, $\rho_{\boldsymbol{z}} = 1$ (fully cooperative, e.g., federated learning) and when they have distinct f_v, $\rho_{\boldsymbol{z}} = M$ (no cooperation).
- When $\mathcal{G}$ is complete, $(\bar{\chi}(\mathcal{G}_\gamma) \cdot \gamma) = 1$, and $(\bar{\chi}(\mathcal{G}_\gamma) \cdot \gamma) = M$ in the worst case (line graph).
- When K_z is being estimated simultaneously (by our method), the regret obtained has an additional factor of $O(\log T)$.

Conclusion

- We study the cooperative kernel bandit problem with non-identical reward functions on networks with delays.
- Our algorithm is scalable, efficient, and provides near-optimal regret guarantees.
- Experiments on synthetic and real-world network problems demonstrate that our algorithm performs competitively.
- Future directions:
  - We use subsampling, which we believe is suboptimal and introduces an additional $\sqrt{\gamma}$ factor in the regret. Future work with more sophisticated partition techniques and analysis can shave off this factor.
  - Messages are not private, which is required for real-world federated learning.
  - Tighter bounds on the kernel Gram matrix can improve the analysis.

Thank You!
Paper ID: 281