Learning for Contextual Bandits

Alina Beygelzimer [1] John Langford [2]

IBM Research[1]

Yahoo! Research[2]

NYC ML Meetup, Sept 21, 2010

Example of Learning through Exploration

Repeatedly:

1. A user comes to Yahoo! (with history of previous visits, IP address, data related to his Yahoo! account)
2. Yahoo! chooses information to present (from urls, ads, news stories)
3. The user reacts to the presented information (clicks on something, clicks, comes back and clicks again, et cetera)

Yahoo! wants to interactively choose content and use the observed feedback to improve future content choices.

Another Example: Clinical Decision Making

Repeatedly:

1. A patient comes to a doctor with symptoms, medical history, test results
2. The doctor chooses a treatment
3. The patient responds to it

The doctor wants a policy for choosing targeted treatments for individual patients.

The Contextual Bandit Setting

For $t = 1, \ldots, T$:

1. The world produces some context $x_t \in X$
2. The learner chooses an action $a_t \in \{1, \ldots, K\}$
3. The world reacts with reward $r_t(a_t) \in [0, 1]$

Goal:

The Contextual Bandit Setting

For $t = 1, \ldots, T$:

1. The world produces some context $x_t \in X$
2. The learner chooses an action $a_t \in \{1, \ldots, K\}$
3. The world reacts with reward $r_t(a_t) \in [0, 1]$

Goal: Learn a good policy for choosing actions given context.

The Contextual Bandit Setting

For $t = 1, \ldots, T$:

1. The world produces some context $x_t \in X$
2. The learner chooses an action $a_t \in \{1, \ldots, K\}$
3. The world reacts with reward $r_t(a_t) \in [0, 1]$

Goal: Learn a good policy for choosing actions given context.

What does learning mean?

The Contextual Bandit Setting

For $t = 1, \ldots, T$:

1. The world produces some context $x_t \in X$
2. The learner chooses an action $a_t \in \{1, \ldots, K\}$
3. The world reacts with reward $r_t(a_t) \in [0, 1]$

Goal: Learn a good policy for choosing actions given context.

What does learning mean? Efficiently competing with a large reference class of possible policies $\Pi = \{\pi : X \to \{1, ..., K\}\}$:

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

The Contextual Bandit Setting

For $t = 1, \ldots, T$:

1. The world produces some context $x_t \in X$
2. The learner chooses an action $a_t \in \{1, \ldots, K\}$
3. The world reacts with reward $r_t(a_t) \in [0, 1]$

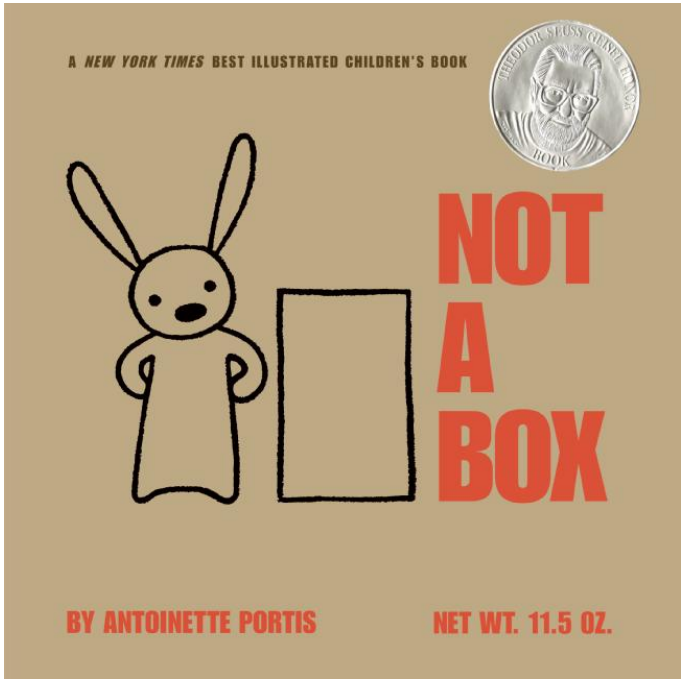
Goal: Learn a good policy for choosing actions given context.

What does learning mean? Efficiently competing with a large reference class of possible policies $\Pi = \{\pi : X \to \{1, ..., K\}\}$:

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

Other names: associative reinforcement learning, associative bandits, learning with partial feedback, bandits with side information

Basic Observation #1



This is not a supervised learning problem:

- We don't know the reward of actions not taken—loss function is unknown even at training time.
- Exploration is required to succeed (but still simpler than reinforcement learning – we know which action is responsible for each reward)

Basic Observation #2

This is not a bandit problem:

- In the bandit setting, there is no x, and the goal is to compete with the set of constant actions. Too weak in practice.
- Generalization across x is required to succeed.

