STATS 710 – Seq. Decision Making with mHealth Applications Nov 8, 2016

Lecture 16: Taming the Monster: AKA How I Learned to Love Conning Bandits

Instructors: Susan Murphy and Ambuj Tewari *Scribe: Ian Fox*

1 Setup

Let's recall the contextual bandit setting

Algorithm 1 The Contextual Bandit Setting

1: **for** t=1 **to** T **do**
2: Learner receives $X_t \in \mathcal{X}$
3: Learner takes action $A_t \in \mathcal{A}$
4: Learner receives reward $R_t^{A_t}$
5: **end for**

The learner acts according to policy $\pi \in \Pi : \mathcal{X} \to \mathcal{A}$. For this lecture we will assume $|\Pi|$ is finite. We define $\pi^* = \text{argmax}_{\pi \in \Pi} V(\pi)$, where $V(\pi) = \mathbf{E}[R^{\pi(X)}] = \mathbf{E}[\sum_{a \in \mathcal{A}}^{R^a} \mathbf{1}[\pi(X) = a]]$. We assume $(X, (R_a)_{a \in \mathcal{A}}) \sim_{iid} \mathcal{D}$ where $\mathcal{D}$ is fixed but unknown.

Note: the following explanations/proofs are less complete than usual due to the complexity of the paper. Consider the following an extended, friendly abstract and look to the paper for a more complete treatment.

2 Contextual Bandits: What we Crave

The ideal contextual bandit algorithm has two properties:

1. The algorithm accesses the policy class Π only through an argmax oracle (AMO), that is, an existing algorithm that solves the problem:

$$\underset{\pi \in \Pi}{\text{argmax}} \frac{1}{|D|} \sum_{(X,(r^a)_{a \in \mathcal{A}}) \in \mathcal{D}}$$

. In other words, the AMO finds the policy that gives optimal performance on dataset D.

2. The algorithm is statistically optimal, that is, it only suffers $O(\sqrt{T})$ regret.

It's clear why property 2 is desirable, but property 1 can be equally important. This is because as the size of the policy space increases, algorithms that need to directly interact with the Π become intractable. Scalable AMO's exist, and are a convenient abstraction in any event (in CS we believe the more layers of abstraction the better).

So far we've seen algorithms that have one or the other of these properties, but not both. Epoch-Greedy satisfies property 1, but because it explicitly separates exploration and exploitation it suffers $O(T^{\frac{2}{3}})$ regret. On the other hand, EXP4 satisfies property 2, achieving $O(\sqrt{T})$ regret, but it requires enumerating all policies in Π. This is infeasible for even fairly simple policy classes.

3 ILOVETOCONBANDITS

Now that we're all fired up and ready to go, let's introduce an algorithm that satisfies both properties, introduced in [AHK+14]. First, we'll give a natural template for this algorithm, then we'll fill in the details.

Algorithm 2 Template

1: **for** t=1 **to** T **do**
2: Maintain distribution q_t over Π {Don't worry about this for now}
3: Observe $X_t \in \mathcal{X}$
4: Draw $\pi_t \sim q_t$
5: Take action $A_t = \pi_t(X_t)$ {This means $p_t(a) = \sum_{\pi \in \Pi} q_t(\pi) \mathbf{1}[\pi(X_t) = a]$}
6: Apply importance weighting:

$$\hat{V}_t(\pi) = \frac{1}{t} \sum_{s=1}^{t} \sum_{a \in \mathcal{A}} \frac{R_s^a \mathbf{1}[A_s = a] \mathbf{1}[\pi(X_s) = a]}{p_s(a)}$$

7: Update q_t to get q_{t+1}
8: **end for**

Note the estimation on line 6 is unbiased, but depending on the size of $p_s(a)$ it could have arbitrary variance. So when we go to fill in this template, we need to ensure:

1. p_s never gets too small
2. We can work with q_t without enumerating Π

Now, how do we go about choosing our q_t? We'll choose by solving the following optimization problem (OP):

$$\operatorname*{argmin}_q \sum_{\pi \in \Pi} q(\pi) \hat{Reg}_t(\pi) + k\mu_t \frac{1}{t} \sum_{j=1}^{t} KL(\mathcal{U}, q^{\mu_s}(a|X_s))$$

Where:

