The Contextual Bandits Problem

A New, Fast, and Simple Algorithm

Alekh Agarwal (MSR)
Daniel Hsu (Columbia)
Satyen Kale (Yahoo)
John Langford (MSR)
Lihong Li (MSR)
Rob Schapire (MSR/Princeton)
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
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
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
• 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
• selection bias: if explore while exploiting, will tend to get highly skewed data
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
- selection bias: if explore while exploiting, will tend to get highly skewed data
- efficiency
This Talk

• new and general algorithm for contextual bandits
• optimal statistical performance
• far faster and simpler than predecessors
Formal Model

- repeat

  1a. learner observes context $x_t$

  2. learner selects action $a_t \in \{1, \ldots, K\}$
  3. learner receives observed reward $r_t(a_t)$

• goal: maximize total reward:

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

• assume pairs $(x_t, r_t)$ chosen at random i.i.d.
Formal Model

• repeat
  1a. learner observes context $x_t$
  1b. reward vector $r_t \in [0, 1]^K$ chosen (but not observed)
  2. learner selects action $a_t \in \{1, \ldots, K\}$
  3. learner receives observed reward $r_t(a_t)$
Formal Model

• repeat
  1a. learner observes context $x_t$
  1b. reward vector $r_t \in [0, 1]^K$ 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:

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

- repeat
  1a. learner observes context $x_t$
  1b. reward vector $r_t \in [0, 1]^K$ 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:
  \[
  \sum_{t=1}^{T} r_t(a_t)
  \]

- assume pairs $(x_t, r_t)$ chosen at random i.i.d.
Example

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(Male, 50, ...)</em></td>
<td>1 2 3</td>
</tr>
</tbody>
</table>

\[ \text{total reward} = 0.2 + 1.0 + 0.1 + \cdots \]
<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Male, 50, ...)</td>
<td>1.0 0.2 0.0</td>
</tr>
<tr>
<td>Context</td>
<td>Actions</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
</tr>
<tr>
<td>(Male, 50, ...)</td>
<td>1.0  0.2  0.0</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(Male, 50, ...)</em></td>
<td>1.0</td>
<td>0.2</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**total reward** = 0.2 +
Example

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Male, 50, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>0.2</td>
</tr>
</tbody>
</table>

total reward = 0.2 +
### Example

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>((Male, 50,\ldots))</td>
<td>1.0 0.2 0.0</td>
</tr>
<tr>
<td>((Female, 18,\ldots))</td>
<td>1.0 0.0 1.0</td>
</tr>
</tbody>
</table>

Total reward = 0.2 +
### Example

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Male, 50, ... )</td>
<td>1.0</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**Total reward** = 0.2 + 1.0 +
### Example

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(Male, 50, \ldots)$</td>
<td>$1.0$</td>
</tr>
<tr>
<td>$(Female, 18, \ldots)$</td>
<td>$1.0$</td>
</tr>
<tr>
<td>$(Female, 48, \ldots)$</td>
<td>$1.0$</td>
</tr>
</tbody>
</table>

Total reward $= 0.2 + 1.0 + $
### Example

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>((Male, 50, \ldots))</td>
<td>1.0</td>
</tr>
<tr>
<td>((Female, 18, \ldots))</td>
<td>1.0</td>
</tr>
<tr>
<td>((Female, 48, \ldots))</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**total reward** = \(0.2 + 1.0 +\)
### Example

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\text{Male}, 50, \ldots)$</td>
<td>1.0 0.2 0.0</td>
</tr>
<tr>
<td>$(\text{Female}, 18, \ldots)$</td>
<td>1.0 0.0 1.0</td>
</tr>
<tr>
<td>$(\text{Female}, 48, \ldots)$</td>
<td>0.5 0.1 0.7</td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\vdots$</td>
</tr>
</tbody>
</table>

**total reward** $= 0.2 + 1.0 + 0.1 + \cdots$
Special Case: Multi-armed Bandit Problem

- no context
- try to do as well as best single action
Special Case: Multi-armed Bandit Problem

- no context
- try to do as well as best single action
  - tacitly assuming there is one action that gives high rewards
  - e.g.: single treatment/ad/content that is right for entire population
Policies

• in contextual bandits setting, can use context to choose actions
• may exist good policy (decision rule) for choosing actions based on context
• in **contextual bandits** setting, can use **context** to choose actions

• may exist good **policy** (decision rule) for choosing actions based on context

• e.g.:

  If \((\text{sex} = \text{male})\) choose action 2  
  Else if \((\text{age} > 45)\) choose action 1  
  else choose action 3
Policies

- In contextual bandits setting, can use context to choose actions.
- May exist good policy (decision rule) for choosing actions based on context.
- E.g.:
  
  If \((\text{sex} = \text{male})\) choose action 2
  
  Else if \((\text{age} > 45)\) choose action 1
  
  Else choose action 3

- Policy \(\pi : (\text{context } x) \mapsto (\text{action } a)\)
in contextual bandits setting, can use context to choose actions

may exist good policy (decision rule) for choosing actions based on context

e.g.:

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

policy \( \pi : (\text{context } x) \mapsto (\text{action } a) \)

learner generally working with some rich policy space \( \Pi \)

- e.g.: all decision trees ("if-then-else" rules)
Policies

- in **contextual bandits** setting, can use **context** to choose actions
- may exist good **policy** (decision rule) for choosing actions based on context
- e.g.:

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

- **policy** $\pi : (\text{context } x) \mapsto (\text{action } a)$
- learner generally working with some rich **policy space** $\Pi$
  - e.g.: all decision trees ("if-then-else" rules)
  - assume $\Pi$ finite, but typically extremely large
  - 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 $\pi \in \Pi$

• policies may be very **complex** and **expressive**
  $\Rightarrow$ powerful approach
Learning with Context and Policies