Outline

1. How can we Learn?
   - online, stochastic
   - online, non-stochastic
2. Can we reuse Supervised Learning?
3. How can we Evaluate?
4. Extensions of the Setting

Idea 1: Follow the Leader

Reference class of policies Π

Follow the Leader Algorithm (FTL): For $t = 1, \ldots, T$:

- Let $\pi_t \in \Pi$ be the policy maximizing the sum of rewards $r(a)$ over the previous rounds $(x, a, r(a))$ where $\pi_t(x) = a$
- Observe features x_t
- Choose action $a_t = \pi_t(x_t)$
- Receive $r_t(a_t)$

Idea 1: Follow the Leader

Reference class of policies Π

Follow the Leader Algorithm (FTL): For $t = 1, \ldots, T$:

- Let $\pi_t \in \Pi$ be the policy maximizing the sum of rewards $r(a)$ over the previous rounds $(x, a, r(a))$ where $\pi_t(x) = a$
- Observe features x_t
- Choose action $a_t = \pi_t(x_t)$
- Receive $r_t(a_t)$

Even in the stochastic setting, expected regret of FTL can be $\Omega(T)$:

Idea 1: Follow the Leader

Reference class of policies Π

Follow the Leader Algorithm (FTL): For $t = 1, \ldots, T$:

- Let $\pi_t \in \Pi$ be the policy maximizing the sum of rewards $r(a)$ over the previous rounds $(x, a, r(a))$ where $\pi_t(x) = a$
- Observe features x_t
- Choose action $a_t = \pi_t(x_t)$
- Receive $r_t(a_t)$

Even in the stochastic setting, expected regret of FTL can be $\Omega(T)$:

Assume examples are drawn independently from D, and $\Pi = \{\pi_1 = a_1, \pi_2 = a_2\}$:

D	π_1	π_2
x_1 with probability $1/2$	0.1	0.1
x_2 with probability $1/2$	0	1

Expected regret is at least $(T-1)/8$: x_1 is generated in round 1, FTL chooses π_1, and then always acts according to π_1.

Idea 2: Explore τ then Follow the Leader (EFTL-τ)

EFTL-τ:

1. Choose an action uniformly at random for the first τ rounds
2. Let π = FTL on the first τ rounds
3. Use π for the remaining $T - \tau$ rounds

Idea 2: Explore τ then Follow the Leader (EFTL-τ)

EFTL-τ:

1. Choose an action uniformly at random for the first τ rounds
2. Let π = FTL on the first τ rounds
3. Use π for the remaining $T - \tau$ rounds

Suppose all examples are drawn independently from a fixed distribution D over $X \times [0,1]^K$.

Theorem:

For all D and Π, EFTL-τ has regret $O(T^{2/3}(K \ln |\Pi|)^{1/3})$ with high probability for $\tau \sim T^{2/3}(K \ln |\Pi|)^{1/3}$.

Theorem:

For all D and Π, EFTL-τ has regret $O(T^{2/3}(K \ln |\Pi|)^{1/3})$ with high probability for $\tau \sim T^{2/3}(K \ln |\Pi|)^{1/3}$.

Proof:

Let

$$\mathrm{FTL}_\tau(\pi) = \sum_{(x,a,r(a))} K \cdot \mathbf{1}[\pi(x) = a] r(a),$$

where $(x, a, r(a))$ ranges over the τ exploration examples.

A large deviation bound implies that with probability $1 - \delta$,

$$\frac{\mathrm{FTL}_\tau(\pi)}{\tau} \quad \text{deviates from} \quad \mathbb{E}_{(x,r_1,\ldots,r_K)\sim D}[r_{\pi(x)}]$$

by at most $\sqrt{\frac{K \ln(|\Pi|/\delta)}{\tau}}$ simultaneously for all $\pi \in \Pi$. Thus regret is bounded by

$$\tau + T\sqrt{\frac{K \ln(|\Pi|/\delta)}{\tau}}.$$

Optimizing τ completes the proof.

Unknown T: Dependence on T is removable by exploring with probability = deviation bound in each round [Langford, Zhang '07].

A key trick is to use *importance-weighted* empirical estimates of the reward of each policy π:

$$\hat{r}_t(\pi(x_t)) = \begin{cases} r_t/p_t(a_t) & \text{if } \pi(x_t) = a_t \\ 0 & \text{otherwise} \end{cases}$$

where $p_t(a_t) > 0$ is the probability of choosing action a_t in round t.

Idea 3: Exponential Weight Algorithm for Exploration and Exploitation with Experts

(EXP4) [Auer et al. '95]

Initialization: $\forall \pi \in \Pi : w_t(\pi) = 1$

For each $t = 1, 2, \ldots$:

1. Observe x_t and let for $a = 1, \ldots, K$

$$p_t(a) = (1 - K p_{\min}) \frac{\sum_\pi \mathbf{1}[\pi(x_t) = a]\, w_t(\pi)}{\sum_\pi w_t(\pi)} + p_{\min},$$

   where $p_{\min} = \sqrt{\frac{\ln |\Pi|}{KT}}$.

