Exploration Scavenging

John Langford Alex Strehl Jennifer Wortman

Presented by Nikos Karamapatziakis and Devin Kennedy

8 April, 2010

Motivating application - web advertising

A common framework for many systems (ad = result)

- System shows an ad.
- User clicks on it if she likes it.
- What if the system had shown a different ad?
- System should learn to suggest relevant ads.

Motivating application - web advertising

A common framework for many systems (ad = result)

- System shows an ad.
- User clicks on it if she likes it.
- What if the system had shown a different ad?
- System should learn to suggest relevant ads.

The multi-armed bandit problem

Captures the essence of this setting, and many others.

Round 

The multi-armed bandit problem

Captures the essence of this setting, and many others.

Round				
1	?	?	1	?

The multi-armed bandit problem

Captures the essence of this setting, and many others.

Round				
1	?	?	1	?
2	?	0	?	?

The multi-armed bandit problem

Captures the essence of this setting, and many others.

Round				
1	?	?	1	?
2	?	0	?	?
3	5	?	?	?
4	?	?	?	2
5	0	?	?	?
6	?	?	?	4

The multi-armed bandit problem

Captures the essence of this setting, and many others.

Round				
1	5	0	1	2
2	0	0	3	2
3	5	3	1	3
4	0	1	2	2
5	0	1	7	2
6	0	1	0	4
Total	10	6	14	15

The multi-armed bandit problem

Captures the essence of this setting, and many others.

Round				
1	5	0	1	2
2	0	0	3	2
3	5	3	1	3
4	0	1	2	2
5	0	1	7	2
6	0	1	0	4
Total	10	6	14	15

Gambler's reward: $1 + 0 + 5 + 2 + 0 + 4 = 12$.
Gambler's *regret*: $15 - 12 = 3$

Exploration vs. exploitation

Suppose the gambler has discovered a slot machine with fairly good reward rate.

- Should he continue playing on (exploiting) that machine?
- What if he does?
- What if he does not?

A bandit algorithm (policy) must balance exploring different options and exploiting the best option so far.

UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

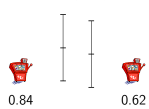
Play machine with highest $\hat{\mu}_j + c_j$.

UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

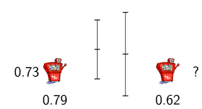


UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

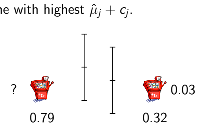


UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

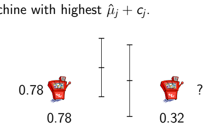


UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

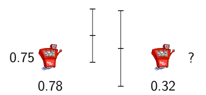


UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

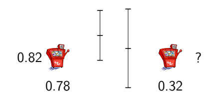


UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

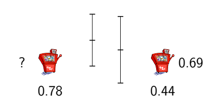


UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

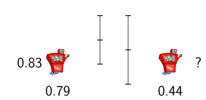


UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

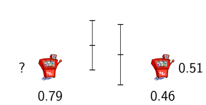


UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.

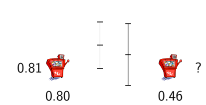


UCB1: a simple bandit algorithm

Initially play each machine once.
On round $t > k$ determine intervals $(\hat{\mu}_j - c_j, \hat{\mu}_j + c_j)$ s.t.

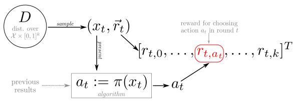
$$\Pr(\hat{\mu}_j - c_j < \hat{\mu}_j < \hat{\mu}_j + c_j) \geq 1 - \frac{1}{t^4}$$

Play machine with highest $\hat{\mu}_j + c_j$.



The contextual k-armed bandit problem

What is the *contextual* k-armed bandit problem?



Online multi-class classification with partial feedback.

