Pseudo-reward Algorithms for Contextual Bandits with Linear Payoff Functions

Ku-Chun Chou, Chao-Kai Chiang, Hsuan-Tien Lin, Chi-Jen Lu

from Chou’s MS thesis (algorithm) and part of Chiang’s Ph.D. thesis (theory)
National Taiwan University & Academia Sinica

ACML 2014, 11/28/2014
Contextual bandit problems

**Setting:** \textbf{online} game between algorithm $A$ and environment

for $t = 1, \cdots, T$:

1. $A$ observes context $x_t \in \mathbb{R}^d$ from the environment
2. $A$ selects an action $a_t \in [K] = \{1, 2, \cdots, K\}$
3. $A$ receives reward $r_{t,a_t} \in \mathbb{R}$ corresponding to $a_t$ from the environment
4. $A$ updates its selection strategy with $x_t$, $a_t$ and $r_{t,a_t}$

**Goal of $A$**

maximize average cumulative reward, $\frac{1}{T} \sum_{t=1}^{T} r_{t,a_t}$ by implementing

(2) $A.select(x_t)$

(4) $A.update(x_t, a_t, r_{t,a_t})$
ICML 2012 challenge

- news recommendation on Yahoo!’s front page
- 30 million user visits, 652 news articles

- design $A.\text{select}(x_t)$ and $A.\text{update}(x_t, a_t, r_{t,a_t})$
- aim for best click through rate (CTR)
ICML 2012 challenge (cont.)

for each user visit $t = 1, \cdots, T$ (30 million):

1. observes user features $x_t$ (gender, age, location, etc...)
2. selects an news article $a_t = A$.select($x_t$) to display to the user
3. receives a click ($r_{t,a_t} = 1$) or no-click ($r_{t,a_t} = 0$)
4. performs $A$.update($x_t, a_t, r_{t,a_t}$)

Achievement of Ku-Chun Chou
first place in 1st phase (otherwise cannot graduate :-))

<table>
<thead>
<tr>
<th>NAME</th>
<th>AFFILIATION</th>
<th>LAST SCORE (CTR * 10 000)</th>
<th>BEST SCORE (CTR * 10 000)</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ku-Chun</td>
<td>NTU</td>
<td>882.9</td>
<td>905.9</td>
<td>1</td>
</tr>
<tr>
<td>tvirot</td>
<td>MIT</td>
<td>903.9</td>
<td>903.9</td>
<td>2</td>
</tr>
<tr>
<td>edjoesu</td>
<td>MIT</td>
<td>889.9</td>
<td>903.4</td>
<td>3</td>
</tr>
</tbody>
</table>
Partial feedback

for $t = 1, \cdots, T$:

1. $\mathcal{A}$ observes context $x_t \in \mathbb{R}^d$ from the environment
2. $\mathcal{A}$ selects an action $a_t \in [K] = \{1, 2, \cdots, K\}$
3. $\mathcal{A}$ receives reward $r_{t,a_t} \in \mathbb{R}$ corresponding to $a_t$ from the environment
4. $\mathcal{A}$ updates its selection strategy with $x_t, a_t$ and $r_{t,a_t}$

- reward $r_{t,a_t}$ of the selected action $a_t$: revealed at $t$
- other rewards: unknown (such as $r_{3,a}$ or $r_{4,a}$ below)

$$X_{t,a} = \begin{pmatrix} - & x_1 & - \\ - & x_2 & - \\ - & x_5 & - \\ \vdots & & \vdots \end{pmatrix}, \quad r_{t,a} = \begin{pmatrix} r_{1,a} \\ r_{2,a} \\ r_{5,a} \\ \vdots \end{pmatrix}$$
Linear upper confidence bound (LinUCB)

- part of Ku-Chun’s winning solution (Li et al., WWW 2010; Chu et al., JMLR 2011)
- ridge regression on $X_{t,a}$ and $r_{t,a}$ to update weights $w_{t+1,a}$ only

**LinUCB.update($x_t$, $a_t$, $r_{t,a}$)**

$$w_{t+1,a} = \left( \lambda I + X_{t,a}^\top X_{t,a} \right)^{-1} X_{t,a}^\top r_{t,a}$$

$- (w_{t,a}^\top x)$ estimates reward of selecting action $a$ subject to $x$
- partial feedback $\Leftrightarrow$ need explore the less-certain actions
- select based on upper confidence bound of ridge regression

**LinUCB.select($x_t$)**

$$a_t = \arg\max_{a \in [K]} \left( \text{estimated reward} + \alpha \sqrt{x_t \left( \lambda I + X_{t-1,a}^\top X_{t-1,a} \right)^{-1} x_t} \right)$$

Chou et al. (NTU CSIE)

Pseudo-reward Algorithms for Contextual Bandits
Motivation: conquering partial feedback

- **LinUCB way**: enforce **exploration** through UCB — **slower** in some sense
- another idea: can we **CHEAT**? — what if all rewards revealed?