2. Draw a_t from p_t, and observe reward $r_t(a_t)$.

3. Update for each $\pi \in \Pi$

$$w_{t+1}(\pi) = \begin{cases} w_t(\pi) \exp\left(p_{\min} \frac{r_t(a_t)}{p_t(a_t)}\right) & \text{if } \pi(x_t) = a_t \\ w_t(\pi) & \text{otherwise} \end{cases}$$

Theorem: [Auer et al. '95] For all oblivious sequences $(x_1, r_1), \ldots, (x_T, r_T)$, EXP4 has expected regret

$$O\left(\sqrt{TK \ln |\Pi|}\right).$$

Theorem: [Auer et al. '95] For all oblivious sequences $(x_1, r_1), \ldots, (x_T, r_T)$, EXP4 has expected regret

$$O\left(\sqrt{TK \ln |\Pi|}\right).$$

Theorem: [Auer et al. '95] For any T, there exists an iid sequence such that the expected regret of any player is $\Omega(\sqrt{TK})$.

Theorem: [Auer et al. '95] For all oblivious sequences $(x_1, r_1), \ldots, (x_T, r_T)$, EXP4 has expected regret

$$O\left(\sqrt{TK \ln |\Pi|}\right).$$

Theorem: [Auer et al. '95] For any T, there exists an iid sequence such that the expected regret of any player is $\Omega(\sqrt{TK})$.

EXP4 can be modified to succeed with high probability [Beygelzimer, Langford, Li, Reyzin, Schapire '10].

The update step changes to use an upper confidence bound on its reward:

$$w_{t+1}(\pi) = w_t(\pi) \exp\left(\frac{p_{\min}}{2}\left(\mathbf{1}[\pi(x_t) = a_t]\frac{r_t(a_t)}{p_t(a_t)} + \frac{1}{p_t(\pi(x_t))}\sqrt{\frac{\ln N/\delta}{KT}}\right)\right)$$

Summary so far

	Supervised (ERM)	Explore then FTL	EXP4.P
Setting	stochastic, full	stochastic, bandit	adversarial, bandit
Regret	$O(\sqrt{T \ln N})$	$O(T^{2/3}(K \ln N)^{1/3})$	$O(\sqrt{TK \ln N})$
Oraclizable?	yes	yes	no

All are high probability results.

Outline

1. How can we Learn?
   - online, stochastic
   - online, non-stochastic
2. Can we reuse Supervised Learning?
   - Argmax Regression
   - Importance Weighted
   - Offset Tree
3. How can we Evaluate?
4. Extensions of the Setting

The Optimization Problem

How do you compute

$$\arg\max_{\pi \in \Pi} \sum_{(x,a,r,p)} \frac{r}{p} \mathbf{1}(\pi(x) = a)$$

for reasonable policy classes Π?

The Optimization Problem

How do you compute

$$\arg\max_{\pi \in \Pi} \sum_{(x,a,r,p)} \frac{r}{p} \mathbf{1}(\pi(x) = a)$$

for reasonable policy classes Π?

A tough question in general, but we can reuse solutions from supervised learning.

Approach 1: The Regression Approach

Fact: The minimizer of squared loss is the conditional mean.

1. Convert each example (x, a, r, p) into $((x, a), r, 1/p)$ where $1/p$ is the importance weight of predicting r on (x, a)
2. Learn a regressor f to predict r given (x, a)
3. Let $\pi_f(x) = \arg\max_a f(x, a)$.

Approach 1: The Regression Approach

Fact: The minimizer of squared loss is the conditional mean.

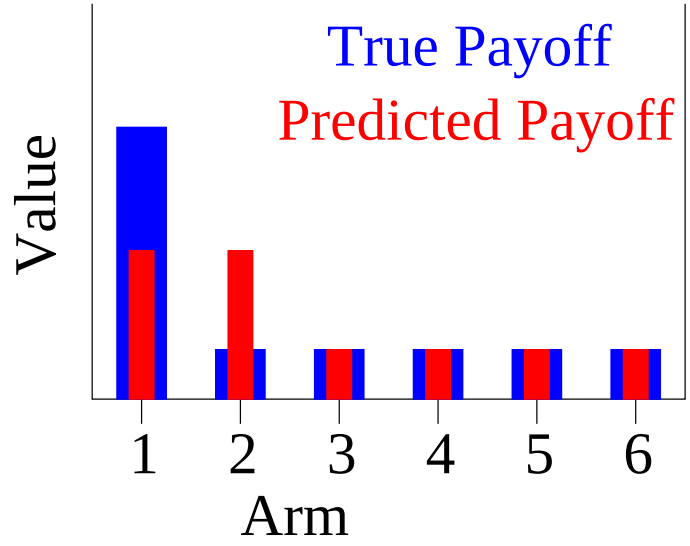
1. Convert each example (x, a, r, p) into $((x, a), r, 1/p)$ where $1/p$ is the importance weight of predicting r on (x, a)
2. Learn a regressor f to predict r given (x, a)
3. Let $\pi_f(x) = \arg\max_a f(x, a)$.

