Taming the Monster: A Fast and Simple Algorithm for Contextual Bandits

Authors: Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, Robert E. Schapire

Microsoft Research, Columbia University, Yahoo! Labs, Princeton University

Table of Contents

Table of Contents

Problem & Motivation

Example in healthcare:

- Loop:
  1. Patient arrives with symptoms, medical history, genome, $\cdots$
  2. Doctor prescribes treatment
  3. Patient's health responses (e.g., better or worse)
- **Goal**: Build a robot doctor to prescribe treatments that yield good health outcomes

Problem & Motivation

Contextual bandit setting:

- Set X of contexts and K arms
- For $t \in [T]$ do
  1. Draws (x_t, r_t) from distribution D over $\mathcal{X} \times [0,1]^K$
  2. Observe context x_t (e.g., patient profile)
  3. Choose action $a_t \in [K]$ (e.g, prescribe treatments)
  4. Collect reward $r_t(a_t)$ (e.g., patient's health responses)
- **Goal**: algorithm for choosing a_t that yield high reward
- **Contextual setting**: use feature x_t to choose good action a_t
- **Bandit setting**: $r_t(a)$ for $a \neq a_t$ is not observed.
  - Exploration v.s. exploitation

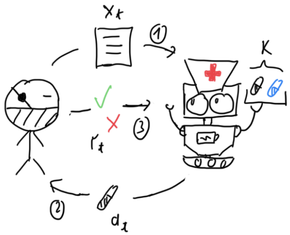

Figure: Healthcare example

Problem & Motivation

Contextual bandit setting:

- Set X of contexts and K arms
- For $t \in [T]$ do
  1. Draws (x_t, r_t) from distribution D over $\mathcal{X} \times [0,1]^K$
  2. Observe context x_t (e.g., patient profile)
  3. Choose action $a_t \in [K]$ (e.g, prescribe treatments)
  4. Collect reward $r_t(a_t)$ (e.g., patient's health responses)
- **Goal**: algorithm for choosing a_t that yield high reward
- **Contextual setting**: use feature x_t to choose good action a_t
- **Bandit setting**: $r_t(a)$ for $a \neq a_t$ is not observed.
  - Exploration v.s. exploitation

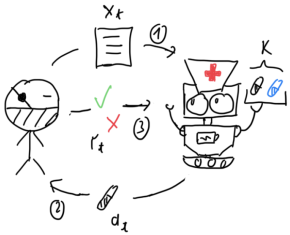

Figure: Healthcare example

Learning objective and difficulties

- **No single action is good in all situations, need to exploit context**
- **Policy class** $\prod$: set of functions from $\mathcal{X} \to [K]$
  (e.g., advice of experts, linear classifier, neural networks)
- **Regret** (i.e., relative performance to policy class π):

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

  ... **a strong benchmark if** $\prod$ **contains a policy with high reward**.
- Difficulties: feedback on action **only informs about subset of policies**; explicit bookkeeping is **computationally infeasible** when $\prod$ is large.

Problem & Motivation

arg max **oracle (AMO)**

- Given fully-labeled data $(x_1, r_1), \cdots, (x_t, r_t)$, AMO returns

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

- **Abstraction for efficient search** of policy class $\prod$, running time is polynomial, cost $\mathcal{O}(1)$.
- **In practice**: implement using standard heuristics (e.g., convex relax., backprop) for cost-sensitive multiclass learning algorithms.

Problem & Motivation

arg max **oracle (AMO)**

- Given fully-labeled data $(x_1, r_1), \cdots, (x_t, r_t)$, AMO returns

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

- **Abstraction for efficient search** of policy class $\prod$, running time is polynomial, cost $\mathcal{O}(1)$.
- **In practice**: implement using standard heuristics (e.g., convex relax., backprop) for cost-sensitive multiclass learning algorithms.