- **goal:** learn through experimentation to do (almost) as well as best $\pi \in \Pi$
- policies may be very complex and expressive
  $\Rightarrow$ powerful approach
- **challenges:**
  - $\Pi$ 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
  1a. learner observes context $x_t$
  1b. reward vector $r_t \in [0, 1]^K$ 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 $\pi \in \Pi$
Formal Model (revisited)

- repeat
  1a. learner observes context $x_t$
  1b. reward vector $r_t \in [0, 1]^K$ 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:

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

learner’s average reward
Formal Model (revisited)

- repeat
  1a. learner observes context $x_t$
  1b. reward vector $r_t \in [0, 1]^K$ 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:

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

  best policy’s average reward \hspace{1cm} learner’s average reward
An Algorithm that Solves this Problem

[Auer, Cesa-Bianchi, Freund, Schapire]

- **Exp4** solves this problem
  - maintains weights over all policies in $\Pi$
Exp4 solves this problem
- maintains weights over all policies in \( \Pi \)
- regret is essentially optimal:

\[
O \left( \sqrt{\frac{K \ln |\Pi|}{T}} \right)
\]

- even works for adversarial (i.e., non-random, non-iid) data
An Algorithm that Solves this Problem

[Auer, Cesa-Bianchi, Freund, Schapire]

- Exp4 solves this problem
  - maintains weights over all policies in $\Pi$
- regret is essentially optimal:
  $$O \left( \sqrt{\frac{K \ln |\Pi|}{T}} \right)$$
- even works for adversarial (i.e., non-random, non-iid) data
- but: time/space are linear in $|\Pi|$  
  - too slow if $|\Pi|$ gigantic
An Algorithm that Solves this Problem

[Auer, Cesa-Bianchi, Freund, Schapire]

- Exp4 solves this problem
  - maintains weights over all policies in $\Pi$
- regret is essentially optimal:

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

- even works for adversarial (i.e., non-random, non-iid) data
- but: time/space are linear in $|\Pi|$  
  - too slow if $|\Pi|$ gigantic
- seems hopeless to do better for fully general policy spaces
- this talk: aim for time/space only $\text{poly}(\log |\Pi|)$ when $\Pi$ is “well structured”
The (Fantasy) Full-Information Setting

- say see rewards for all actions
### The (Fantasy) Full-Information Setting

- say see rewards for **all** actions

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(Male, 50, ...)</em></td>
<td>1  2  3</td>
</tr>
</tbody>
</table>

• for any $\pi$, can compute rewards would have received

- average is good estimate of $\pi$’s expected reward
- choose empirically best $\pi \in \Pi$
- regret = $O(\sqrt{\ln |\Pi| T})$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>□</td>
</tr>
<tr>
<td></td>
<td>learner’s action</td>
</tr>
</tbody>
</table>
The (Fantasy) Full-Information Setting

- say see rewards for all actions

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(Male, 50,...)</em></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
</tr>
</tbody>
</table>

= learner’s action
### The (Fantasy) Full-Information Setting

- say see rewards for all actions

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Male, 50, …)</td>
<td>1</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0</td>
</tr>
</tbody>
</table>

learner’s total reward = 0.2 +

[ ] = learner’s action
The (Fantasy) Full-Information Setting

- say see rewards for all actions

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>((\textit{Male}, 50, \ldots))</td>
<td>1.0</td>
</tr>
<tr>
<td>((\textit{Female}, 18, \ldots))</td>
<td>1.0</td>
</tr>
<tr>
<td>((\textit{Female}, 48, \ldots))</td>
<td>0.5</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

learner’s total reward = 0.2 + 1.0 + 0.1 + \cdots
The (Fantasy) Full-Information Setting

- say see rewards for all actions

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>(Male, 50, ... )</td>
<td>1.0</td>
<td>0.2</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>(Female, 48, ...)</td>
<td>0.5</td>
<td>0.1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

learner’s total reward = 0.2 + 1.0 + 0.1 + ⋯

- for any $\pi$, can compute rewards would have received

$\pi$'s action

\[ \text{learner’s action} \]

\[ \overset{\square}{} \]
The (Fantasy) Full-Information Setting

- say see rewards for all actions

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Male, 50, ...)</td>
<td>1.0</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 48, ...)</td>
<td>0.5</td>
<td>0.1</td>
<td>0.7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

learner’s total reward = 0.2 + 1.0 + 0.1 + ...  
π’s total reward = 0.0 + 1.0 + 0.5 + ...  

- for any π, can compute rewards would have received
The (Fantasy) Full-Information Setting

- say see rewards for all actions

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(Male, 50, ...)</em></td>
<td>1.0 0.2 0.0</td>
</tr>
<tr>
<td><em>(Female, 18, ...)</em></td>
<td>1.0 0.0 1.0</td>
</tr>
<tr>
<td><em>(Female, 48, ...)</em></td>
<td>0.5 0.1 0.7</td>
</tr>
</tbody>
</table>

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

- for any \( \pi \), can compute rewards would have received
  - average is good estimate of \( \pi \)'s expected reward
The (Fantasy) Full-Information Setting

- say see rewards for all actions

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>(Male, 50, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 48, ...)</td>
<td>0.5</td>
</tr>
<tr>
<td>:                 :</td>
<td>:</td>
</tr>
</tbody>
</table>

learner’s total reward = 0.2 + 1.0 + 0.1 + \cdots

\pi’s total reward = 0.0 + 1.0 + 0.5 + \cdots

- for any \pi, can compute rewards would have received
  - average is good estimate of \pi’s expected reward
- choose empirically best \pi \in \Pi

\text{regret} = O\left(\sqrt{\frac{\ln |\Pi|}{T}}\right)
“Arg-Max Oracle” (AMO)

- to apply, just need “oracle” (algorithm/subroutine) for finding best $\pi \in \Pi$ on observed rewards
- input: $(x_1, r_1), \ldots, (x_T, r_T)$
  \begin{align*}
  x_t & = \text{context} \\
  r_t & = (r_t(1), \ldots, r_t(K)) = \text{rewards for all actions}
  \end{align*}
- output:
  \[
  \hat{\pi} = \arg \max_{\pi \in \Pi} \sum_{t=1}^{T} r_t(\pi(x_t))
  \]
“Arg-Max Oracle” (AMO)