Theorem: For all D generating $(x, \vec{r})$, all probability distributions $p(a \mid x) > 0$ and f,

$$\underbrace{\text{policy-reg}(\pi_f, D)}_{\mathbb{E}_{(x,r)\sim D}\left[r_{\pi^*(x)} - r_{\pi_f(x)}\right]} \leq (2K \underbrace{\text{square-reg}(f, D)}_{\frac{1}{K}\sum_{a=1}^{K} \mathbb{E}_{x\sim D}(f(x,a) - \mathbb{E}_{\vec{r}\sim D|x}[r_a])^2})^{1/2}$$

where π^* is an optimal policy.

Proof sketch: Fix x. Worst case turns out to be:



The regressor's squared loss regret is $2\left(\frac{\mathbb{E}_{\vec{r}\sim D|x}\left[r_{f^*(x)}-r_{f(x)}\right]}{2}\right)^2$, out of k regression estimates. Thus the average squared loss regret is

$$\frac{1}{2k}\left(\mathbb{E}_{\vec{r}\sim D|x}\underbrace{\left[r_{f^*(x)}-r_{f(x)}\right]}_{\text{policy regret}}\right)^2.$$

Solving for policy regret, finishes the proof.

Approach 2: Importance-Weighted Classification Approach (Zadrozny'03)

1. For each (x, a, r, p) example, create an importance weighted multiclass example $(x, a, r/p)$, where

a	multiclass label
r/p	loss incurred if a is not predicted on x

2. Apply any importance weighted multiclass classification algorithm, and use the output classifier to make predictions.

Approach 2: Importance-Weighted Classification Approach (Zadrozny'03)

1. For each (x, a, r, p) example, create an importance weighted multiclass example $(x, a, r/p)$, where

a	multiclass label
r/p	loss incurred if a is not predicted on x

2. Apply any importance weighted multiclass classification algorithm, and use the output classifier to make predictions.

Importance-weighting multiclass classification can be reduced to binary classification using known techniques, giving the following theorem.

Theorem: Same quantification as before, for all binary classifiers,

$$\text{policy-reg} \leq 4K \text{ binary-reg}$$

Approach 3: The Offset Trick for $K = 2$ (two actions)

Partial label sample $(x, a, r, p) \mapsto$ binary importance weighted sample

$$\begin{cases} \left(x, a, \frac{r-\frac{1}{2}}{p}\right) & \text{if } r \geq \frac{1}{2} \\ \left(x, \overline{a}, \frac{\frac{1}{2}-r}{p}\right) & \text{if } r < \frac{1}{2} \end{cases}$$

$\overline{a} =$ the other label (action)

$\frac{\left|r-\frac{1}{2}\right|}{p} =$ importance weight (instead of r/p as before)

Approach 3: The Offset Trick for $K = 2$ (two actions)

Partial label sample $(x, a, r, p) \mapsto$ binary importance weighted sample

$$\begin{cases} \left(x, a, \frac{r-\frac{1}{2}}{p}\right) & \text{if } r \geq \frac{1}{2} \\ \left(x, \overline{a}, \frac{\frac{1}{2}-r}{p}\right) & \text{if } r < \frac{1}{2} \end{cases}$$

$\overline{a} =$ the other label (action)

$\frac{\left|r-\frac{1}{2}\right|}{p} =$ importance weight (instead of r/p as before)

Learn a binary classifier and use it as our policy

Induced binary distribution D'

- Draw contextual bandit sample $(x, r) \sim D$ and action a.
- With probability $\sim \frac{1}{p} \left| r - \frac{1}{2} \right|$:

  If $r \geq \frac{1}{2}$, generate (x, a); otherwise generate $(x, \overline{a})$.
- The induced problem is noisy. The importance trick reduces the range of importances, reducing the noise rate.

Induced binary distribution D'

- Draw contextual bandit sample $(x, r) \sim D$ and action a.
- With probability $\sim \frac{1}{p} \left| r - \frac{1}{2} \right|$:

  If $r \geq \frac{1}{2}$, generate (x, a); otherwise generate $(x, \overline{a})$.
- The induced problem is noisy. The importance trick reduces the range of importances, reducing the noise rate.

Example 1

Given example $(x, (1/2, 1))$, where x is a feature vector, $1/2$ is the reward of action Left, and 1 is the reward of action Right, what is the probability of generating (x, Left) and (x, Right)?

Induced binary distribution D'

- Draw contextual bandit sample $(x, r) \sim D$ and action a.
- With probability $\sim \frac{1}{p}\left|r - \frac{1}{2}\right|$:

  If $r \geq \frac{1}{2}$, generate (x, a); otherwise generate $(x, \overline{a})$.
- The induced problem is noisy. The importance trick reduces the range of importances, reducing the noise rate.

Example 1

Given example $(x, (1/2, 1))$, where x is a feature vector, $1/2$ is the reward of action Left, and 1 is the reward of action Right, what is the probability of generating (x, Left) and (x, Right)?

We draw action Left with probability p_{Left}, contributing probability $\frac{p_{\text{Left}}}{p_{\text{Left}}}|\frac{1}{2} - \frac{1}{2}| = 0$ to Left.
We draw action Right with probability p_{Right}, contributing probability $\frac{p_{\text{Right}}}{p_{\text{Right}}}|1 - \frac{1}{2}| = 1/2$ to Right.
Learn to predict Right.

