The Contextual Bandits Problem

Techniques for Learning to Make High-Reward Decisions

Rob Schapire

Example: Ad/Content Placement

- repeat:
  1. website visited by user (with profile, browsing history, etc.)
  2. website chooses ad/content to present to user
  3. user responds (clicks, leaves page, etc.)
- goal: make choices that elicit desired user behavior

Example: Medical Treatment

- repeat:
  1. doctor visited by patient (with symptoms, test results, etc.)
  2. doctor chooses treatment
  3. patient responds (recovers, gets worse, etc.)
- goal: make choices that maximize favorable outcomes

The Contextual Bandits Problem

- repeat:
    1. learner presented with context
    2. learner chooses an action
    3. learner observes reward (but only for chosen action)
- goal: learn to choose actions to maximize rewards
- general and fundamental problem: how to learn to make intelligent decisions through experience

Issues

- classic dilemma:
  - exploit what has already been learned
  - explore to learn which behaviors give best results
- in addition, must use context effectively
  - many choices of behavior possible
  - may never see same context twice — need to generalize
- selection bias: if explore while exploiting, will tend to get highly skewed data
- efficiency

This Talk

- overview of some of the algorithms and techniques used for contextual bandits (and variants)
- want algorithms that:
  - are general-purpose and practical — fast and simple to implement
  - can learn complex behaviors based on context
  - have provably strong statistical guarantees

Outline

- formalizing the learning problem
- algorithms

- an application and a next step

Outline

- formalizing the learning problem
- algorithms
- an application and a next step

Formal Model

- repeat, for $t = 1, \ldots, T$:
  1a. learner observes context x_t
  1b. reward vector $\mathbf{r}_t \in [0,1]^K$ chosen (but not observed)
  2. learner selects action $a_t \in \{1, \ldots, K\}$
  3. learner receives observed reward $r_t(a_t)$
- goal: maximize total reward:

$$\sum_{t=1}^{T} r_t(a_t)$$

- for now: assume pairs $(x_t, \mathbf{r}_t)$ chosen at random i.i.d.

Example

Context	Actions 1	2	3
$(\textit{Male}, 50, \ldots)$	1.0	$\boxed{0.2}$	0.0
$(\textit{Female}, 18, \ldots)$	1.0	0.0	$\boxed{1.0}$
$(\textit{Female}, 48, \ldots)$	0.5	$\boxed{0.1}$	0.7
$\vdots$		$\vdots$	

$$\text{total reward} = 0.2 + 1.0 + 0.1 + \cdots$$

Policies

- aim: learn to choose actions based on context
- want good policy: rule for selecting action from context
- e.g.:

If (sex = male)	choose action 2
Else if (age $>$ 45)	choose action 1
else	choose action 3

- policy π : (context x) $\mapsto$ (action a)
- before learning, must choose general form of policies to be used

  $\Rightarrow$ defines policy space Π

  - e.g.: all decision trees (nested "if-then-else" rules)
  - tacit assumption:
    $\exists$ (unknown) policy $\pi \in \Pi$ that gives high rewards

Learning with Context and Policies

- goal: learn through experimentation to do (almost) as well as best $\pi \in \Pi$
- assume Π finite, but typically extremely large
- policies may be very complex and expressive
  $\Rightarrow$ powerful approach
- challenges:
  - Π extremely large
  - need to be learning about all policies simultaneously while also performing as well as the best
  - when action selected, only observe reward for policies that would have chosen same action
  - exploration versus exploitation on a gigantic scale!