- to apply, just need “oracle” (algorithm/subroutine) for finding best $\pi \in \mathcal{P}$ on observed rewards
- input: $(x_1, r_1), \ldots, (x_T, r_T)$
  - $x_t = \text{context}$
  - $r_t = (r_t(1), \ldots, r_t(K)) = \text{rewards for all actions}$
- output:

$$\hat{\pi} = \arg \max_{\pi \in \mathcal{P}} \sum_{t=1}^{T} r_t(\pi(x_t))$$

- really just (cost-sensitive) classification:
  - context $\leftrightarrow$ example
  - action $\leftrightarrow$ label/class
  - policy $\leftrightarrow$ classifier
  - reward $\leftrightarrow$ gain/(negative) cost
"Arg-Max Oracle" (AMO)

- to apply, just need “oracle” (algorithm/subroutine) for finding best $\pi \in \Pi$ on observed rewards
- input: $(x_1, r_1), \ldots, (x_T, r_T)$
  
  $x_t =$ context

  $r_t = (r_t(1), \ldots, r_t(K)) =$ rewards for all actions

- output:

  $\hat{\pi} = \arg \max_{\pi \in \Pi} \sum_{t=1}^{T} r_t(\pi(x_t))$

- really just (cost-sensitive) classification:

  context $\leftrightarrow$ example

  action $\leftrightarrow$ label/class

  policy $\leftrightarrow$ classifier

  reward $\leftrightarrow$ gain/(negative) cost

- so: if have “good” classification algorithm for $\Pi$, can use to find good policy (in full-information setting)
But in the Bandit Setting...

- ...only see rewards for actions taken
But in the Bandit Setting...

- ...only see rewards for actions taken

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Male, 50, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 48, ...)</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>(Male, 50, ...)</td>
<td>0.2</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>0.0</td>
</tr>
<tr>
<td>(Female, 48, ...)</td>
<td>0.1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>(Male, 50, ...)</td>
<td>0.0</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 48, ...)</td>
<td>0.7</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

□ = learner’s action
But in the Bandit Setting...

- ...only see rewards for actions taken

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>((Male, 50, \ldots))</td>
<td>1.0</td>
<td>0.2</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>((Female, 18, \ldots))</td>
<td>1.0</td>
<td>0.0</td>
<td>(1.0)</td>
<td></td>
</tr>
<tr>
<td>((Female, 48, \ldots))</td>
<td>0.5</td>
<td>(0.1)</td>
<td>0.7</td>
<td></td>
</tr>
</tbody>
</table>

\[\square = \text{learner's action}\]
But in the Bandit Setting...

- ...only see rewards for actions taken

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>((Male, 50, \ldots))</td>
<td>1.0  0.2  0.0</td>
</tr>
<tr>
<td>((Female, 18, \ldots))</td>
<td>1.0  0.0  1.0</td>
</tr>
<tr>
<td>((Female, 48, \ldots))</td>
<td>0.5  0.1  0.7</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

learner’s total reward = 0.2 + 1.0 + 0.1 + \ldots
But in the Bandit Setting...

- ...only see rewards for actions taken

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Male, 50, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 48, ...)</td>
<td><strong>0.5</strong></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

learner’s total reward = 0.2 + 1.0 + 0.1 + ...  

- for any policy $\pi$, only observe $\pi$’s rewards on subset of rounds
But in the Bandit Setting...

- ...only see rewards for actions taken

<table>
<thead>
<tr>
<th>Context</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Male, 50, ...)</td>
<td>1.0</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 48, ...)</td>
<td>0.5</td>
<td>0.1</td>
<td>0.7</td>
</tr>
</tbody>
</table>

learner's total reward = 0.2 + 1.0 + 0.1 + ···
π’s total reward = ?? + 1.0 + ?? + ···

- for any policy π, only observe π’s rewards on subset of rounds
But in the Bandit Setting...

- ...only see rewards for actions taken

<table>
<thead>
<tr>
<th>Context</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>(Male, 50, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 18, ...)</td>
<td>1.0</td>
</tr>
<tr>
<td>(Female, 48, ...)</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

learner’s total reward = 0.2 + 1.0 + 0.1 + ⋯
π’s total reward = ?? + 1.0 + ?? + ⋯

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

- still: AMO is a natural primitive
- key question: can we solve the contextual bandits problem given access to AMO?
Key Question

- still: AMO is a natural primitive
- key question: can we solve the contextual bandits problem given access to AMO?
- can we use an AMO on bandit data by somehow:
  - filling in missing data
  - overcoming bias
- want: optimal regret, time/space bounds poly(log |Π|)
- AMO is theoretical idealization, captures structure in policy space
- in practice, can use off-the-shelf classification algorithm
still: AMO is a natural primitive

key question: can we solve the contextual bandits problem given access to AMO?

can we use an AMO on bandit data by somehow:
  • filling in missing data
  • overcoming bias

want:
  • optimal regret
  • time/space bounds $\text{poly}(\log |\Pi|)$
• still: AMO is a natural primitive
• **key question**: can we solve the contextual bandits problem given access to AMO?
• can we use an AMO on bandit data by somehow:
  • filling in missing data
  • overcoming bias
• **want**:
  • optimal regret
  • time/space bounds $\text{poly}(\log |\Pi|)$
• AMO is **theoretical idealization**
• captures **structure** in policy space
• in **practice**, can use off-the-shelf classification algorithm
\( \epsilon \)-Greedy/Epoch-Greedy

- partially solved by the \( \epsilon \)-greedy/epoch-greedy algorithm
- on each round, choose action:
  - according to “best” policy so far (with probability \( 1 - \epsilon \))
  - uniformly at random (with probability \( \epsilon \))
\( \epsilon \)-Greedy/Epoch-Greedy

[Langford & Zhang]