Induced binary distribution D'

- Draw contextual bandit sample $(x, r) \sim D$ and action a.
- With probability $\sim \frac{1}{p} \left| r - \frac{1}{2} \right|$:

  If $r \geq \frac{1}{2}$, generate (x, a); otherwise generate $(x, \overline{a})$.
- The induced problem is noisy. The importance trick reduces the range of importances, reducing the noise rate.

Example 2

Given example $(x, (0, 1))$, where x is a feature vector, 0 is the reward of action Left, and 1 is the reward of action Right, what is the probability of generating (x, Left) and (x, Right)?

Induced binary distribution D'

- Draw contextual bandit sample $(x, r) \sim D$ and action a.
- With probability $\sim \frac{1}{p} \left| r - \frac{1}{2} \right|$:

  If $r \geq \frac{1}{2}$, generate (x, a); otherwise generate $(x, \overline{a})$.
- The induced problem is noisy. The importance trick reduces the range of importances, reducing the noise rate.

Example 2

Given example $(x, (0, 1))$, where x is a feature vector, 0 is the reward of action Left, and 1 is the reward of action Right, what is the probability of generating (x, Left) and (x, Right)?

We draw action Left with probability p_{Left}, contributing probability $\frac{p_{\text{Left}}}{p_{\text{Left}}} |0 - \frac{1}{2}| = 1/2$ to Right.
We draw action Right with probability p_{Right}, contributing probability $\frac{p_{\text{Right}}}{p_{\text{Right}}} |1 - \frac{1}{2}| = 1/2$ to Right.
Learn to predict Right, with double emphasis.

Induced binary distribution D'

- Draw contextual bandit sample $(x, r) \sim D$ and action a.
- With probability $\sim \frac{1}{p} \left| r - \frac{1}{2} \right|$:

  If $r \geq \frac{1}{2}$, generate (x, a); otherwise generate $(x, \overline{a})$.
- The induced problem is noisy. The importance trick reduces the range of importances, reducing the noise rate.

Example 3

Given example $(x, (0.75, 1))$, where x is a feature vector, 0.75 is the reward of action Left, and 1 is the reward of action Right, what is the probability of generating (x, Left) and (x, Right)?

Induced binary distribution D'

- Draw contextual bandit sample $(x, r) \sim D$ and action a.
- With probability $\sim \frac{1}{p} \left| r - \frac{1}{2} \right|$:

  If $r \geq \frac{1}{2}$, generate (x, a); otherwise generate $(x, \overline{a})$.
- The induced problem is noisy. The importance trick reduces the range of importances, reducing the noise rate.

Example 3

Given example $(x, (0.75, 1))$, where x is a feature vector, 0.75 is the reward of action Left, and 1 is the reward of action Right, what is the probability of generating (x, Left) and (x, Right)?

Action Left contributes probability $\frac{p_{\text{Left}}}{p_{\text{Left}}} |0.75 - \frac{1}{2}| = 1/4$ to Left.
Action Right contributes probability $\frac{p_{\text{Right}}}{p_{\text{Right}}} |1 - \frac{1}{2}| = 1/2$ to Right.
Action Right is preferred, with action Left occurring $1/3$ of the time. Noise rate is reduced from $3/7$ (with offset 0) to $1/3$.

