Online Learning Applications

Part 10: Contextual bandits

Motivating example

Consider a **Movies recommendation** software.

What film should a streaming platform recommend **based on the previous views and ratings?**

Motivating example

Consider a **Movies recommendation** software.

What film should a streaming platform recommend **based on the previous views and ratings?**

A good system should take into account such information. More in general:

How can we design an algorithm which action (e.g., recommendation) depends on the context (e.g., information about the user)?

Contextual bandits

Contextual bandits set-up

For each round $t \in 1, \ldots, T$

- Learner observes **context** $x_t \in \mathcal{X}$ (**prior to decision!**)
- Learner picks arm a_t
- Learner gets reward $r_t(a_t) \in [0, 1]$

We will consider the following set-up (many variants are studied in the literature):

- Each $r_t(a)$ is drawn independently from a fixed **context-dependent** distribution $\mathbb{P}(\cdot|x_t, a)$. So $r_t(a)$ depends both on the context and on the action a
- We denote the expected reward of arm a under context x as $\mu(a|x)$
- The sequence of contexts $(x_1, \ldots, x_T)$ is chosen by an (oblivious) adversary

Baseline and regret

We consider the following baseline:

Baseline

The reward of the best policy that maps contexts to actions:

$$\pi^*(x) = \arg\max_{a \in A} \mu(a|x) \quad \forall x \in \mathcal{X}$$

The pseudo-regret with respect to the baseline is:

Pseudo-regret

$$\mathcal{R}_T = \sum_{t=1}^{T} (\mu(\pi^*(x_t)|x_t) - \mu(a_t|x_t))$$

Small number of contexts

Small number of contexts

Idea: run a separate copy of a known bandit algorithm (e.g., UCB1) for each context.

Algorithm: Contextual algorithm

Init: instantiate a (non-contextual) regret minimizer ALG_x for each context x;
for $t = 1, \ldots, T$ **do**
  invoke ALG_x with $x = x_t$, that is: play $a_t \leftarrow \text{ALG}_x$;
  observe $r_t(a_t)$ and return it to ALG_x, that is: ALG_x receive as feedback the reward $r_t(a_t)$;

Regret Guarantees

Assumption: ALG_x has pseudo-regret $\mathcal{R}_{\text{ALG}_x} \leq O(\sqrt{KT \log T})$.

Theorem

The preudo-regret of the contextual algorithm is

$$\mathcal{R}_T \leq O(\sqrt{KT|\mathcal{X}| \log T}).$$

Proof sketch.

Let $n_x = \sum_{t=1}^{T} \mathbb{I}[x_t = x]$ be the number of times context x appears. Then, the pseudo-regret accumulated under context x is at most

$$\mathcal{R}_{\mathsf{ALG}_x} = O(\sqrt{K n_x \log T}).$$

Proof sketch.

Let $n_x = \sum_{t=1}^{T} \mathbb{I}[x_t = x]$ be the number of times context x appears. Then, the pseudo-regret accumulated under context x is at most

$$\mathcal{R}_{\mathsf{ALG}_x} = O(\sqrt{K n_x \log T}).$$

⚠ The previous bound holds only with high probability (see, e.g., UCB1 proof)

Proof sketch.

Let $n_x = \sum_{t=1}^{T} \mathbb{I}[x_t = x]$ be the number of times context x appears. Then, the pseudo-regret accumulated under context x is at most

$$\mathcal{R}_{\mathsf{ALG}_x} = O(\sqrt{K n_x \log T}).$$

⚠ The previous bound holds only with high probability (see, e.g., UCB1 proof)

Hence, the overall regret it at most:

$$\sum_{x \in \mathcal{X}} \mathcal{R}_{\mathsf{ALG}_x} = \sum_{x \in \mathcal{X}} O(\sqrt{K n_x \log T}) \leq O(\sqrt{KT|\mathcal{X}| \log T}),$$

where the last inequality follows from the Cauchy–Schwarz inequality. □

Drawbacks

Regret bound is very high if $|\mathcal{X}|$ is large, e.g., if contexts are feature vectors with a large number of features. To handle contextual bandits with a large (or infinite) $|\mathcal{X}|$, we either assume some **structure**, or change the **objective**.