- partially solved by the \( \epsilon \)-greedy/epoch-greedy algorithm
- on each round, choose action:
  - according to “best” policy so far (with probability \( 1 - \epsilon \))
    [can find with AMO]
  - uniformly at random (with probability \( \epsilon \))
\(\epsilon\)-Greedy/Epoch-Greedy

- partially solved by the \(\epsilon\)-greedy/epoch-greedy algorithm
- on each round, choose action:
  - according to "best" policy so far (with probability \(1 - \epsilon\))
  - uniformly at random (with probability \(\epsilon\))

\[\text{regret} = O\left(\left(\frac{K \ln |\Pi|}{T}\right)^{1/3}\right)\]

- fast and simple, but not optimal

[Langford & Zhang]
“Monster” Algorithm

[Dudík, Hsu, Kale, Karampatziakis, Langford, Reyzin & Zhang]

- RandomizedUCB (aka “Monster”) algorithm gets optimal regret using AMO
- solves multiple optimization problems using ellipsoid algorithm
- very slow: calls AMO about $\tilde{O}(T^4)$ times on every round
Main Result

• new, simple algorithm for contextual bandits with AMO access
• (nearly) optimal regret: $\tilde{O}\left(\sqrt{\frac{K \ln |\Pi|}{T}}\right)$
• fast: calls AMO far less than once per round!
  • on average, calls AMO

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

times per round
Main Result

• new, simple algorithm for contextual bandits with AMO access
• (nearly) optimal regret: \( \tilde{O}\left(\sqrt{\frac{K \ln |\Pi|}{T}}\right) \)
• fast: calls AMO far less than once per round!
  • on average, calls AMO
    \[ \tilde{O}\left(\sqrt{\frac{K}{T \ln |\Pi|}}\right) \ll 1 \]
    times per round
• rest of talk: sketching main ideas of the algorithm
De-biasing Biased Estimates

- **selection bias** is major problem:
  - only observe reward for single action
  - exploring while exploiting leads to **inherently biased** estimates
De-biasing Biased Estimates

- **selection bias** is major problem:
  - only observe reward for single action
  - exploring while exploiting leads to **inherently biased** estimates
- nevertheless: can use **simple trick** to get unbiased estimates for **all** actions
De-biasing Biased Estimates (cont.)

- say $r(a) = \text{(unknown) reward for action } a$
  $p(a) = \text{(known) probability of choosing } a$

$E[\hat{r}(a)] = r(a) \quad \text{— unbiased!}$

$\therefore$ can estimate reward for all actions

$\therefore$ can estimate expected reward for any policy $\pi$:

$\hat{R}(\pi) = \sum_{t=1}^{T-1} \hat{r}_{\tau}(\pi(x_{\tau})) = \hat{E}[\hat{r}(\pi(x)))]$

$\therefore$ can estimate regret of any policy $\pi$:

$\hat{\text{Regret}}(\pi) = \max_{\hat{\pi} \in \Pi} \hat{R}(\hat{\pi}) - \hat{R}(\pi)$

- can find maximizing $\hat{\pi}$ using AMO
De-biasing Biased Estimates (cont.)

• say \( r(a) \) = (unknown) reward for action \( a \)
  \( p(a) \) = (known) probability of choosing \( a \)

• define \( \hat{r}(a) = \begin{cases} \frac{r(a)}{p(a)} & \text{if } a \text{ chosen} \\ 0 & \text{else} \end{cases} \)

• then \( \mathbb{E}[\hat{r}(a)] = r(a) \)

De-biasing Biased Estimates (cont.)

- say $r(a) =$ (unknown) reward for action $a$
  $p(a) =$ (known) probability of choosing $a$

- define $\hat{r}(a) = \begin{cases} r(a)/p(a) & \text{if } a \text{ chosen} \\ 0 & \text{else} \end{cases}$

- then $E[\hat{r}(a)] = r(a)$ — unbiased!

∴ can estimate reward for all actions

∴ can estimate expected reward for any policy $\pi$

$\hat{R}(\pi) = \sum_{\tau=1}^{t-1} \hat{r}(\pi(x_{\tau})) = E[\hat{r}(\pi(x))]$

∴ can estimate regret of any policy $\pi$

$\hat{\text{Regret}}(\pi) = \max_{\hat{\pi} \in \Pi} \hat{R}(\hat{\pi}) - \hat{R}(\pi)$

• can find maximizing $\hat{\pi}$ using AMO
De-biasing Biased Estimates (cont.)

- say \( r(a) = \) (unknown) reward for action \( a \)
  \( p(a) = \) (known) probability of choosing \( a \)
- define \( \hat{r}(a) = \begin{cases} 
  r(a)/p(a) & \text{if } a \text{ chosen} \\
  0 & \text{else} 
\end{cases} \)
- then \( E[\hat{r}(a)] = r(a) \) — unbiased!
  \therefore \text{can estimate reward for all actions}
  \therefore \text{can estimate expected reward for any policy } \pi:

\[
\hat{R}(\pi) = \frac{1}{t-1} \sum_{\tau=1}^{t-1} \hat{r}_{\tau}(\pi(x_{\tau})) = \hat{E}[\hat{r}(\pi(x))] 
\]
De-biasing Biased Estimates (cont.)

• say $r(a) =$ (unknown) reward for action $a$
  $p(a) =$ (known) probability of choosing $a$

• define $\hat{r}(a) = \begin{cases} r(a)/p(a) & \text{if } a \text{ chosen} \\ 0 & \text{else} \end{cases}$

• then $E[\hat{r}(a)] = r(a)$ — unbiased!