- yes, better than LinUCB, even with **noisy rewards**! — but **honor code**? :-)

---

legal (mimic) cheating $\iff$ pseudo-reward
Using $x_t$ with pseudo-reward

for unselected actions $a$

1. store $x_t$ into $\tilde{X}_{t,a}$
2. generate and store corresponding pseudo-reward $p_{t,a}$
3. use $(x_t, p_{t,a})$ to update $w_{t+1,a}$ as well

contexts/true rewards

$$X_{t,a} = \begin{pmatrix} - & x_1 & - \\ - & x_2 & - \\ - & x_5 & - \\ \vdots & & \vdots \\ r_{1,a} & & \\ r_{2,a} & & \\ r_{5,a} & & \\ \vdots & & \end{pmatrix}$$

$$r_{t,a} = \begin{pmatrix} r_{1,a} \\ r_{2,a} \\ r_{5,a} \\ \vdots \end{pmatrix}$$

contexts/pseudo-rewards

$$\tilde{X}_{t,a} = \begin{pmatrix} - & x_3 & - \\ - & x_4 & - \\ - & x_6 & - \\ \vdots & & \vdots \\ p_{3,a} & & \\ p_{4,a} & & \\ p_{6,a} & & \\ \vdots & & \end{pmatrix}$$

$$p_{t,a} = \begin{pmatrix} p_{3,a} \\ p_{4,a} \\ p_{6,a} \\ \vdots \end{pmatrix}$$
Designing a suitable pseudo-reward

\[ \text{LINPRUCB.update}(x_t, a_t, r_{t,a}) \]

\[ w_{t+1,a} = \arg\min_{w \in \mathbb{R}^d} \left( \lambda \|w\|^2 + \|X_{t,a}w - r_{t,a}\|^2 + \|\tilde{X}_{t,a}w - p_{t,a}\|^2 \right) \]

- **feasible** pseudo-reward: \textbf{estimate} of the actual reward
  - how about \( p_{t,a} = w_{t,a}^T x_t \)?
  - just \textbf{rechewing} \( w_{t,a} \)'s own predictions
- proposed pseudo-reward: \textbf{slight over-estimate} of actual reward
  - \( \approx \textbf{close} \) estimate
  - encourage \textbf{exploration} of the unselected action
  - how about \( p_{t,a} = w_{t,a}^T x_t + \beta \cdot (\text{inconfidence of } w_{t,a}) \)?
    —easily obtained by LinUCB-like calculations
Forgetting needed

- ratio of information from pseudo-rewards and true rewards:
  \[ \sim K - 1 : 1 \]

- \( w_{t,a} \) biased towards early, inaccurate pseudo-rewards

- proposed scheme: forgetting pseudo-rewards exponentially (see paper)
Linear pseudo-reward upper confidence bound (LinPRUCB)

**LINPRUCB.select\(x_t\)**

like LINUCB, but now with inconfidence term calculated with both \(X_{t,a}\) and (unforgotten) \(\tilde{X}_{t,a}\)

**pseudo-reward \(p_{t,a}\) for all unselected actions a**

\[ p_{t,a} := w_{t,a}^\top x_t + \beta \cdot \text{inconfidence term} \]

**LINPRUCB.update\(x_t, a_t, r_{t,a}\)**

\[ w_{t+1,a} = \arg\min_{w \in \mathbb{R}^d} \left( \lambda \|w\|^2 + \|X_{t,a}w - r_{t,a}\|^2 + \text{unforgotten} \|\tilde{X}_{t,a}w - p_{t,a}\|^2 \right) \]

similar theoretical guarantee to LinUCB in the long term
Long term performance on artificial simulations

Table: Comparisons of average cumulative reward.

<table>
<thead>
<tr>
<th></th>
<th>LINPRUCB</th>
<th>LINUCB</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>0.460 ± 0.010</td>
<td>0.461 ± 0.017</td>
</tr>
<tr>
<td>O</td>
<td>0.558 ± 0.005</td>
<td>0.563 ± 0.007</td>
</tr>
<tr>
<td>P</td>
<td>0.270 ± 0.008</td>
<td>0.268 ± 0.008</td>
</tr>
<tr>
<td>Q</td>
<td>0.297 ± 0.003</td>
<td>0.297 ± 0.005</td>
</tr>
</tbody>
</table>

- N: small $d$, small $K$
- O: small $d$, large $K$
- P: large $d$, small $K$
- Q: large $d$, large $K$

- LinPRUCB and LinUCB: roughly same long term performance (matching theory)
Short term performance on artificial simulations

- **LinPRUCB** better than **LinUCB** in the short term (*promising in practice*)
• using **slightly over-estimated** pseudo-reward improves short term performance
• **forgetting** reduces disadvantages of pseudo-rewards
• LinPRUCB similar to LinUCB in long term; **practically better in short term**
• other variants for **fast action selection**: see paper

Thank you! Questions?