Contextual Bandits

Akshay Krishnamurthy
akshay@cs.umass.edu

July 25, 2017

1 Introduction

Contextual bandits are a way to formulate online decision making problems that leads to statistically and computationally tractable solutions. A motivating application involves training an online personalization system to make recommendations to users. In such systems, users arrive online, with some contextual information (e.g., historical browsing data, search query, etc.) and the online system must make a decision to present some content to the user with the goal of optimizing some metric like clicks or purchases.

The main issues here are:

1. **Explore/Exploit.** We must explore the decision space to learn the best decisions. However we do not want to make bad decisions, since this will lower clicks or purchases. Thus there is tension between collecting new information, and using the information we already have to make good decisions.

2. **Generalization.** We want to be able to share information across users to lower data requirements, but still have some level of personalization. This is the main departure from multi-armed bandits.

Contextual bandits is a mathematical model for thinking about these issues. The problem is formulated as a proceeding for T rounds $1, \dots, T$. In round t:

1. The environment produces a context $x_t \in X$. (User information, browsing history, search query)

2. The algorithm chooses an action $a_t \in A$. (Recommended product)

3. The environment reacts with a reward $r_t(a_t) \in [0, 1]$. (Click/no click)

The learning goal is to accumulate lots of reward, measured by $\sum_{t=1}^{T} r_t(a_t)$. This requires learning a policy for choosing actions based on the contexts. To learn such a policy, we will provide the algorithm with a policy class $\Pi \subset X \to A$ of many such mappings. Then we will measure success by *regret* to the policy class:

$$\text{Regret} = \max_{\pi \in \Pi} \sum_{t=1}^{T} r_t(\pi(x_t)) - r_t(a_t). \tag{1}$$

We would like to minimize the regret.

The **Explore-Exploit** tradeoff is present here, since we only see the reward for the action we chose, and must explore to learn about other actions. At the same time, choosing bad actions incurs high regret. The **generalization** aspect is also present through the policy class, which allows us to share information across contexts. We want the policy class to be big, so that we learn something useful, but not too big, so that learning is tractable, which is analogous to supervised learning.

As notation, we will also have $|A| = K$ and $|\Pi| = N$. Thus the three parameters to the problem are K, N, and T, the number of rounds. We usually think of $K \ll N$.

Most of the literature cares primarily about three features: statistical efficiency (regret), computational efficiency (which we will formalize later), and assumptions on the environment. We will discuss algorithms with different tradeoffs among these three features.

2 Failures

The standard approaches in practice to these problems are:

1. **Predict the metric.** Train a regression model to predict the metric based on the contextual information and features about the decision. With such a model, we can evaluate other decision making policies to optimize our system. However, unless your regression model can accurately predict rewards (i.e., assuming realizability), this method has uncontrollable bias. Even still, this does not solve the exploration problem.

2. **Run A/B tests.** The gold-standard method is to deploy a new policy to a fraction of the traffic in an A/B test. This approach works, but is very data intensive when we have a lot of policies. Specifically, this gives us at best $N^{1/3}T^{2/3}$ regret and we can do much better.

3. **Run a bandit algorithm.** We can also run a standard multi-armed bandit algorithm. However, either we will not be able to generalize across users, but then we will not compete with the policy class, or we can treat each policy as an arm, and get $\sqrt{NT}$ regret.

3 Algorithms

Exp4 [3]. Exp4 is the first CB algorithm, which demonstrates how to avoid the pitfalls of the bandit approach. The idea is that, since there are only a few actions, you can actually learn about all the policies using randomization and *importance weighting*. At round t, we will choose a distribution $p_t(\cdot|x_t)$ over actions, conditional on the context, play $a_t \sim p_t(\cdot|x_t)$, and observe $r_t(a_t)$. We then estimate the rewards as

$$\hat{r}_t(a) = r_t(a_t)\frac{\mathbf{1}\{a_t = a\}}{p_t(a|x_t)}. \tag{2}$$

This reward estimator is great because (a) it is unbiased, and (b) it has variance K when $p_t(\cdot|x_t)$ is the uniform distribution. This leads to much better estimation of policy performance.

Exp4 just uses this estimator in a standard bandit algorithm, Exp3. Start with $w_1(\pi) = 1$ for all π and at each $t = 1, \dots, T$

1. Set $p_t(a) = (1-\gamma)\sum_\pi \frac{w_t(\pi)\mathbf{1}\{\pi(x_t)=a\}}{\sum_\pi w_t(\pi)} + \gamma/K$.

2. Play $a_t \sim p_t$, receive $r_t(a_t)$.

3. Set $w_{t+1}(\pi) = w_t(\pi)\exp(\gamma\hat{r}_t(\pi(x_t))/K)$.

This is just like Exp3 except that we project the distribution from the policy space down to the action space and use importance weighting. The algorithm works by using w to track the performance of the best policy. Whenever a policy gets high reward, we increase its weight quite dramatically. Thus, w zooms in on the policies with good performance, and we essentially just explore amongst them, which handles the exploration exploitation tradeoff.

Theorem 1. *Exp4 has expected regret $O(\sqrt{KT\log N})$ when contexts and rewards are generated by an adversary. This is optimal.*

On our three axes, Exp4 is statistically optimal and is robust to adversarial environments, which are good. However, it is computationally expensive since the running time is $O(NT)$ but the regret depends only logarithmically on N. Thus, for computational reasons, we are unable to work with large policy sets.