∴ can estimate reward for all actions
∴ can estimate expected reward for any policy $\pi$:

$$\hat{R}(\pi) = \frac{1}{t-1} \sum_{\tau=1}^{t-1} \hat{r}_\tau(\pi(x_\tau)) = \hat{E}[\hat{r}(\pi(x))]$$

∴ can estimate regret of any policy $\pi$:

$$\text{Regret}(\pi) = \max_{\hat{\pi} \in \Pi} \hat{R}(\hat{\pi}) - \hat{R}(\pi)$$

• can find maximizing $\hat{\pi}$ using AMO
Variance Control

- estimates are unbiased — done?
Variance Control

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

no! — variance may be extremely large

can show \( \text{variance}(\hat{r}(a)) \leq \frac{1}{p(a)} \)
Variance Control

• estimates are unbiased — done?
• no! — variance may be extremely large
• can show \text{variance}(\hat{r}(a)) \leq \frac{1}{p(a)}

\therefore \text{to get good estimates, must ensure that } 1/p(a) \text{ not too large}
Randomizing over Policies

- need to choose actions (semi-)randomly
Randomizing over Policies

- need to choose actions (semi-)randomly
- approach: on each round,
  - compute distribution $Q$ over policy space $\Pi$
  - randomly pick $\pi \sim Q$
  - on current context $x$, choose action $\pi(x)$
Randomizing over Policies

• need to choose actions (semi-)randomly
• approach: on each round,
  • compute distribution $Q$ over policy space $\Pi$
  • randomly pick $\pi \sim Q$
  • on current context $x$, choose action $\pi(x)$
• $Q$ induces distribution over actions (for any $x$):

$$Q(a|x) = \Pr_{\pi \sim Q}[\pi(x) = a]$$
Randomizing over Policies

- need to choose actions (semi-)randomly
- approach: on each round,
  - compute distribution $Q$ over policy space $\Pi$
  - randomly pick $\pi \sim Q$
  - on current context $x$, choose action $\pi(x)$
- $Q$ induces distribution over actions (for any $x$):
  \[
  Q(a|x) = \Pr_{\pi \sim Q} [\pi(x) = a]
  \]
- seems will require time/space $O(|\Pi|)$ to compute $Q$ over space $\Pi$
  - will see later how to avoid!
How to Pick Q

- on each round, want to pick Q with:
  1. low (estimated) regret
     i.e., choose actions think will give high reward
How to Pick $Q$

- on each round, want to pick $Q$ with:
  1. low (estimated) regret
     i.e., choose actions think will give high reward
  2. low (estimated) variance
     i.e., ensure future estimates will be accurate
How to Pick $Q$

on each round, want to pick $Q$ with:

1. low (estimated) regret
   i.e., choose actions think will give high reward

2. low (estimated) variance
   i.e., ensure future estimates will be accurate
Low Regret

- \( \hat{\text{Regret}}(\pi) = \) estimated regret of \( \pi \)
Low Regret

- $\hat{\text{Regret}}(\pi) =$ estimated regret of $\pi$
- so: estimated regret for random $\pi \sim Q$ is

$$\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) = E_{\pi \sim Q} \left[ \hat{\text{Regret}}(\pi) \right]$$
Low Regret

- \( \hat{\text{Regret}}(\pi) = \) estimated regret of \( \pi \)
- so: estimated regret for random \( \pi \sim Q \) is

\[
\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) = E_{\pi \sim Q} [\hat{\text{Regret}}(\pi)]
\]

- want small:

\[
\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) \leq [\text{small}]
\]
Low Variance

\[
\frac{1}{Q(a|x)} = \text{variance of estimate of reward for action } a
\]
Low Variance

- \( \frac{1}{Q(a|x)} \) = variance of estimate of reward for action \( a \)
- so \( \frac{1}{Q(\pi(x)|x)} \) = variance if action chosen by \( \pi \)
Low Variance

- \( \frac{1}{Q(a|x)} \) = variance of estimate of reward for action \( a \)
- so \( \frac{1}{Q(\pi(x)|x)} \) = variance if action chosen by \( \pi \)
- can estimate expected variance for actions chosen by \( \pi \):

\[
\hat{V}^Q(\pi) = \hat{E} \left[ \frac{1}{Q(\pi(x)|x)} \right] = \frac{1}{t-1} \sum_{\tau=1}^{t-1} \frac{1}{Q(\pi(x_\tau)|x_\tau)}
\]
Low Variance

- \( \frac{1}{Q(a|x)} \) = variance of estimate of reward for action \( a \)
- so \( \frac{1}{Q(\pi(x)|x)} \) = variance if action chosen by \( \pi \)
- can estimate expected variance for actions chosen by \( \pi \):

\[
\hat{V}^Q(\pi) = \hat{E} \left[ \frac{1}{Q(\pi(x)|x)} \right] = \frac{1}{t-1} \sum_{\tau=1}^{t-1} \frac{1}{Q(\pi(x_\tau)|x_\tau)}
\]

- want small:

\[
\hat{V}^Q(\pi) \leq \text{[small]} \quad \text{for all } \pi \in \Pi
\]
Low Variance

- \( \frac{1}{Q(a|x)} \) = variance of estimate of reward for action \( a \)
- so \( \frac{1}{Q(\pi(x)|x)} \) = variance if action chosen by \( \pi \)
- can estimate expected variance for actions chosen by \( \pi \):

\[
\hat{\mathbb{V}}_Q(\pi) = \hat{\mathbb{E}} \left( \frac{1}{Q(\pi(x)|x)} \right) = \frac{1}{t-1} \sum_{\tau=1}^{t-1} \frac{1}{Q(\pi(x_\tau)|x_\tau)}
\]

- want small:

\[
\hat{\mathbb{V}}_Q(\pi) \leq \text{[small]} \quad \text{for all } \pi \in \Pi
\]