Contribution

- **New fast and simple algorithm for contextual bandits**
  - Optimal regret bound (up to log factor): $\tilde{\mathcal{O}}(\sqrt{KT \log |\prod|})$
  - Amortized $\tilde{\mathcal{O}}(\sqrt{K/T})$ calls to arg max oracle per round.
- **Comparison to previous work**
  - Thompson no general analysis
  - Exp4 algorithm: optimal regret, **maintains weights over $\prod$ at each round**
  - ϵ-greedy variant: **sub-optimal regret**, one AMO call/round
  - Monster paper: optimal regret, $\mathcal{O}(T^5 K^4)$ AMO calls/round

Problem & Motivation

arg max **oracle (AMO)**

- Given fully-labeled data $(x_1, r_1), \cdots, (x_t, r_t)$, AMO returns

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

- **Abstraction for efficient search** of policy class $\prod$, running time is polynomial, cost $\mathcal{O}(1)$.
- **In practice**: implement using standard heuristics (e.g., convex relax., backprop) for cost-sensitive multiclass learning algorithms.

Contribution

- **New fast and simple algorithm for contextual bandits**
  - Optimal regret bound (up to log factor): $\tilde{\mathcal{O}}(\sqrt{KT \log |\prod|})$
  - Amortized $\tilde{\mathcal{O}}(\sqrt{K/T})$ calls to arg max oracle per round.
- **Comparison to previous work**
  - Thompson no general analysis
  - Exp4 algorithm: optimal regret, **maintains weights over $\prod$ at each round**
  - ϵ-greedy variant: **sub-optimal regret**, one AMO call/round
  - Monster paper: optimal regret, $\mathcal{O}(T^5 K^4)$ AMO calls/round

Key techniques

- Action distributions, reward estimates via inverse probability weights (oldies but goodies)
- Algorithm for finding **policy distributions** that balance exploration/exploitation
- Warm-start/epoch trick

Table of Contents

Method

Basic algorithm structure (same as Exp4)

- Start with initial distribution $Q_1 \in \mathbb{R}^{\Pi}$, over policies $\prod$
- For $t \in [T]$
  1. Draw (x_t, r_t) i.i.d. from distribution D over $\mathcal{X} \times [0, 1]^K$
  2. Observe context x_t
  3. Compute distribution p_t over actions $[K]$ based on Q_t and x_t
  4. Draw action a_t from p_t
  5. Collect reward $r_t(a_t)$
  6. **Compute new distribution Q_{t+1} over policies $\prod$**

Method cont.

Inverse probability weighting (old trick)

- Importance-weighted estimate of reward from round t:

$$\hat{r}_t(a) := \frac{r_t(a_t) \cdot \mathbb{I}\{a = a_t\}}{p_t(a_t)}$$

- **Unbiased**, and has range and variance bounded by $1/p_t(a)$
- Can estimate total reward and regret of any policy:

$$\widehat{\text{Reward}}_t(\pi) = \sum_{i=1}^{t} \hat{r}_i(\pi(x_i))$$

$$\widehat{\text{Regret}}_t(\pi) = \max_{\pi' \in \prod} \widehat{\text{Reward}}_t(\pi') - \widehat{\text{Reward}}_t(\pi)$$

Method cont. & Regret

Constructing policy distributions

- Optimization problem: Find policy distribution Q s.t.

$$\sum_{\pi \in \prod} Q(\pi)\widehat{\text{Regret}}_t(\pi) \leq K\sqrt{t} \tag{1}$$

- **Low estimated regret (LR)** - skews distribution to put more mass on good policies (exploitation)

$$\frac{1}{t}\sum_{i=1}^{t} \frac{1}{Q(\pi(x_i)|x_i)} \leq K + \lambda \frac{\widehat{\text{Regret}}_t(\pi)}{\sqrt{t}}, \forall \pi \in \prod \tag{2}$$

- **Low estimation variance (LV)** - place sufficient mass on the actions chosen by each policy (exploration)

Method cont. & Regret

Constructing policy distributions

- Optimization problem: Find policy distribution Q s.t.

$$\sum_{\pi \in \prod} Q(\pi)\widehat{\text{Regret}}_t(\pi) \le K\sqrt{t} \tag{1}$$

- **Low estimated regret (LR)** - skews distribution to put more mass on good policies (exploitation)

$$\frac{1}{t}\sum_{i=1}^{t} \frac{1}{Q(\pi(x_i)|x_i)} \le K + \lambda \frac{\widehat{\text{Regret}}_t(\pi)}{\sqrt{t}}, \forall \pi \in \prod \tag{2}$$