Formal Model (*revisited*)

- repeat, for $t = 1, \ldots, T$:
  1a. learner observes context x_t
  1b. reward vector $\mathbf{r}_t \in [0,1]^K$ chosen (but not observed)
  2. learner selects action $a_t \in \{1, \ldots, K\}$
  3. learner receives observed reward $r_t(a_t)$
- goal: want high total (or average) reward *relative to best policy* $\pi \in \Pi$
  - i.e., want small regret:

$$\underbrace{\max_{\pi \in \Pi} \frac{1}{T} \sum_{t=1}^{T} r_t(\pi(x_t))}_{\text{best policy's average reward}} - \underbrace{\frac{1}{T} \sum_{t=1}^{T} r_t(a_t)}_{\text{learner's average reward}}$$

  - "no regret" if regret $\to 0$ as $T \to \infty$

Outline

- formalizing the learning problem
- algorithms
  - full-information setting
  - bandit setting
- an application and a next step

Outline

- formalizing the learning problem
- algorithms
  - full-information setting
  - bandit setting
- an application and a next step

Starting Point: Full-Information Setting

- full-information setting: same as bandit, but learner can see rewards for all actions

Context	Actions 1	2	3
$(\mathit{Male}, 50, \ldots)$	1.0	$\boxed{0.2}$	0.0
$(\mathit{Female}, 18, \ldots)$	1.0	0.0	$\boxed{1.0}$
$(\mathit{Female}, 48, \ldots)$	0.5	$\boxed{0.1}$	0.7
$\vdots$		$\vdots$	

$\square$ = learner's action

$\bigcirc$ = π's action

$$\text{learner's total reward} = 0.2 + 1.0 + 0.1 + \cdots$$
$$\pi\text{'s total reward} = 0.0 + 1.0 + 0.5 + \cdots$$

- for any π, can compute rewards would have received
  - average is good estimate of π's expected reward

Follow-the-Leader Algorithm

- at round t:
  - find empirically best $\pi \in \Pi$ $\leftarrow$ main challenge
  - use to choose action: $a_t = \pi(x_t)$
- optimal regret: $O\left(\sqrt{\frac{\ln|\Pi|}{T}}\right)$
- to apply, need "oracle" (algorithm/subroutine) for finding best $\pi \in \Pi$ on observed contexts and rewards
  - "arg-max oracle" (aka: ERM oracle, classification oracle, linear oracle, ...)
- same as standard classification learning
- so: if have "good" classification algorithm for Π, can use to find good policy

technique: estimate expected reward of each policy

technique: use existing method ("oracle") to find best policy

Proof Ideas

- show every policy's empirical average reward close to expected reward
- implies empirically best policy has reward close to truly best policy $\Rightarrow$ regret bound

Non-Stochastic (Adversarial) Setting

- so far, assumed stochastic setting: each $(x_t, \mathbf{r}_t)$ i.i.d.
- not always realistic, e.g.:
  - temporally correlated or drifting data
  - truly adversarial environment (as in game playing)
- non-stochastic (adversarial) setting:
  - contexts x_t and rewards $\mathbf{r}_t$ are arbitrary
    - not assumed random
    - possibly selected by adversary
- follow-the-leader does not work here
  - adversary can force very low reward while ensuring one policy gets fairly high reward

Hedge Algorithm

[Littlestone & Warmuth][Freund & Schapire]

- maintain one weight for every $\pi \in \Pi$
- on each round t:
  - choose random policy π with probability proportional to weights
  - use action chosen by π
  - increase weight of each policy according to reward it would have received
- yields optimal regret, even in adversarial setting
- but: time/space are linear in $|\Pi|$
  - too slow if $|\Pi|$ gigantic
- applications:
  - game-playing: can use to play/solve games
  - boosting: AdaBoost derived from Hedge

technique: use weighted combination of policies

Proof Ideas

- keep track of sum of weights of all policies
  - upper bound in terms of reward of algorithm
  - lower bound in terms of reward of best policy
- combine to get regret bound

Follow-the-Leader versus Hedge

- follow-the-leader:
  - stochastic setting only
  - optimal regret
  - efficient, given access to oracle
- Hedge:
  - non-stochastic setting
  - optimal regret
  - inefficient if $|\Pi|$ huge
- is best of both possible?
  - i.e., no-regret, oracle-efficient algorithm for non-stochastic setting?
  - appears impossible [Hazan & Koren]