Explore-First [4]. We will address the computational issue by working in an oracle model of computation. Specifically we assume access to a supervised learning algorithm for the policy class, that given $D = \{(x_i, r_i)\}_i$ where $x_i \in X$ and $r_i \in \mathbb{R}^K$ can solve

$$\text{Oracle}(D) = \operatorname*{argmax}_{\pi\in\Pi} \sum_{(x,r)\in D} r(\pi(x)) \tag{3}$$

With this oracle, the Explore-First algorithm is quite straightforward. For the first τ rounds we play actions uniformly at random to collect a dataset of importance weighted rewards $(x_i, \hat{r}_i)$. Then we call the oracle once on this dataset to find $\hat{\pi}$, which we use for the remaining $T - \tau$ rounds.

Theorem 2. *Explore-First makes one oracle call, and with properly tuned τ, it has regret $O((K \log N)^{1/3} T^{2/3})$ in stochastic environments.*

Proof. After τ rounds, we have a uniform deviation bound

$$\forall \pi. |\mathbb{E} r(\pi(x)) - \frac{1}{\tau} \sum_{i=1}^{t} \hat{r}_i(\pi(x_i))| \leq \sqrt{(K \log N)/\tau}$$

Thus the regret is

$$\tau + (T - \tau)\sqrt{(K \log N)/\tau},$$

At which point optimizing for τ proves the result. □

Explore-First is statistically suboptimal, works only in stochastic environments, but is computationally tractable. One other positive is that it is conceptually simple and something that people use in practice.

ILTCB [1]. ILTCB is a more recent efficient and optimal algorithm for stochastic environments. The algorithm more optimally balances exploration and exploitation, yet still operates in the oracle model. At each round t, we compute a distribution Q_t over the policies using the past data, sample $\pi_t \sim Q_t$ and play $a_t \sim \pi_t$, with a little bit of uniform exploration mixed in. The important part is in how we compute Q_t, which should solve a program that looks something like:

$$\text{Exploit:} \quad \sum_{\pi} Q(\pi) \widehat{\text{Regret}}_t(\pi) \leq \text{small} \tag{4}$$

$$\text{Explore:} \quad \forall \pi. \mathbb{E} \frac{1}{Q(\pi(x)|x)} \leq \text{small} + \widehat{\text{Regret}}_t(\pi). \tag{5}$$

Here $\widehat{\text{Regret}}_t(\pi)$ is the empirical regret of policy π after t rounds, using the importance weighted rewards. The LHS of the second inequality is the variance of the importance weighted reward estimates for policy π, and the main point here is that we should have good coverage on all of the policies with low empirical regret. Don't worry too much about the details.

The point is that this algorithm has the following guarantee:

Theorem 3. *ILTCB has $O(\sqrt{KT \log N})$ regret and is computationally efficient with $O(\sqrt{T})$ oracle calls under stochastic environments.*

Ctx-FTPL [5]. The last algorithm is an oracle-based algorithm for an adversarial setting, but it does not achieve the optimal regret. We operate in the *transductive setting*, where we know the contexts $x_1, \ldots, x_T$ before hand, but they and the respective rewards can be adversarially chosen. The idea is to do "Follow-the-perturbed Leader" where the perturbation is via fake (x, r) pairs that we pass to the oracle. Specifically, at round t:

1. Create dataset $D_t = \{(x_i, \tilde{r}_i)\}_{i=1}^{T} \cup \{(x_i, \hat{r}_i)\}_{i=1}^{t-1}$ where $\tilde{r}_i(a) \sim \text{Laplace}(\epsilon)$. (density $f(z) = \frac{\epsilon}{2} \exp(-\epsilon|z|)$.)
2. Play $\pi_t(x_t)$ where $\pi_t = \text{Oracle}(D_t)$.

The idea here is that the perturbing the oracle is equivalent to sampling from a distribution over policies, and this distribution puts more weight on the good policies, since they have higher reward on the second part of the dataset. Thus this algorithm, in spirit, is achieving the same goal as Exp4 and ILTCB.

Theorem 4. *Ctx-FTPL has $O(T^{3/4}\sqrt{K \log N})$ regret and makes $O(T^{3/2})$ oracle calls in the adversarial transductive setting.*

4 Open Problems

Let me mention two open problems that I would be very interested in seeing a resolution.

In this lecture I showed you four algorithms and their properties can be summarized as:

Algorithm	Optimal Regret?	Computationally Efficient?	Adversarial Environment?
Exp4	+	−	+
Explore-First	−	+	−
ILTCB	+	+	−
Ctx-FTPL	−	+	+

The obvious question is:

Problem 5. *Is there a contextual bandit algorithm for the adversarial setting that is efficient and optimal?*

The second open problem involves getting sharper bounds for benign settings. If we think of losses instead of rewards $\ell(a) \in [0, 1]$ with the goal of minimizing total loss $\sum_{t=1}^{T} \ell_t(a_t)$, we have a similar setting to before. However in the full information version, we can get regret $\sqrt{L^\star \log N}$ where $L^\star = \min_\pi \sum_{t=1}^{T} \ell_t(\pi(x_t))$ is the best loss achievable. This can be much better than $\sqrt{T \log N}$ worst case bound. However it is not known whether meaningful $L^\star$ bounds are achievable for contextual bandits [2].

Problem 6. *Can we get meaningful $L^\star$ bounds for contextual bandits?*

References

[1] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert E Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *ICML*, 2014.

[2] Alekh Agarwal, Akshay Krishnamurthy, John Langford, Haipeng Luo, and Robert E Schapire. Open problem: First-order regret bounds for contextual bandits. In *COLT*, 2017.

[3] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 2002.

[4] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, 2008.

[5] Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert E Schapire. Efficient algorithms for adversarial contextual learning. In *ICML*, 2016.