- **Low estimation variance (LV)** - place sufficient mass on the actions chosen by each policy (exploration)

Theorem

If we obtain policy distributions Q_t via solving (OP), then with high probability, regret after T rounds is at most

$$\tilde{\mathcal{O}}\left(\sqrt{KT \log\left(\left|\prod\right|\right)}\right).$$

Method cont. & Regret

Constructing policy distributions

- Optimization problem: Find policy distribution Q s.t.

$$\sum_{\pi \in \prod} Q(\pi)\widehat{\text{Regret}}_t(\pi) \leq K\sqrt{t} \tag{1}$$

- **Low estimated regret (LR)** - skews distribution to put more mass on good policies (exploitation)

$$\frac{1}{t}\sum_{i=1}^{t} \frac{1}{Q(\pi(x_i)|x_i)} \leq K + \lambda \frac{\widehat{\text{Regret}}_t(\pi)}{\sqrt{t}}, \forall \pi \in \prod \tag{2}$$

- **Low estimation variance (LV)** - place sufficient mass on the actions chosen by each policy (exploration)

Theorem

If we obtain policy distributions Q_t via solving (OP), then with high probability, regret after T rounds is at most

$$\tilde{\mathcal{O}}\left(\sqrt{KT \log\left(\left|\prod\right|\right)}\right).$$

Sketch proof:

- *Lemma: By Eq. 2, then with high prob., each round t in epoch m, $\forall \pi \in \prod$,*
  $\text{Regret}_t(\pi) \leq 2\widehat{\text{Regret}}_t(\pi) + \mathcal{O}(K\mu_m)$, where $\mu_m := \min\{1/2K, \sqrt{\ln(16\tau_m^2|\prod|/\delta)/(K\tau_m)}\}, \forall m$.
- *Using Lemma and Eq. 1, then with high prob., at round t, $\sum_{\pi \in \prod} Q_{m-1}\text{Regret}_t(\pi) \leq \mathcal{O}(K\mu_{m-1})$.*
- *Summing these terms up over all T rounds and applying martingale concentration gives the Theorem.*

Algorithm

Basic algorithm structure (same as Exp4)

- Initial distribution $Q_1 \in \mathbb{R}^{\Pi}$, over policies $\prod$, epoch schedule $0 < \tau_1 < \tau_2 < \cdots$, history set $H_t = \emptyset$
- For $t \in [T]$
  1. Draw (x_t, r_t) i.i.d. from distribution D over $\mathcal{X} \times [0,1]^K$
  2. Observe context x_t
  3. Compute distribution p_t over actions $[K]$ based on Q_t and x_t
  4. Draw action a_t from p_t
  5. Collect reward $r_t(a_t)$
  6. **Save** $H_t \leftarrow (x_t, a_t, r_t(a_t), p_t)$
  7. **If** $t = \tau_m$**: compute Coordinate descent algorithm based on** H_t

Algorithm

Basic algorithm structure (same as Exp4)

- Initial distribution $Q_1 \in \mathbb{R}^{\prod}$, over policies $\prod$, epoch schedule $0 < \tau_1 < \tau_2 < \cdots$, history set $H_t = \emptyset$
- For $t \in [T]$
  1. Draw (x_t, r_t) i.i.d. from distribution D over $\mathcal{X} \times [0,1]^K$
  2. Observe context x_t
  3. Compute distribution p_t over actions $[K]$ based on Q_t and x_t
  4. Draw action a_t from p_t
  5. Collect reward $r_t(a_t)$
  6. **Save** $H_t \leftarrow (x_t, a_t, r_t(a_t), p_t)$
  7. **If** $t = \tau_m$**: compute Coordinate descent algorithm based on** H_t