The contextual k-armed bandit problem

Given an arbitrary input space $\mathcal{X}$ and a set of actions $\mathcal{A} = \{1, \cdots, k\}$, in each round t:

- a tuple $(x_t, \vec{r}_t)$ is drawn from some distribution over tuples of inputs and k-dimensional reward vectors, and x_t is presented to the algorithm;
- the algorithm chooses an action $a_t \in \mathcal{A}$;
- a reward r_{t,a_t} of action a_t is announced.

Goal: maximize the sum of rewards over the rounds of interaction.

Problem Statement

Suppose we already have a data set generated by following a policy (algorithm) π. Want to estimate the value of a *different* policy h:

$$V_D(h) := E_{(x,\vec{r})\sim D}[r_{h(x)}].$$

Where D is the distribution over tuples $(x, \vec{r})$ of inputs $x \in \mathcal{X}$ and rewards $\vec{r} \in [0,1]^k$.

When is this possible?

Impossibility Theorem

Evaluation is not possible when the exploration policy π depends on the current input. Consider the two problems (distributions) defined by:

	Under D		Under D'	
	$r_{t,0}$	$r_{t,1}$	$r_{t,0}$	$r_{t,1}$
$x_t = 0$	0	0	0	1
$x_t = 1$	0	1	1	1

Exploration policy: $\pi(x) = x$. Want to evaluate the policy $h(x) = 1 - x$. What happens?

Estimating value with special restrictions

Suppose we severely restrict the behavior of π:

- for each action a, π chooses *a exactly* T_a times, where $T_a > 0$;
- π chooses a_t independent of x_t.

Then for all D,

$$V_D(h) = E_{\{x_t,\vec{r}_t\}\sim D^T}\left[\sum_{t=1}^{T} \frac{r_{t,a_t} I(h(x_t) = a_t)}{T_{a_t}}\right].$$

Understanding the estimator

This expression for $V_D(h)$ is actually very simple.

reward for choosing a_t

indicator for when h acts the same as π

expected value over T rounds

$$E_{\{x_t,\vec{r}_t\}\sim D^T}\left[\sum_{t=1}^{T} \frac{r_{t,a_t} I(h(x_t) = a_t)}{T_{a_t}}\right]$$

number of times π chooses a_t

Quantifying usefulness of the estimator

For every sequence T of actions, for any $\delta \in (0,1)$, with probability $1-\delta$, it holds that

$$\left| V_D(h) - \sum_{t=1}^{T} \frac{r_{t,a_t} I(h(x_t) = a_t)}{T_{a_t}} \right| \leq \sum_{a=1}^{k} \sqrt{\frac{2\ln(2kT/\delta)}{T_a}}.$$

Accordingly, as $T \to \infty$, the estimator

$$\hat{V}_D(h) = \sum_{t=1}^{T} \frac{r_{t,a_t} I(h(x_t) = a_t)}{T_{a_t}}$$

grows arbitrarily close to $V_D(h)$ with probability 1.

Application

- Evaluation of different ad serving algorithms.
- Costly to evaluate on live system.
- Instead use proposed estimator with logged data.

How is this a contextual k-armed bandit problem?

Direct approach

- Input space $\mathcal{X}$ is set of all pages.
- Set of actions A is set of all advertisements.
- In each round, algorithm chooses an advertisement for the page. Reward computed based on user action.

Direct approach for *multiple* advertisements:

- Have an action for every slate of ads
  - exponentially large set of actions ...

Factoring Assumption

Assumption: probability of clicking ad a at position i on page x is

$$\mathcal{P}(x, a, i) = C_i \cdot \mathcal{P}(x, a)$$

$\mathcal{P}(x, a)$: position independent click through rate.
C_i: Attention Decay Coefficient (ADC). $C_1 = 1$

- Transform a slate of ℓ ads to ℓ examples.
- Set the reward for clicking on i-th ad to

$$r_i' = r / C_i \quad 1 \leq i \leq \ell$$

where r indicates whether the slate received clicks.