Analysis for $K = 2$

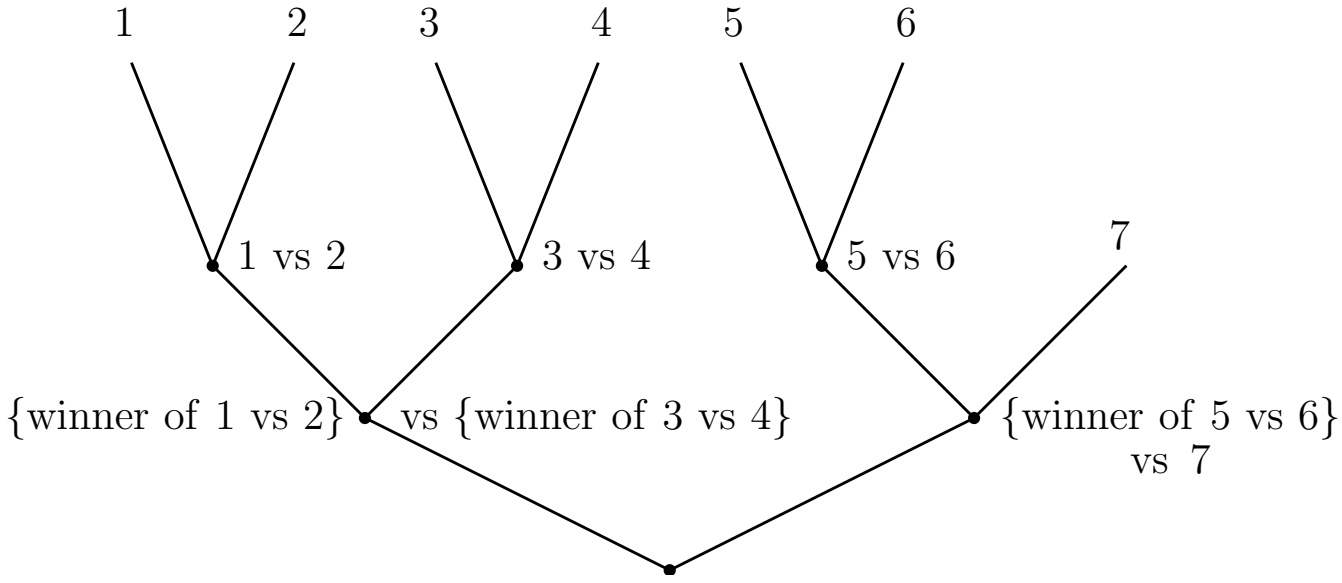
Binary Offset Theorem

For all 2-action contextual bandit problems D, all action choosing distributions, and all binary classifiers h,

$$\underbrace{\text{policy-reg}(h, D)}_{\mathbf{E}_{(x,\vec{r})\sim D}\left[r_{h^*(x)} - r_{h(x)}\right]} \leq \underbrace{\text{binary-reg}(h, D')}_{\text{err}(h,D') - \min_{h'} \text{err}(h',D')}$$

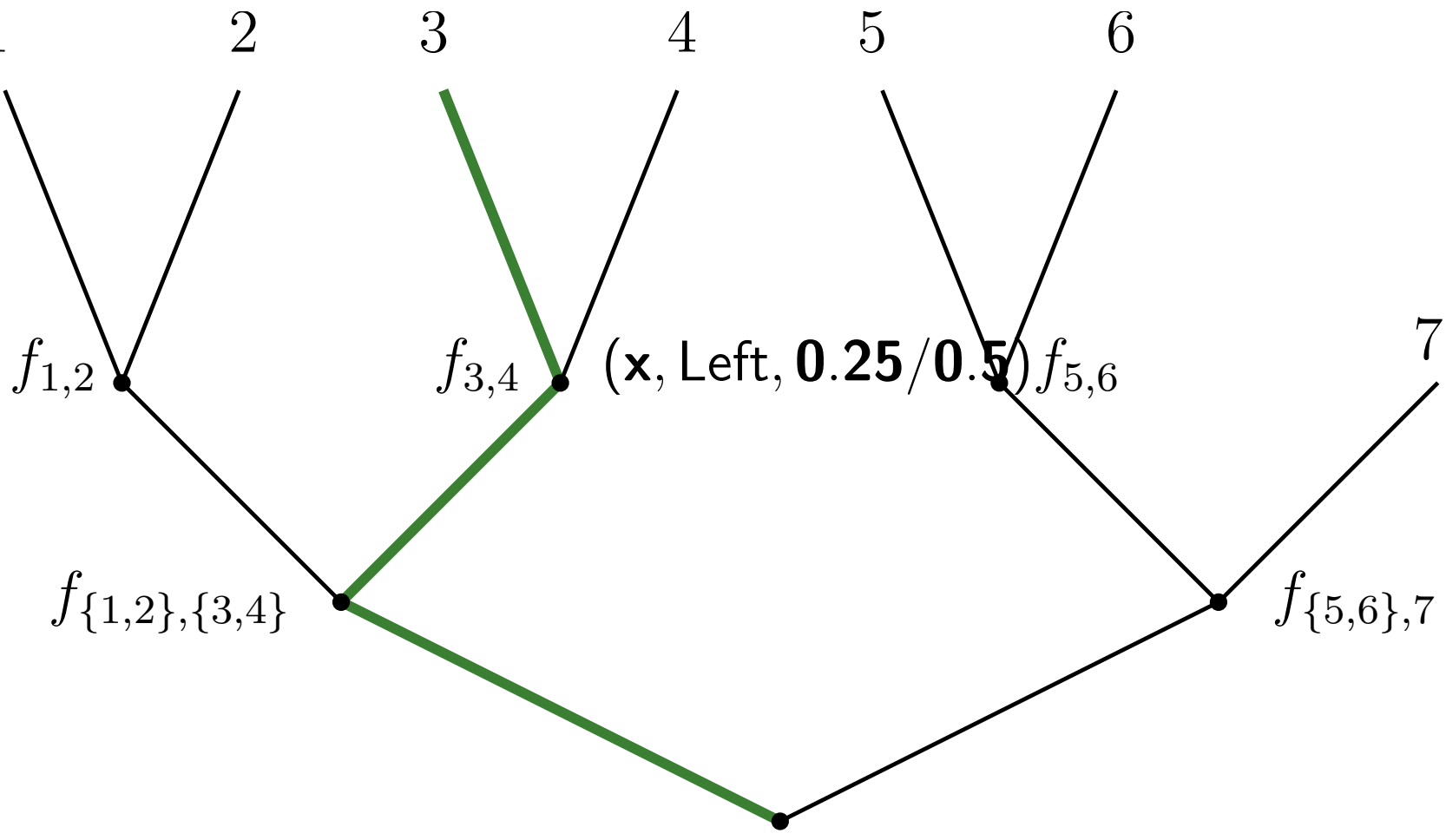
where h^* is an optimal policy. For $K = 2$, the policy using h is h.