- k is the number of actions,
- q can be any probability distribution over the policy classes,
- $\mathcal{U}$ is the uniform distribution over actions,
- $\mu_t = \sqrt{\frac{\log(\frac{|\Pi|}{\delta})}{kt}}$, and
- $q^{\mu_s}(a|X_s) = (1 - \mu_s) \sum_{\pi \in \Pi} q(\pi) \mathbf{1}[\pi(X_s) = a] + \mu_s \mathcal{U}(a)$.
- KL is the KL-Divergence

This optimization makes intuitive sense, by minimizing $\hat{Reg}_t(\pi)$ we're exploiting our current knowledge. At the same time minimizing $KL(\mathcal{U}, q^{\mu_s}(a|X_s))$ builds in some exploration, ensuring that

our distribution doesn't stray too far from uniform. Note that as t increases the penalty incurred from the second term goes to zero, satisfying our criteria of vanishing exploration in the limit.

This instantiation of the template is the algorithm ILOVETOCONBANDITS. First we'll show that it satisfies property 2, then we'll elaborate on how this optimization is done using coordinate descent to show it satisfies property 1. It should be noted that the 'monster' algorithm which ILOVETOCONBANDITS tames was very similar, but used an elipsoid method to solve the optimization problem. This lead to nice theoretical properties, but had an even more complicated analysis and was computationally infeasible to use as it required something like $O(T^3)$ AMO calls by time T. In contrast this method only requires $O(\sqrt{T})$. Now, onto the properties

3.1 ILOVESMALLREGRET

Theorem 1. *The regret incurred by ILOVETOCONBANDITS is, with probability at least* $1-\delta$*,*

$$O(\sqrt{kT\log(\frac{T|\Pi|}{\delta})} + k\log(\frac{T|\Pi|}{\delta})$$

Very high level sketch: We assume two claims about OP (which are ensured by the solver)

1. Small empirical regret,
$$\sum_{\pi} q(\pi)\hat{Reg}_t(\pi) \leq ck\mu_t$$
2. Low variance,
$$\forall \pi \in \Pi : \frac{1}{t}\sum_{s=1}^{t} \frac{\mu_t}{(1-\mu_t)q(\pi(X)|X) + \mu_t} \leq c(k\mu_t + \hat{Reg}_t(\pi))$$

For some universal constant c (given in the paper).

Recalling that $Reg(\pi) = V(\pi^*) - V(\pi)$, we introduce the following lemma:

Lemma 2 (Key Lemma)**.**

$$\forall \pi \in \Pi : Reg(\pi) \leq 2\hat{Reg}_t(\pi) + O(K\mu_t)$$

Using this lemma, let's examine the instantaneous regret at time t:

$$\begin{aligned}
\text{Regret at time t} &= \sum_{\pi\in\Pi} q_t(\pi)Reg(\pi) \\
&\leq \sum_{\pi\in\Pi} q_t(\pi)(2\hat{Reg}_t(\pi) + O(K\mu_t)) \\
&= 2\sum_{\pi\in\Pi} q_t(\pi)\hat{Reg}_t(\pi) + q_t(\pi)O(K\mu_t) \\
&\leq O(K\mu_t) && \text{By claim 1}
\end{aligned}$$

proving the theorem.

3.2 ILOVEOPTIMIZINGWITHAMO

Now, let's dig into how we actually solve OP. Earlier we claimed that it only takes $\sqrt{T}$ calls to the AMO to solve the OP, but that's less than one per step...how does that work? Two enhancements:

1. Epochs: q_t is not updated at every step, but only on steps that are perfect squares (2, 4, 9, 16, etc)

2. Warm Start: the initial conditions for the optimization are set using the solution to the previous epoch

We solve the OP using coordinate descent:

Algorithm 3 Coordinate descent for OP

Input: initial weights for q
while True **do**
  if Claim 1 (small empirical regret) violated **then**
    $q \leftarrow cq$ $\{0 < c < 1$ is constantant given in paper$\}$
  else if $\exists \pi$ such that claim 2 (low variance) is violated **then**
    $q(\pi) = q(\pi) + \alpha$ $\{\alpha > 0$ is another constant given in paper$\}$
  else
    Return q
  end if
end while

If this algorithm terminates, we're all done! But how do we know it will?

Theorem 3. *The number of loops required to approximately solve OP using our coordinate descent procedure is* $\leq \frac{\log(\frac{1}{k\mu_t})}{\mu_t}$

Proof of this theorem was not given in class (we ran out of time). One result of this theorem is that, since the support of q can increase by at most 1 each loop, there will only be at most $\sqrt{T}$ nonzero elements by termination.

References

[AHK+14] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1638–1646, 2014.