- detail: problematic if \( Q(a|x) \) too close to zero
Low Variance

\[ \frac{1}{Q^\mu(a|x)} = \text{variance of estimate of reward for action } a \]

\[ \frac{1}{Q^\mu(\pi(x)|x)} = \text{variance if action chosen by } \pi \]

\[ \text{so can estimate expected variance for actions chosen by } \pi: \]

\[ \hat{V}^Q(\pi) = \hat{\mathbb{E}} \left[ \frac{1}{Q^\mu(\pi(x)|x)} \right] = \frac{1}{t-1} \sum_{\tau=1}^{t-1} \frac{1}{Q^\mu(\pi(x_\tau)|x_\tau)} \]

\[ \text{want small:} \]

\[ \hat{V}^Q(\pi) \leq \text{[small]} \quad \text{for all } \pi \in \Pi \]

\[ \text{detail: problematic if } Q(a|x) \text{ too close to zero} \]

\[ \text{to avoid, “smooth” probabilities by occasionally picking action uniformly at random:} \]

\[ Q^\mu(a|x) = (1 - K\mu)Q(a|x) + \mu \]
Pulling Together

- want $Q$ such that:

$$\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) \leq \text{[small]}$$

$$\hat{V}^Q(\pi) \leq \text{[small]} \quad \text{for all } \pi \in \Pi$$
Pulling Together

• want $Q$ such that:

$$\sum_{\pi} Q(\pi) \overset{\text{Regret}(\pi)}{\leq} [\text{small}]$$

$$\hat{V}^Q(\pi) \leq [\text{small}] \quad \text{for all } \pi \in \Pi$$

$$\sum_{\pi} Q(\pi) = 1$$
Pulling Together

• want $Q$ such that:

$$\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) \leq C_0$$

$$C_1 \cdot \hat{\mathcal{V}}^Q(\pi) \leq C_0 \quad \text{for all } \pi \in \Pi$$

$$\sum_{\pi} Q(\pi) = 1$$

• can fill in constants
Pulling Together

- want $Q$ such that:

$$
\sum_{\pi} Q(\pi) \widehat{\text{Regret}}(\pi) \leq C_0
$$

$$
C_1 \cdot \widehat{V}^Q(\pi) \leq C_0 + \widehat{\text{Regret}}(\pi) \quad \text{for all } \pi \in \Pi
$$

$$
\sum_{\pi} Q(\pi) = 1
$$

- can fill in constants
- make easier by:
  - allowing higher variance for policies with higher regret
    (poor policies can be eliminated even with fairly poor performance estimates)
Pulling Together

- want $Q$ such that:

\[
\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) \leq C_0
\]

\[
C_1 \cdot \hat{V}^Q(\pi) \leq C_0 + \hat{\text{Regret}}(\pi) \quad \text{for all } \pi \in \Pi
\]

\[
\sum_{\pi} Q(\pi) \leq 1
\]

- can fill in constants
- make easier by:
  - allowing higher variance for policies with higher regret (poor policies can be eliminated even with fairly poor performance estimates)
  - only require $Q$ to be sub-distribution (can put all remaining mass on $\hat{\pi}$ with maximum estimated reward)
Optimization Problem “OP”

find $Q$ such that:

$$\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) \leq C_0$$

$$C_1 \cdot \hat{V}^Q(\pi) \leq C_0 + \hat{\text{Regret}}(\pi) \quad \text{for all } \pi \in \Pi$$

$$\sum_{\pi} Q(\pi) \leq 1$$
find $Q$ such that:

\[
\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) \leq C_0 \quad \text{[regret constraint]}
\]

\[
C_1 \cdot \hat{V}^Q(\pi) \leq C_0 + \hat{\text{Regret}}(\pi) \quad \text{for all } \pi \in \Pi \quad \text{[variance constraint]}
\]

\[
\sum_{\pi} Q(\pi) \leq 1 \quad \text{[sub-distribution]}
\]
find $Q$ such that:

\[
\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) \leq C_0 \quad \text{[regret constraint]}
\]

\[
C_1 \cdot \hat{V}^Q(\pi) \leq C_0 + \hat{\text{Regret}}(\pi) \quad \text{for all } \pi \in \Pi \quad \text{[variance constraint]}
\]

\[
\sum_{\pi} Q(\pi) \leq 1 \quad \text{[sub-distribution]}
\]

- similar to [Dudík et al.]
Optimization Problem “OP”

find $Q$ such that:

$$\sum_{\pi} Q(\pi) \widehat{\text{Regret}}(\pi) \leq C_0$$  \hspace{1cm} \text{[regret constraint]}

$$C_1 \cdot \widehat{V}^Q(\pi) \leq C_0 + \widehat{\text{Regret}}(\pi) \quad \text{for all } \pi \in \Pi$$ \hspace{1cm} \text{[variance constraint]}

$$\sum_{\pi} Q(\pi) \leq 1$$ \hspace{1cm} \text{[sub-distribution]}

• similar to [Dudík et al.]
• seems awful:
  • $|\Pi|$ variables
  • $|\Pi|$ constraints
  • constraints involve nasty non-linear functions
    (recall $\widehat{V}^Q(\pi) = \widehat{E} \left[ \frac{1}{Q^\mu(\pi(x)|x)} \right]$)
  • not even clear if feasible
• **Theorem:** if can solve OP on every round (for appropriate constants), then will get regret

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


Theorem: if can solve OP on every round (for appropriate constants), then will get regret

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

proof idea:
- regret constraint ensures low regret (if estimates are good enough)
- variance constraint ensures that they actually will be good enough

essentially same approach as [Dudík et al.]
How to Solve?

- basic idea:
  - find a violated constraint
  - (attempt to) fix it
  - repeat
How to Solve? (cont.)

