Resourceful Contextual Bandits

Ashwin Badanidiyuru (Cornell → Google Research)
John Langford (MSR-NYC)
Alex Slivkins (MSR-NYC)

COLT 2014

Basics: bandits with IID rewards

- In each round $t = 1, \ldots, T$
  - algorithm picks action $a \in A$
  - reward is drawn indep. from ***unknown*** distribution D_a
- Goal: maximize *total expected reward* (REW)

  Regret = OPT – REW[algorithm]

  benchmark
- OPT = REW [best all-knowing algorithm]

  = REW [best action]

  in this setting

  knows the D_a 's

Example: dynamic pricing

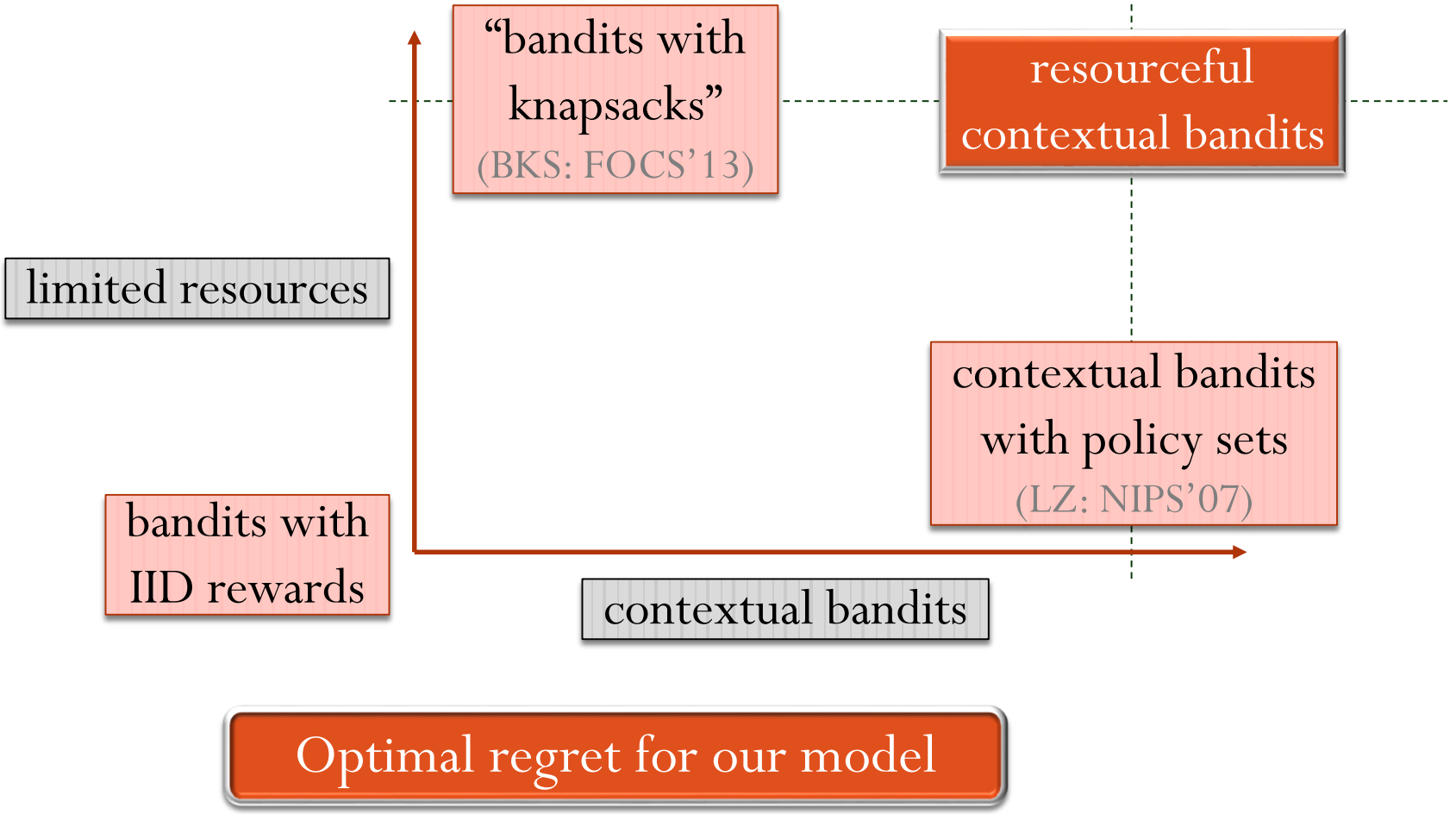
- Algorithm = seller with unlimited supply of items
- In each round $t = 1, \ldots, T$
  - a new customer arrives
  - algorithm picks *action*: price p_t
  - customer either buys $@p_t$ or leaves
  - sale happens indep. from ***unknown*** probability $S(p_t)$
- Goal: maximize *total expected revenue*