Denoising for $K > 2$ arms

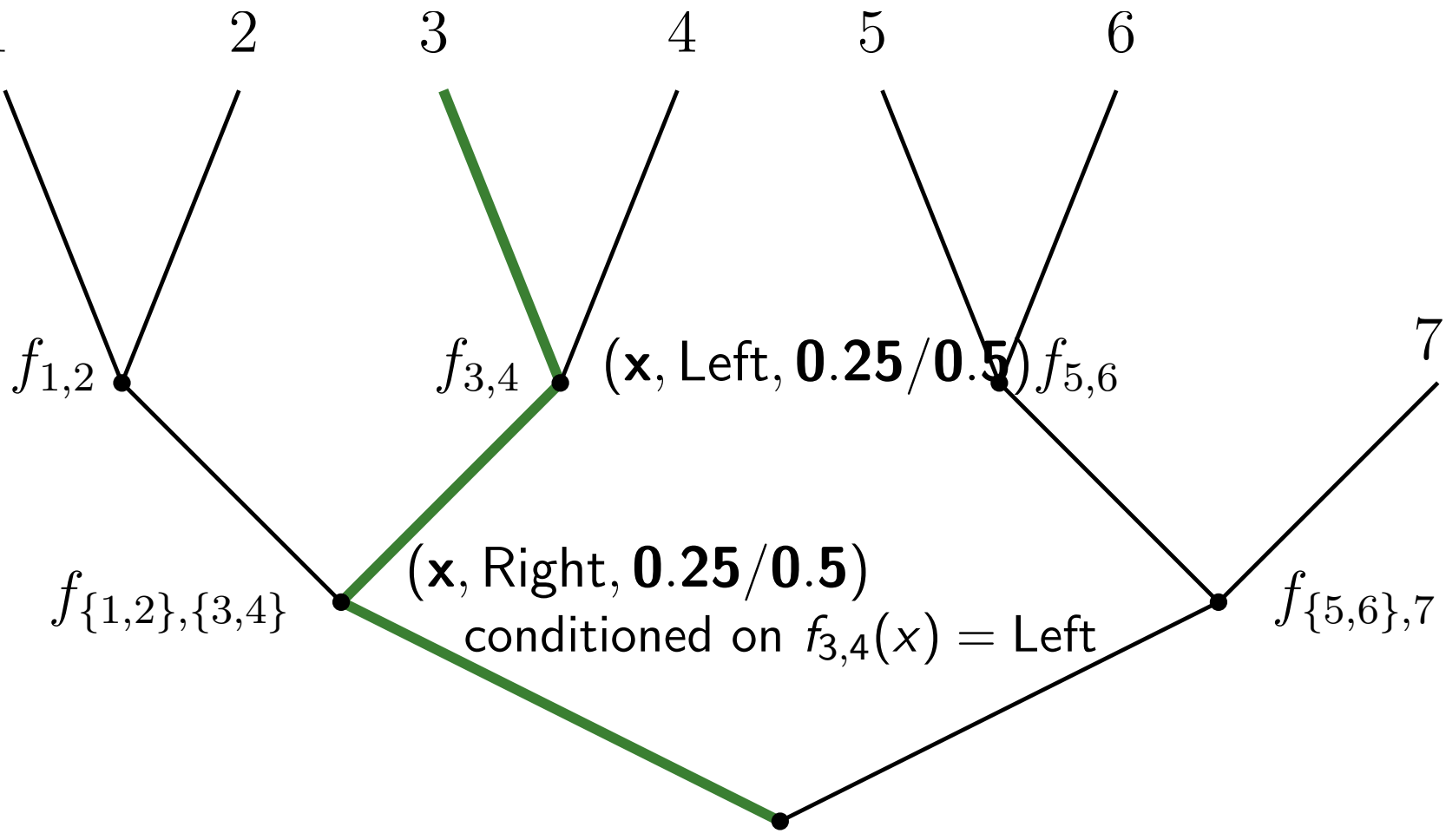


- Each non-leaf predicts the best of a pair of winners from the previous round.
- Use the same offsetting construction at each node.
- Filtering: A training example for a node is formed *conditioned* on the predictions of classifiers closer to the leaves.
- Policy: Follow the chain of predictions from root to leaf, output the leaf.