Estimating ADC Naively

New estimator:

$$\hat{V}_D(h) = \sum_{t=1}^{T} \sum_{i=1}^{\ell} \frac{r_i' C_{\sigma(a_i,x)}}{T_{a_i}}$$

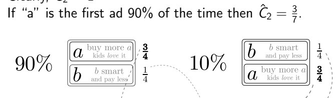
Only need to estimate C_i. However the straightforward

$$\hat{C}_i := \frac{\frac{\sum_a Clicks(a,i)}{\sum_a Impressions(a,i)}}{\frac{\sum_a Clicks(a,1)}{\sum_a Impressions(a,1)}}$$

is biased towards underestimating C_i.
Current policy already fairly good.

An example

Assume two slots, and two ads "a" and "b".
User always clicks: 3/4 of the time on "a", 1/4 on "b".
Clearly, $C_2 = 1$
If "a" is the first ad 90% of the time then $\hat{C}_2 = \frac{3}{7}$.



$$\hat{C}_2 = \frac{\frac{9}{10} \cdot \frac{1}{4} + \frac{1}{10} \cdot \frac{3}{4}}{\frac{9}{10} \cdot \frac{3}{4} + \frac{1}{10} \cdot \frac{1}{4}}$$

Better Estimator of ADC

Average the click-through rates instead of the clicks.

$$\hat{C}_i := \frac{\sum_a \lambda_a CTR(a,i)}{\sum_a \lambda_a CTR(a,1)}$$

λ_a should be set so as to minimize $\mathrm{Var}[\hat{C}_i]$.
Alternatively, set λ_a so as to minimize variance of

$$\sum_a \lambda_a CTR(a,i) + \sum_a \lambda_a CTR(a,1) \quad s.t \sum_a \lambda_a = 1$$

which is analytically tractable.

Empirical comparison

Estimating ADCs from Yahoo! logs leads to similar values as the much advocated $DCG(i) = 1/\log_2(i+2)$.

For evaluating ad serving policies restrict attention to h_π that reorder the results of π. Much smaller variance.

Two reordering policies were evaluated

- h_π reorders results of π according to their CTR.
- h'_π reorders results of π randomly.

Estimator is higher for h_π as expected.

Thank you

Questions?

A newer paper (Strehl, Langford, Kakade)

Lifts assumption that π, h do not depend on x_t

Estimate the probability that π will choose a. E.g.

$$\hat{\pi}(a|x) = \frac{|\{t|a_t = a \wedge x_t = x\}|}{|\{t|x_t = x\}|}$$

To evaluate a *non-adaptive* h

$$\hat{V}^h_{\hat{\pi}} = \frac{1}{|S|} \sum_{(x,a,r_a)\in S} \frac{r_a I(h(x) = a)}{\max\{\hat{\pi}(a|x), \tau\}}$$

Easy corollary of their theorem

If $\pi(h(x)|x) > \tau$ for all x then

$$E[|\hat{V}^h_{\hat{\pi}} - V^h|] \leq \frac{\sqrt{E_x[\max_a(\pi(a|x) - \hat{\pi}(a|x))^2]}}{\tau}$$

Assumption makes bound prettier; not necessary for the theorem.

Learning a policy

Let $C(x) = \{a|\hat{\pi}(a_t|x_t) > 0\}$
Learn a policy $h(x) = \mathrm{argmax}_{a\in C(x)} f(x,a)$ by minimizing

$$\sum_t \frac{(y_t - f(x_t, a_t))^2}{\max\{\hat{\pi}(a_t|x_t), \tau\}}$$

over a set of (x_t, a_t, y_t) triples. $y_t = 1$ iff a_t was clicked.
Finally estimate quality of h on test data by

$$\hat{V}^h_{\hat{\pi}} = \frac{1}{T}\sum_{t=1}^{T} \frac{y_t I(h(x_t) = a_t)}{\max\{\hat{\pi}(a_t|x_t), \tau\}}$$