Adding structure: Lipshitz contextual bandits

Adding structure: Lipshitz contextual bandits

Let $\mathcal{X} \subseteq [0,1]$ and assume that, for each $x, x' \in \mathcal{X}$,

$$|\mu(a|x) - \mu(a|x')| \leq L\,|x - x'| \qquad \forall a \in \mathcal{A}.$$

Simple idea:

- Discretize the space of contexts
- Let S_ϵ be the ϵ-uniform grid on $[0,1]$
- Use S_ϵ in place of $\mathcal{X}$
- Map the observed context to the closest point in the grid

We have $1 + 1/\epsilon$ points in our grid. If this number is small enough we can use the strategy just discussed for small number of contexts.

Adding structure: Lipshitz contextual bandits

What is the optimal trade-off between:

- having a grid that is precise enough
- having a "small enough" number of points?

We have seen a similar problem regarding pricing.

Setting $\epsilon = T^{-1/3}$, we suffer:

- $L \cdot \epsilon \cdot T = L \cdot T^{2/3}$ regret from optimizing over $L \cdot \epsilon$ optimal solutions
- $O(\sqrt{KT|S_\epsilon| \log T} \simeq T^{2/3})$ regret of the contextual no-regret algorithm

Adding structure: Lipshitz contextual bandits

What is the optimal trade-off between:

- having a grid that is precise enough
- having a "small enough" number of points?

We have seen a similar problem regarding pricing.

Setting $\epsilon = T^{-1/3}$, we suffer:

- $L \cdot \epsilon \cdot T = L \cdot T^{2/3}$ regret from optimizing over $L \cdot \epsilon$ optimal solutions
- $O(\sqrt{KT|S_\epsilon| \log T} \simeq T^{2/3})$ regret of the contextual no-regret algorithm

This discretization techniques can be applied also to multi-dimensional contexts $\mathcal{X} \subseteq [0,1]^d$, but **the number of contexts increases exponentially in d providing much worse performances**.

Adding structure: Lipshitz contextual bandits

What is the optimal trade-off between:

- having a grid that is precise enough
- having a "small enough" number of points?

We have seen a similar problem regarding pricing.

Setting $\epsilon = T^{-1/3}$, we suffer:

- $L \cdot \epsilon \cdot T = L \cdot T^{2/3}$ regret from optimizing over $L \cdot \epsilon$ optimal solutions
- $O(\sqrt{KT|S_\epsilon| \log T} \simeq T^{2/3})$ regret of the contextual no-regret algorithm

This discretization techniques can be applied also to multi-dimensional contexts $\mathcal{X} \subseteq [0,1]^d$, but **the number of contexts increases exponentially in d providing much worse performances**.

Can we do better in multi-dimensional settings?

Adding structure: Linear contextual bandits

Adding structure: Linear contextual bandits

Contextual linear bandits

For each $t \in 1, \ldots, T$:

1. Observe $x_t = \{x_{t,a} \in [0,1]^d$ for each $a \in A\}$
2. play arm $a_t \in A$
3. get reward $r_t = \langle \theta^\star, x_{t,a_t} \rangle + \epsilon_t$

where

- $\theta^\star$ is an **unknown** regression vector
- ϵ_t is a subgaussian noise

Goal: **learn** unknown $\theta^\star$.... while **maximizing** rewards!

Lin-UCB (informal)

Linear-contextual bandit problem can be solved by a more complex version of UCB
idea: build "confidence region" for the $\theta^\star$ vector

Idea of Lin-UCB

- In each round we construct a confidence region C_t such that $\theta^\star \in C_t$ with high probability
- Use C_t to construct an UCB on the mean reward of each arm given contexts: $UCB_t(a|x_t) = \sup_{\theta \in C_t} \langle x_{t,a}, \theta \rangle$
- Pick arm a maximizing $UCB_t(a|x_t)$

Known results: regret UB of $\widetilde{O}(d\sqrt{T})$, LB of $\Omega(\sqrt{dT})$ (see, e.g., Abbasi-Yadkori et al. [2011])

Estimate $\theta^\star$

An estimate of $\theta^\star$ can be obtained through least square regression

Design matrix with regularization parameter λ:

$$D_t = \lambda I_d + \sum_{t'=1}^{t} x_{t',a_{t'}} x_{t',a_{t'}}^\top$$

Regularized least-square estimate:

$$\hat{\theta}_t = D_t^{-1} \sum_{t'=1}^{t} r_{t'} x_{t',a_{t'}}$$

The estimated reward of an arm a at round t is:

$$x_{t,a} = \langle \hat{\theta}_t, x_{t,a} \rangle$$

Estimate $\theta^\star$

An estimate of $\theta^\star$ can be obtained through least square regression

Design matrix with regularization parameter λ:

$$D_t = \lambda I_d + \sum_{t'=1}^{t} x_{t',a_{t'}} x_{t',a_{t'}}^\top$$

Regularized least-square estimate:

$$\hat{\theta}_t = D_t^{-1} \sum_{t'=1}^{t} r_{t'} x_{t',a_{t'}}$$

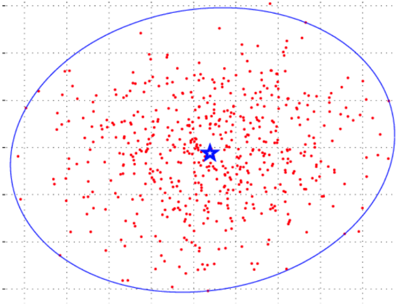
The estimated reward of an arm a at round t is:

$$x_{t,a} = \langle \hat{\theta}_t, x_{t,a} \rangle$$

⚠ It is not enough to have an estimate of the rewards. **We need a confidence interval!**

Confidence sets (informal)

The confidence set C_t will be an ellipsoid centered in the estimate $\hat{\theta}_t$.



$$C_t = \left\{ \theta : \|\theta - \hat{\theta}_t\|_{D_t} \right\} \leq \beta_t$$

Where

- $\|\theta\|_D = \sqrt{\theta^\top A \, \theta}$
- β_t is a parameter

Changing objective: Contextual bandits with a policy class

Changing the objective: contextual bandits with a policy class

No assumptions on rewards. We make the problem tractable by **restricting the benchmark** in the definition of regret

Policies and model classes

- A policy π is a mapping from contexts to actions
- A value function f is a mapping $f : \mathcal{X} \times \mathcal{A} \to [0, 1]$ modeling the mean of the rewards distribution (e.g., a linear model, a neural network etc.). We denote by $\mathcal{F}$ the class of value functions to which we have access
- A value function f induces a policy $\pi_f(x) = \arg\max_{a \in \mathcal{A}} f(x, a)$. The class of policies induced by $\mathcal{F}$ is Π

Goal: minimize $R_T = \sum_{t=1}^{T} r_t(\pi^*(x_t)) - \sum_{t=1}^{T} r_t(a_t)$.

Connection with supervised ML

Clear connection to “traditional” supervised machine learning: re-use its tools!!
We need the following two **assumptions**:

Access to a **regression oracle** SqAlg. At each round the regressor is trained using the feedback of the first $t-1$ rounds and outputs a regressor f_t.

Realizability assumption: there exists a regressor $f^* \in \mathcal{F}$ such that for all t, $f^*(x, a) = \mathbb{E}[r_t(a) | x_t = x]$.

SquareCB by Foster and Rakhlin [2020]

Algorithm 1 SquareCB

parameters:

Learning rate $\gamma > 0$, exploration parameter $\mu > 0$.

Online regression oracle SqAlg.

for $t = 1, \ldots, T$ **do**

Receive context x_t.

// Compute oracle's predictions $\widehat{y}_t(x, a) := \mathsf{SqAlg}_t(x, a\,; (z_1, y_1), \ldots, (z_{t-1}, y_{t-1}))$.

For each action $a \in \mathcal{A}$, compute $\widehat{y}_{t,a} := \widehat{y}_t(x_t, a)$.

Let $b_t = \arg\min_{a \in \mathcal{A}} \widehat{y}_{t,a}$.

For each $a \neq b_t$, define $p_{t,a} = \frac{1}{\mu + \gamma(\widehat{y}_{t,a} - \widehat{y}_{t,b_t})}$, and let $p_{t,b_t} = 1 - \sum_{a \neq b_t} p_{t,a}$.

Sample $a_t \sim p_t$ and observe loss $\ell_t(a_t)$.

Update SqAlg with example $((x_t, a_t), \ell_t(a_t))$.

(pseudo-code is for minimization. It works analogously replacing losses with rewards)

Regret guarantees

The regret of the algorithm will depends on the regret of the sequence $\{f_t\}$. We assume that:

$$\sum_{t \in [T]} (\hat{y}_{t,a_t} - f^\star(x_t, a_t))^2 \leq \mathrm{Reg}_{Sq}$$

Then, SquareCB guarantees regret:

$$R_T = O\left(\sqrt{K \;\; T \; \mathrm{Reg}_{Sq}}\right)$$

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011.

Dylan Foster and Alexander Rakhlin. Beyond ucb: Optimal and efficient contextual bandits with regression oracles. In *International Conference on Machine Learning*, pages 3199–3210. PMLR, 2020.