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: online game between algorithm $\mathbb{A}$ and environment

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

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

Goal of $\mathbb{A}$

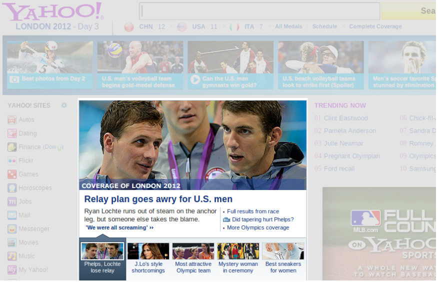
maximize average cumulative reward, $\frac{1}{T}\sum_{t=1}^{T} r_{t,a_t}$ by implementing

(2) $\mathbb{A}$.select($\mathbf{x}_t$)

(4) $\mathbb{A}$.update($\mathbf{x}_t$, a_t, r_{t,a_t})

ICML 2012 challenge

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



- design $\mathbb{A}$.select($\mathbf{x}_t$) and $\mathbb{A}$.update($\mathbf{x}_t$, 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 $\mathbf{x}_t$ (gender, age, location, etc...)
2. selects an news article $a_t = \mathbb{A}$.select($\mathbf{x}_t$) to display to the user
3. receives a click ($r_{t,a_t} = 1$) or no-click ($r_{t,a_t} = 0$)
4. performs $\mathbb{A}$.update($\mathbf{x}_t$, a_t, r_{t,a_t})

Achievement of Ku-Chun Chou

first place in 1st phase (**otherwise cannot graduate :-)**)

NAME	AFFILIATION	LAST SCORE (CTR * 10 000)	BEST SCORE (CTR * 10 000)	RANK
Ku-Chun	NTU	882.9	**905.9**	1
tvirot	MIT	903.9	**903.9**	2
edjoesu	MIT	889.9	**903.4**	3

Partial feedback

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

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

$$\mathbf{X}_{t,a} = \begin{pmatrix} - & \mathbf{x}_1 & - \\ - & \mathbf{x}_2 & - \\ - & \mathbf{x}_5 & - \\ & \vdots & \end{pmatrix}, \mathbf{r}_{t,a} = \begin{pmatrix} r_{1,a} \\ r_{2,a} \\ r_{5,a} \\ \vdots \end{pmatrix}$$

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 $\mathbf{X}_{t,a_t}$ and $\mathbf{r}_{t,a_t}$ to **update** weights $\mathbf{w}_{t+1,a_t}$ only

LinUCB.update($\mathbf{x}_t$, a_t, r_{t,a_t})

$$\mathbf{w}_{t+1,a_t} = \left(\lambda \mathbf{I} + \mathbf{X}_{t,a_t}^\top \mathbf{X}_{t,a_t}\right)^{-1} \left(\mathbf{X}_{t,a_t}^\top \mathbf{r}_{t,a_t}\right)$$

—($\mathbf{w}_{t,a}^\top \mathbf{x}$) estimates reward of selecting action a subject to $\mathbf{x}$

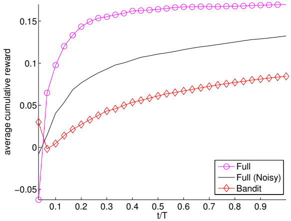
- partial feedback $\Leftrightarrow$ need explore the less-certain actions
  —**select** based on upper confidence bound of ridge regression

LinUCB.select($\mathbf{x}_t$)

$$a_t = \operatorname*{argmax}_{a \in [K]} \left(\underbrace{\mathbf{w}_{t,a}^\top \mathbf{x}_t}_{\text{estimated reward}} + \alpha \underbrace{\sqrt{\mathbf{x}_t \left(\lambda \mathbf{I} + \mathbf{X}_{t-1,a}^\top \mathbf{X}_{t-1,a}\right)^{-1} \mathbf{x}_t}}_{\text{inconfidence}} \right)$$

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 $\Longleftrightarrow$ pseudo-reward

Using $\mathbf{x}_t$ with pseudo-reward

for unselected actions a

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

contexts/true rewards

$$\mathbf{X}_{t,a} = \begin{pmatrix} - & \mathbf{x}_1 & - \\ - & \mathbf{x}_2 & - \\ - & \mathbf{x}_5 & - \\ & \vdots & \end{pmatrix},$$