Outline

- formalizing the learning problem
- algorithms
  - full-information setting
  - bandit setting
- an application and a next step

Back to Bandit Setting

- only see rewards for actions taken

	Actions		
Context	1	2	3
$(\mathit{Male}, 50, \ldots)$	1.0	$\boxed{0.2}$	0.0
$(\mathit{Female}, 18, \ldots)$	1.0	0.0	$\boxed{1.0}$
$(\mathit{Female}, 48, \ldots)$	0.5	$\boxed{0.1}$	0.7
$\vdots$		$\vdots$	

$\square$ = learner's action

$\bigcirc$ = π's action

learner's total reward $= 0.2 + 1.0 + 0.1 + \cdots$

π's total reward $= ?? + 1.0 + ?? + \cdots$

- for any policy π, only observe π's rewards on subset of rounds
- might like to use oracle to find empirically good policy
- problems:
  - only see some rewards
  - observed rewards highly biased
    (due to skewed choice of actions)

Exploration is Necessary

- e.g.:
  - drug A is "pretty good" (cure rate $= 60\%$)
  - drug B is "much better" (cure rate $= 80\%$)
- in early trials, by chance, A might appear better than B

$\Rightarrow$ follow-the-leader can "get stuck" only picking A

- need exploration!
- problem even more extreme with more complex policies

ϵ-Greedy/Epoch-Greedy Algorithm

[Langford & Zhang]

- modified follow-the-leader for bandit stochastic setting
  - explicit exploitation and exploration
- on each round, choose action:
  - according to "best" policy so far (with probability $1 - \epsilon$)
    [can find with oracle]
  - uniformly at random (with probability ϵ)
- simple and fast (given oracle)
- not optimal regret: $O\left(\left(\frac{K \ln |\Pi|}{T}\right)^{1/3}\right)$
- analysis: similar to follow-the-leader

  technique: explicit exploration via uniform sampling of actions

De-biasing Biased Estimates

- selection bias is major problem
- simple (and old) trick: inverse-propensity weighting
  - say want to estimate $\mathrm{E}[X]$
    (e.g.: probability unfair coin comes up heads)
  - with probability p: observe X once
    with probability $1-p$: don't observe X at all!
  - trick: define

$$\hat{X} = \begin{cases} X/p & \text{if observed} \\ 0 & \text{else} \end{cases}$$

  - then $\mathrm{E}[\hat{X}] = \mathrm{E}[X]$ — unbiased!
- can use to get unbiased estimates for rewards of all actions
  (not just observed)

Variance Control

- estimates are unbiased — done?
- no! — variance may be extremely large

∴ to get good estimators, must control variance

  - sometimes can do with uniform sampling of actions
  - sometimes need more sophisticated approach

technique: inverse-propensity weighting to get unbiased estimates

Bandits in Non-Stochastic Setting

[Auer, Cesa-Bianchi, Freund & Schapire]

- Exp4: contextual-bandits algorithm for non-stochastic setting
- combines:
  - Hedge
  - uniform sampling of actions
  - inverse-propensity weighting
- optimal regret: $O\left(\sqrt{\frac{K \ln |\Pi|}{T}}\right)$
- analysis: similar to Hedge, but also must account for variance
- but like Hedge: time/space are linear in $|\Pi|$

Epoch-Greedy versus Exp4

- epoch-greedy:
  - stochastic setting
  - not optimal regret: $O\left(T^{-1/3}\right)$
  - efficient, given access to oracle
- Exp4:
  - non-stochastic setting
  - optimal regret: $O\left(T^{-1/2}\right)$
  - inefficient if $|\Pi|$ huge
- difference in regret is big!
  - to get regret ε, need $O\left(1/\varepsilon^3\right)$ versus $O\left(1/\varepsilon^2\right)$ trials
- best of both?
  - in stochastic setting, is there an algorithm that is fast (given oracle) and has near optimal regret?
    yes!