Example: dynamic pricing

- Algorithm = seller with unlimited supply of items
- In each round $t = 1, \ldots, T$ — Stop if out of items
  - a new customer arrives, with known profile x_t
  - algorithm picks *action*: price p_t
  - customer either buys $@p_t$ or leaves
  - sale happens indep. from ***unknown*** probability $S(p_t, x_t)$
- Goal: maximize *total expected revenue*

Extensions:
- limited supply — limited resources
- customer profile — per-round contexts (contextual bandits)

High-level picture



Resourceful contextual bandits

- d limited resources, known budget on each.
  Stop if some resource is exhausted.
- In each round $t = 1, \ldots, \mathrm{T}$
  - context $x_t \in X$ arrives, drawn IID from known D_X
  - algorithm picks action $a_t \in A$
  - outcome (reward r_t, resource consumption $(c_{t,1}, \ldots, c_{t,d})$) is drawn indep. from ***unknown*** distribution $D_{(x_t, a_t)}$
- Goal: maximize *total expected reward* (REW)

Normalization: all per-round rewards and consumptions are in [0,1]

Resourceful contextual bandits

- Regret = OPT – REW [algorithm]
- OPT = REW [best ***all-knowing*** algorithm]

knows the distributions $D_{(x,a)}$'s

Resourceful contextual bandits

- Regret = OPT_{Π} – REW [algorithm]
- OPT_{Π} = REW [best ***all-knowing*** algorithm restricted to Π]

knows the distributions $D_{(x,a)}$'s

- **Policy** = mapping from contexts to actions
- OPT is relative to a given (sub)set Π of policies

Because of resource constraints, one may have

$\text{OPT}_{\Pi} \gg$ REW[best fixed policy in Π]

Essentially, OPT_{Π} = REW[best fixed mixture of policies in Π]

Generality of the model

Example	Resource	In each round	Reward	Context
Seller	items to sell	one item for sale @price	money	Customer profile
Employer	budget of money	one task to perform @price	tasks	Worker/task profile
allocation of ads	advertiser's budgets	one ad to display (pay-per-click)	clicks	page / user profile

Policy set Π = all policies that can be learned from offline data via given method, e.g. linear regression, decision trees, etc.

Related work

- Special cases
  - *No contexts* $\Longrightarrow$ "bandits with knapsacks"
    Defined & solved in (Badanidiyuru, Kleinberg, S.: FOCS'13).
    Many prior papers on various sub-cases
  - *No resources* $\Longrightarrow$ contextual bandits with policy sets
    (e.g.: Langford & Zhang: NIPS'07; Dudik et al: UAI'11)
- Concurrent & independent work (Devanur & Agrawal: EC'14)
  Another model for *contextual bandits with limited resources*:
  more general than ours on limited resources,
  less general than ours on contexts (linear dependence)

Our main result

Algorithm with regret

$$O(1 + \mathrm{OPT}_{\Pi}/B)\sqrt{dKT\ \log(dKT|\Pi|)}$$

T rounds, K actions, d constraints, policy set Π, smallest budget B

This regret is optimal in several ways:

- all constraints (incl. time) scaled by $\alpha \Rightarrow$ regret scaled by $\sqrt{\alpha}$
- $\mathrm{OPT}_{\Pi} \leq O(B) \Rightarrow$ regret $\tilde{O}(\sqrt{KT})$
- $\sqrt{\log |\Pi|}$ dependence is optimal (even without resources)

Caveat: algorithm is (very) computationally inefficient

New challenges

Due to *limited resources*:

- maximizing expected *per-round* reward is not the right goal.
  must think about expected total reward *over all rounds*
- best all-knowing algorithm is not a fixed policy in Π,
  but (essentially) a ***mixture*** of policies in Π

policy: contexts → actions

Due to *contexts*:

- trivial reduction to prior work on special case of *no contexts*:
  policies in Π are "meta-arms" ⇒ regret scales as $\sqrt{|\Pi|}$
- Whereas our regret scales as $\log |\Pi|$.
  For that, each round must explore many policies at once!

Algorithm (outline)

- Explore-exploit tradeoff:
  adapt exploration to observations, explore & exploit at once
- Goal: zoom in on optimal ***mixture*** of policies in Π

  policy: contexts → actions
- In each round t,
  - pick a *plausibly optimal* mixture M (given observations so far)
  - with low probability pick action a_t u.a.r.,
    else draw policy $\pi \sim M$, pick action $a_t = \pi(\text{context } x_t)$
  - pick M so as to ***explore every policy*** $\pi' \in \Pi$ at once:

    $a_t = \pi'(x_t)$ with "near-optimal" probability

Open questions

- Optimal regret ***and*** good running time
  - Resolved for the special cases of *no contexts* (prior work) and no resources (concurrent & independent: ICML'14)
- Many open questions (even) for the special case of *no contexts*
  - beyond IID environment: change over time (no prior work)
  - regret bounds are not tight for important special cases
  - if actions are prices, how to discretize them?
    very tricky even for a single limited resource,
    open for multiple limited resources (e.g. products for sale)