Coordinate descent algorithm

- Input: Initial weights Q, history set H_t
- Loop:
  - Check OP conditions by Q and H_t
  - If (LR) $\sum_{\pi \in \prod} Q(\pi)\widehat{\text{Regret}}_t(\pi) \le K\sqrt{t}$ is violated, then replace Q by cQ
  - If there is a policy π causing (LV) $\frac{1}{t}\sum_{i=1}^{t} \frac{1}{Q(\pi(x_i)|x_i)} \le K + \lambda \frac{\widehat{\text{Regret}}_t(\pi)}{\sqrt{t}}$ to be violated, then
    - ★ Update $Q(\pi) = Q(\pi) + \alpha$
  - Else
    - ★ Return Q
- Claim: can check the if condition by making one AMO call per iteration
- Above, both $0 < c < 1$ and α have closed-form expressions

Computational complexity

Iteration bound for coordinate descent

- # steps of coordinate descent $= \tilde{\mathcal{O}}(\sqrt{Kt/\log|\prod|})$
- Also gives bound on the sparsity of Q
- Analysis via a potential function argument

Computational complexity

Iteration bound for coordinate descent

- $\#$ steps of coordinate descent $= \tilde{\mathcal{O}}(\sqrt{Kt/\log|\prod|})$
- Also gives bound on the sparsity of Q
- Analysis via a potential function argument

Warm-start

- If we warm-start coordinate descent (initialize with Q_t to get Q_{t+1}), then only need

$$\tilde{\mathcal{O}}(\sqrt{KT/\log|\prod|})$$

coordinate descent iterations over **all** T rounds

Computational complexity

Iteration bound for coordinate descent

- \# steps of coordinate descent $= \tilde{\mathcal{O}}(\sqrt{Kt/\log|\prod|})$
- Also gives bound on the sparsity of Q
- Analysis via a potential function argument

Warm-start

- If we warm-start coordinate descent (initialize with Q_t to get Q_{t+1}), then only need

$$\tilde{\mathcal{O}}(\sqrt{KT/\log|\prod|})$$

  coordinate descent iterations over **all** T rounds

Epoch trick

- **Regret analysis**: Q_t has low instantaneous expected regret (crucially relying on i.i.d. assumption).
  - Therefore same Q_t can be used for $\mathcal{O}(t)$ more rounds!
- If $\tau_m = m$, we need $\tilde{\mathcal{O}}(\sqrt{KT^3/\log|\prod|})$ AMO calls $\Rightarrow$ split T rounds into epochs, solve (OP) per each:
  - **Doubling**: only update on round $2^1, 2^2, 2^3, 2^4, \cdots$
    - Total of $\mathcal{O}(\log(T))$ updates, so overall # AMO calls unchanged (up to log factor)
  - **Squares**: only update on round $1^2, 2^2, 3^2, 4^2, \cdots$
    - Total of $\mathcal{O}(T^{1/2})$ updates, each requiring $\tilde{\mathcal{O}}\sqrt{K/\log|\prod|}$ AMO calls, on average

Empirical results

Table 1. Progressive validation loss, best hyperparameter values, and running times of various algorithm on RCV1.

Algorithm	ϵ-greedy	Explore-first	Bagging	LinUCB	Online Cover	Supervised
P.V. Loss	0.148	0.081	0.059	0.128	0.053	0.051
Searched	$0.1 = \epsilon$	2×10^5 first	16 bags	10^3 dim, minibatch-10	cover $n = 1$	nothing
Seconds	17	2.6	275	212×10^3	12	5.3

Figure: **Bandit problem derived from classification task (RCV1)**. Reporting progressive validation loss

- RCV1: document classification dataset, 781265 examples, and 47152 features.
- "Online Cover": variant with **stateful AMO**, i.e., set size $|H_t| = 1$
  - Achieves the best loss of 0.053
  - Efficient by only requires 12 seconds

Table of Contents

Wrap-up

Taming the Monster: a new fast and simple algorithm for contextual bandits

- Algorithm = Inverse probability weighting + solving Optimization Problem by coordinate descent with warm-start/epoch trick + arg max oracle.
- Optimal regret bound (up to log factor): $\tilde{\mathcal{O}}(\sqrt{KT \log |\prod|})$
- Amortized $\tilde{\mathcal{O}}(\sqrt{K/(T \log |\prod|)})$ calls to arg max oracle per round.