- \( Q \leftarrow 0 \)
- repeat:
  1. if \( Q \) “too big” then rescale
     - (i.e., multiply \( Q \) by scalar \(< 1\)
     - ensures sub-distribution and regret constraints are satisfied
• \( Q \leftarrow 0 \)

• repeat:
  1. if \( Q \) “too big” then rescale
     • (i.e., multiply \( Q \) by scalar < 1)
     • ensures sub-distribution and regret constraints are satisfied
  2. find \( \pi \in \Pi \) for which corresponding variance constraint is violated
     a. if none exists, halt and output \( Q \)
     b. else \( Q(\pi) \leftarrow Q(\pi) + \alpha \) where \( \alpha = \) [some formula]
1. [detailed version]
   if $\sum_{\pi} Q(\pi)(C_0 + \hat{\text{Regret}}(\pi)) > C_0$ then rescale $Q$ (multiply by scalar $< 1$) so holds with equality
More Detail: Rescaling Step

1. [detailed version]
   if $\sum_{\pi} Q(\pi)(C_0 + \hat{\text{Regret}}(\pi)) > C_0$ then rescale $Q$ (multiply by scalar $< 1$) so holds with equality

- after this step, have

$$\sum_{\pi} Q(\pi)(C_0 + \hat{\text{Regret}}(\pi)) \leq C_0$$
1. [detailed version]
   if $\sum_{\pi} Q(\pi)\left(C_0 + \hat{\text{Regret}}(\pi)\right) > C_0$ then rescale $Q$ (multiply by scalar $< 1$) so holds with equality

   - after this step, have

   $$\sum_{\pi} Q(\pi)\left(C_0 + \hat{\text{Regret}}(\pi)\right) \leq C_0$$

   which implies:
   - $\sum_{\pi} Q(\pi) \leq 1$ [sub-distribution]
   - $\sum_{\pi} Q(\pi) \hat{\text{Regret}}(\pi) \leq C_0$ [regret constraint]
More Detail: Checking Variance Constraints

2. [detailed version]
   find \( \pi \in \Pi \) for which \( C_1 \cdot \hat{V}^Q(\pi) - \hat{\text{Regret}}(\pi) > C_0 \)
   a. if none exists, halt and output \( Q \)
   b. else \( Q(\pi) \leftarrow Q(\pi) + \alpha \) where \( \alpha = \) [some formula]
More Detail: Checking Variance Constraints

2. [detailed version]
   find $\pi \in \Pi$ for which $C_1 \cdot \hat{V}^Q(\pi) - \hat{\text{Regret}}(\pi) > C_0$
   a. if none exists, halt and output $Q$
   b. else $Q(\pi) \leftarrow Q(\pi) + \alpha$ where $\alpha = \text{[some formula]}$

• if halts then $C_1 \cdot \hat{V}^Q(\pi) \leq C_0 + \hat{\text{Regret}}(\pi)$ for all $\pi \in \Pi$
  [variance constraint]
More Detail: Checking Variance Constraints

2. [detailed version]
   find \( \pi \in \Pi \) for which \( C_1 \cdot \hat{V}^Q(\pi) - \hat{\text{Regret}}(\pi) > C_0 \)
   a. if none exists, halt and output \( Q \)
   b. else \( Q(\pi) \leftarrow Q(\pi) + \alpha \) where \( \alpha = \text{[some formula]} \)

- if halts then \( C_1 \cdot \hat{V}^Q(\pi) \leq C_0 + \hat{\text{Regret}}(\pi) \) for all \( \pi \in \Pi \)
  [variance constraint]

- can execute step using AMO:
  - can construct “pseudo-rewards” \( \tilde{r}_\tau \) for which (\( \forall \pi \)):

\[
C_1 \cdot \hat{V}^Q(\pi) - \hat{\text{Regret}}(\pi) = \sum_\tau \tilde{r}_\tau(\pi(x_\tau)) + \text{[constant]}
\]
More Detail: Checking Variance Constraints

2. [detailed version]
   find \( \pi \in \Pi \) for which \( C_1 \cdot \hat{V}^Q(\pi) - \hat{\text{Regret}}(\pi) > C_0 \)
   a. if none exists, halt and output \( Q \)
   b. else \( Q(\pi) \leftarrow Q(\pi) + \alpha \) where \( \alpha = \) [some formula]

   • if halts then \( C_1 \cdot \hat{V}^Q(\pi) \leq C_0 + \hat{\text{Regret}}(\pi) \) for all \( \pi \in \Pi \)  
     [variance constraint]

   • can execute step using AMO:
     • can construct “pseudo-rewards” \( \tilde{r}_\tau \) for which \( (\forall \pi) \):
       \[
       C_1 \cdot \hat{V}^Q(\pi) - \hat{\text{Regret}}(\pi) = \sum_\tau \tilde{r}_\tau(\pi(x_\tau)) + [\text{constant}] 
       \]

     • so: can maximize with AMO
     • will find violating constraint (if one exists)
2. [detailed version]
find $\pi \in \Pi$ for which $C_1 \cdot \hat{V}^Q(\pi) - \hat{\text{Regret}}(\pi) > C_0$

a. if none exists, halt and output $Q$
b. else $Q(\pi) \leftarrow Q(\pi) + \alpha$ where $\alpha = \text{[some formula]}$

- if halts then $C_1 \cdot \hat{V}^Q(\pi) \leq C_0 + \hat{\text{Regret}}(\pi)$ for all $\pi \in \Pi$

- can execute step using AMO:
  - can construct “pseudo-rewards” $\tilde{r}_\tau$ for which ($\forall \pi$):
    \[ C_1 \cdot \hat{V}^Q(\pi) - \hat{\text{Regret}}(\pi) = \sum_{\tau} \tilde{r}_\tau(\pi(x_\tau)) + \text{[constant]} \]
    
- so: can maximize with AMO
- will find violating constraint (if one exists)

∴ one AMO call per iteration
Why Does It Work?