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

contexts/pseudo-rewards

$$\tilde{\mathbf{X}}_{t,a} = \begin{pmatrix} - & \mathbf{x}_3 & - \\ - & \mathbf{x}_4 & - \\ - & \mathbf{x}_6 & - \\ & \vdots & \end{pmatrix},$$

$$\mathbf{p}_{t,a} = \begin{pmatrix} p_{3,a} \\ p_{4,a} \\ p_{6,a} \\ \vdots \end{pmatrix}$$

Designing a suitable pseudo-reward

LinPRUCB.update($\mathbf{x}_t$, a_t, r_{t,a_t})

$$\mathbf{w}_{t+1,a} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \left(\lambda\|\mathbf{w}\|^2 + \|\mathbf{X}_{t,a}\mathbf{w} - \mathbf{r}_{t,a}\|^2 + \|\tilde{\mathbf{X}}_{t,a}\mathbf{w} - \mathbf{p}_{t,a}\|^2\right)$$

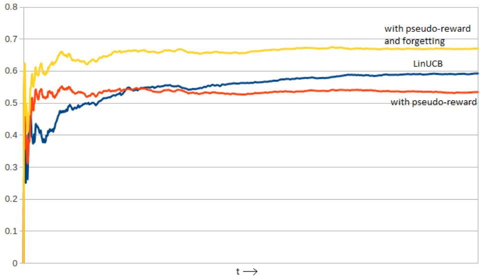
- **feasible** pseudo-reward: **estimate** of the actual reward
  - how about $p_{t,a} = \mathbf{w}_{t,a}^T\mathbf{x}_t$?
  - just **rechewing** $\mathbf{w}_{t,a}$'s own predictions
- proposed pseudo-reward: **slight over-estimate** of actual reward
  - $\approx$ **close** estimate
  - encourage **exploration**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

Forgetting needed

- ratio of information from pseudo-rewards and true rewards:

$$\simeq K - 1 : 1$$

- $\mathbf{w}_{t,a}$ biased towards early, inaccurate pseudo-rewards
- proposed scheme: forgetting pseudo-rewards **exponentially** (*see paper*)



Linear pseudo-reward upper confidence bound (LinPRUCB)

LinPRUCB.select($\mathbf{x}_t$)

like LinUCB, but now with

inconfidence term calculated with both $\mathbf{X}_{t,a}$ and (unforgotten) $\tilde{\mathbf{X}}_{t,a}$

pseudo-reward $p_{t,a}$ for *all* unselected actions a

$$p_{t,a} := \mathbf{w}_{t,a}^{\top}\mathbf{x}_t + \beta \cdot \text{inconfidence term}$$

LinPRUCB.update($\mathbf{x}_t$, a_t, r_{t,a_t})

$$\mathbf{w}_{t+1,a} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \left(\lambda\|\mathbf{w}\|^2 + \|\mathbf{X}_{t,a}\mathbf{w} - \mathbf{r}_{t,a}\|^2 + \text{unforgotten } \|\tilde{\mathbf{X}}_{t,a}\mathbf{w} - \mathbf{p}_{t,a}\|^2\right)$$

similar theoretical guarantee to LinUCB in the long term

Long term performance on artificial simulations

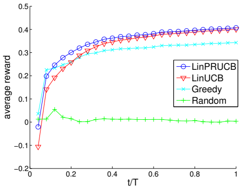
Table: Comparisons of average cumulative reward.

	LinPRUCB	LinUCB
N	0.460 ± 0.010	0.461 ± 0.017
O	0.558 ± 0.005	0.563 ± 0.007
P	0.270 ± 0.008	0.268 ± 0.008
Q	0.297 ± 0.003	0.297 ± 0.005

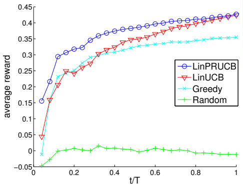
- N: small d, small K
- O: small d, large K
- P: large d, small K
- Q: large d, large K

- LinPRUCB and LinUCB: roughly same long term performance (*matching theory*)

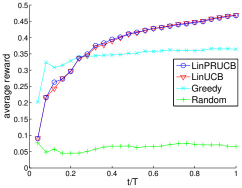
Short term performance on artificial simulations



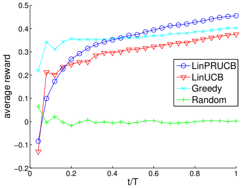
(a) $d = 10, K = 10$



(b) $d = 10, K = 30$



(c) $d = 30, K = 10$



(d) $d = 30, K = 30$

- LinPRUCB better than LinUCB **in the short term** (*promising in practice*)

Conclusion

- 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?