"Mini-Monster" Algorithm (aka: ILOVETOCONBANDITS)

[Agarwal, Hsu, Kale, Langford, Li & Schapire]

- apply all preceding techniques
- every round, find weighted combination of policies satisfying explicitly stated properties:
  1. low (estimated) regret [exploit]
     i.e., choose actions think will give high reward
  2. low (estimated) variance [explore]
     i.e., ensure future estimates will be accurate
- can formulate as very large and complex optimization problem
- solve using simple and efficient algorithm, using oracle
  - find violated contraint and fix it — repeat until done

Mini-Monster (cont.)

- regret (nearly) optimal:

$$\tilde{O}\left(\sqrt{\frac{K \ln |\Pi|}{T}}\right)$$

- fast! only requires an average of

$$\tilde{O}\left(\sqrt{\frac{K}{T \ln |\Pi|}}\right) \ll 1$$

  oracle calls per round

- same approach as RandomizedUCB (aka "Monster")
  but simpler and much faster
  [Dudík, Hsu, Kale, Karampatziakis, Langford, Reyzin & Zhang]

technique: formulate properties as optimization problem and solve

Proof Ideas

- regret bound:
  - regret constraint ensures low regret
    (if estimates are good enough)
  - variance constraint ensures that they actually will be good enough
- efficiency of numerical algorithm:
  - use potential function to measure progress

Outline

- formalizing the learning problem
- algorithms
  - full-information setting
  - bandit setting
- an application and a next step

Application: Multiworld Testing Decision Service

[Agarwal, Bird, Cozowicz, Hoang, Langford, Lee, Li, Melamed, Oshri, Ribas, Sen, Slivkins]

- unified system for solving contextual-bandit problems
  - general-purpose
  - modular
  - easy to interface with existing systems
  - designed to reduce common errors
- e.g.: deployed to select news articles on MSN homepage
  - no previous learning method had been successful
  - 25% relative lift in click-through rate
  - used now in production (thousands of requests per second)

A Next Step: Contextual Bandits with Underlying State

[Jiang, Krishnamurthy, Agarwal, Langford & Schapire]

- decisions made now can significantly impact the future
  - may be underlying state affected by actions
- e.g., medical treatment:
  - see same patient repeatedly
  - state: underlying condition or disease, stage of progression, etc.
    - affected by chosen treatment
- still want to find best policy
  - much harder since choices have impact well into future
  - every policy can define very different sequence of actions
- new exploration algorithm for finding "best" policy
  - assumes feasibility of "value-function approximation"
  - polynomial in new measure of tractability called Bellman rank
  - but: not computationally efficient — more to do!

Conclusions

- contextual bandits is a challenging problem, especially if want
  - computational efficiency
  - very large policy space (for highly complex behaviors)
  - optimal statistical performance (regret)
  - adversarial setting
- building up effective methods for meeting these challenges
- theory is indispensible guide, paying off in practice

Works cited:

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

Elad Hazan and Tomer Koren. The computational power of optimization in online learning. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, pages 128–141, 2016.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems 20*, pages 817–824, 2008.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

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 *Proceedings of The 31st International Conference on Machine Learning*, 2014.

Miroslav Dudík, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, Tong Zhang. Efficient optimal learning for contextual bandits. In *Uncertainty in Artificial Intelligence: Proceedings of the Twenty-Seventh Conference*, pages 169–178, 2011.

Alekh Agarwal, Sarah Bird, Markus Cozowicz, Luong Hoang, John Langford, Stephen Lee, Jiaji Li Dan Melamed, Gal Oshri, Oswaldo Ribas, Siddhartha Sen, Alex Slivkins. Making contextual decisions with low technical debt. arXiv:1606.03966.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford and Robert E. Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In *Proceedings of the 34th International Conference on Machine Learning*, PMLR 70:1704–1713, 2017.

Other textbooks and surveys:

Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Li Zhou. A survey on contextual multi-armed bandits. arXiv:1508.03326.