- so: if halts, then outputs solution to OP
- but how long will it take to halt (if ever)?
- to answer, analyze using a potential function
A Potential Function

- define potential function to quantify progress:

\[
\Phi(Q) = A \cdot \hat{\mathbb{E}} \left[ \mathbb{E} \left( \text{uniform} \ || \ Q^\mu(\cdot|x)) \right) \right] + B \cdot \sum_{\pi} Q(\pi) \ \text{Regret}(\pi)
\]

\(\Phi(Q)\) defined for all nonnegative vectors \(Q\) over \(\Pi\) (not just sub-distributions)

- properties:
  - \(\Phi(Q) \geq 0\)
  - convex
  - if \(Q\) minimizes \(\Phi\) then \(Q\) is a solution to OP

- key proof step:
  \(\partial \Phi / \partial Q(\pi) \propto \text{variance constraint for } \pi\)

\(\therefore\) OP is feasible
A Potential Function

- define **potential function** to quantify progress:

\[
\Phi(Q) = A \cdot \hat{E} \left[ RE \left( \text{uniform} \parallel Q^\mu(\cdot | x) \right) \right] + B \cdot \sum_{\pi} Q(\pi) \text{Regret}(\pi)
\]

\[
\text{low variance}
\]

\[
\text{low regret}
\]

- defined for all **nonnegative vectors** \( Q \) over \( \Pi \)

(not just sub-distributions)
A Potential Function

- define potential function to quantify progress:
  \[
  \Phi(Q) = A \cdot \hat{E}[RE(\text{uniform } \parallel Q^\mu(\cdot|x))] + B \cdot \sum_{\pi} Q(\pi) \text{Regret}(\pi)
  \]
  - low variance
  - low regret

- defined for all nonnegative vectors \(Q\) over \(\Pi\) (not just sub-distributions)

- properties:
  - \(\Phi(Q) \geq 0\)
  - convex
A Potential Function

- define potential function to quantify progress:

\[
\Phi(Q) = A \cdot \hat{E} \left[ \text{RE (uniform \parallel Q^\mu(\cdot|x))} \right] + B \cdot \sum_{\pi} Q(\pi) \text{Regret}(\pi)
\]

  - low variance
  - low regret

- defined for all nonnegative vectors \( Q \) over \( \Pi \)
  - (not just sub-distributions)

- properties:
  - \( \Phi(Q) \geq 0 \)
  - convex
  - if \( Q \) minimizes \( \Phi \) then \( Q \) is a solution to OP
    - key proof step:
      - \( \partial \Phi / \partial Q(\pi) \propto \) variance constraint for \( \pi \)

\[\therefore \text{OP is feasible}\]
Analysis

- algorithm turns out to be (roughly) coordinate descent on $\Phi$
- each step adjusts $Q$ along one coordinate direction $Q(\pi)$
Analysis

- algorithm turns out to be (roughly) coordinate descent on $\Phi$
  - each step adjusts $Q$ along one coordinate direction $Q(\pi)$
- can lower-bound how much $\Phi$ decreases on each update
- can also show rescaling step never increases $\Phi$
Analysis

- algorithm turns out to be (roughly) **coordinate descent** on $\Phi$
  - each step adjusts $Q$ along one coordinate direction $Q(\pi)$
- can **lower-bound** how much $\Phi$ decreases on each update
- can also show rescaling step never increases $\Phi$
- since $\Phi \geq 0$, gives bound on number of iterations of the algorithm
Analysis

• algorithm turns out to be (roughly) *coordinate descent* on $\Phi$
  • each step adjusts $Q$ along one coordinate direction $Q(\pi)$
• can *lower-bound* how much $\Phi$ decreases on each update
• can also show rescaling step never increases $\Phi$
• since $\Phi \geq 0$, gives bound on number of iterations of the algorithm
• **Theorem**: On round $t$, algorithm halts after at most
  $$\tilde{O} \left( \sqrt{\frac{Kt}{\ln |\Pi|}} \right)$$ iterations (and calls to AMO).
Analysis

- algorithm turns out to be (roughly) coordinate descent on $\Phi$
  - each step adjusts $Q$ along one coordinate direction $Q(\pi)$
- can lower-bound how much $\Phi$ decreases on each update
- can also show rescaling step never increases $\Phi$
- since $\Phi \geq 0$, gives bound on number of iterations of the algorithm
- Theorem: On round $t$, algorithm halts after at most
  \[ \tilde{O}\left(\sqrt{\frac{Kt}{\ln|\Pi|}}\right) \]
  iterations (and calls to AMO).
- as corollary, also get bound on sparsity of $Q$
Epochs and Warm Start

• so far, assumed solve OP from scratch on each round
  • naively, gives $\tilde{O}(T^{3/2})$ calls to AMO in $T$ rounds
  • can do much better!

• first improvement: since data iid, can use same solution for many rounds, i.e., for long "epochs"
  • gives same (near optimal) regret
  • essentially no computation required on rounds where $Q$ not updated

• second improvement: can initialize algorithm with the previous solution (rather than starting fresh each time)
  • works because each new example can only cause $\Phi$ to increase slightly
Epochs and Warm Start

• so far, assumed solve OP from scratch on each round
  • naively, gives $\tilde{O}(T^{3/2})$ calls to AMO in $T$ rounds
  • can do much better!

• **first improvement**: since data iid, can use same solution for many rounds, i.e., for long “epochs”
  • gives same (near optimal) regret
  • essentially no computation required on rounds where $Q$ not updated
Epochs and Warm Start

- so far, assumed solve OP from scratch on each round
  - naively, gives $\tilde{O}(T^{3/2})$ calls to AMO in $T$ rounds
  - can do much better!
- first improvement: since data iid, can use same solution for many rounds, i.e., for long “epochs”
  - gives same (near optimal) regret
  - essentially no computation required on rounds where $Q$ not updated
- second improvement: can initialize algorithm with the previous solution (rather than starting fresh each time)
  - works because each new example can only cause $\Phi$ to increase slightly
Epochs and Warm Start (cont.)

- putting together:
  - if only update $Q$ on rounds 1, 4, 9, 16, 25, …
    - get same (near optimal) regret
    - only need $\tilde{O}(\sqrt{KT \ln |\Pi|})$

  calls to AMO total for entire sequence of $T$ rounds
Summary

• new algorithm for contextual bandits problem with AMO access
• (nearly) optimal regret
• simple and fast
• only requires an average of

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

AMO calls per round
Open Problems and Future Directions

- try out experimentally
- is there an algorithm that uses an online (rather than batch) oracle?
- is there a lower bound on number of AMO calls necessary to solve this problem?
- can we find a similar algorithm that handles adversarial data?