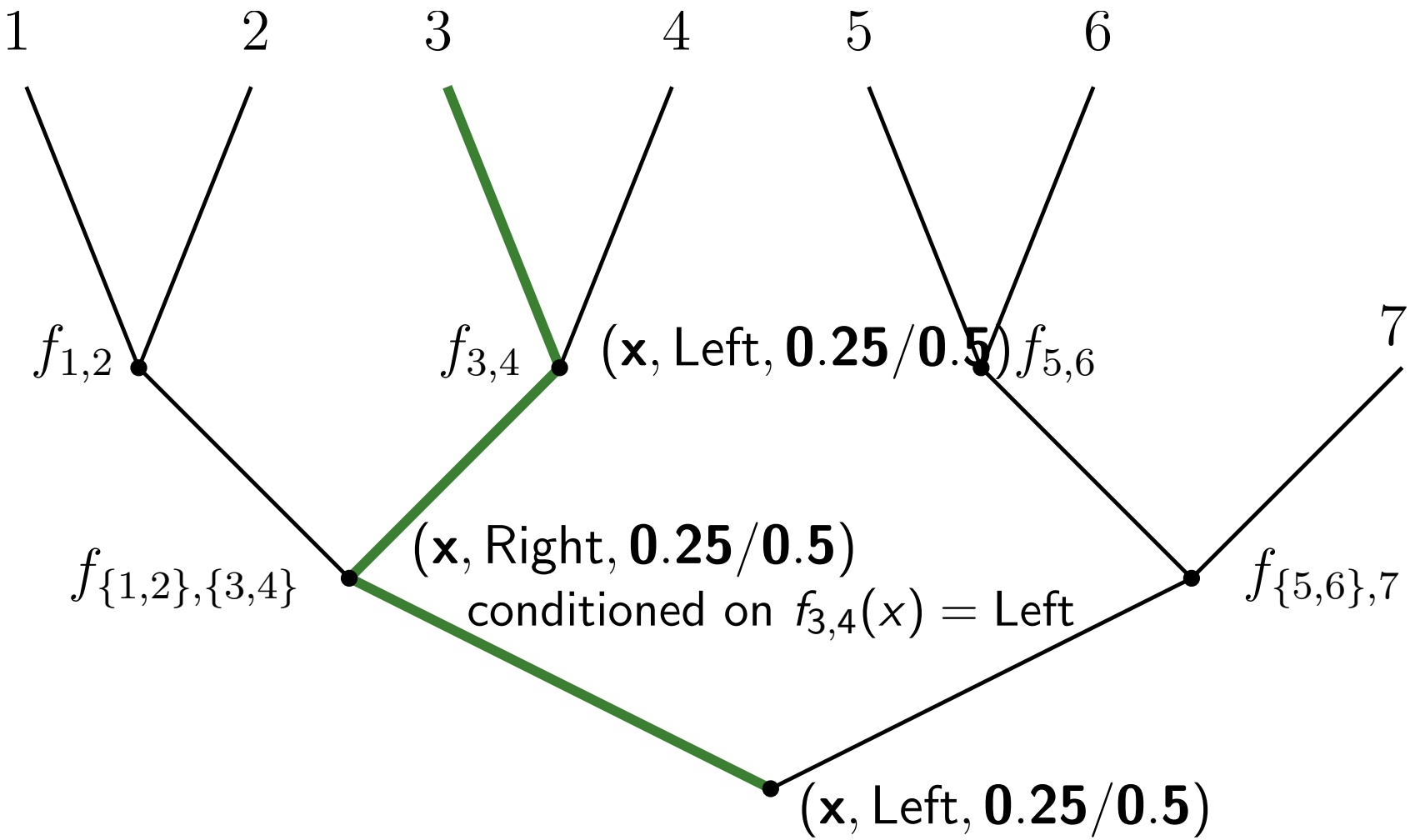
Training on example $(\mathbf{x}, \mathbf{3}, \mathbf{0.75}, \mathbf{0.5})$



Training on example $(\mathbf{x}, \mathbf{3}, \mathbf{0.75}, \mathbf{0.5})$



Training on example $(\mathbf{x}, \mathbf{3}, \mathbf{0.75}, \mathbf{0.5})$



Note: Can be composed with either batch or online base learners

Denoising with K arms: Analysis

D' = random binary problem according to chance that binary problem is fed an example under D.
h = binary classifier that predicts based on x and the choice of binary problem according to D'.
π_h = offset tree policy based on h.

Offset Tree Theorem

For all K-choice contextual bandit problems D and binary classifiers h:

$$\text{policy-reg}(\pi_h, D) \leq (K-1) \cdot \text{binary-reg}(h, D')$$

Lower bound: no reduction has a better regret analysis.

A Comparison of Approaches

Algorithm	Policy Regret Bound
Argmax Regression	$\sqrt{2K \text{ binary-reg}}$
Importance-weighting Classification	$4K$ binary-reg
Offset Tree	$(K-1)$ binary-reg

Experimentally, the performance order is the same.

Outline

1. How can we Learn?
   - online, stochastic
   - online, non-stochastic
2. Can we reuse Supervised Learning?
   - Argmax Regression
   - Importance Weighted
   - Offset Tree
3. How can we Evaluate?
   - A static policy
   - A dynamic policy
4. Setting Extensions

Olivier Chapelle  wanted a Learning to Rank challenge.

There was a training set, a leaderboard test set, and a completely heldout test set that determined the winner.

The challenge design:

1. Minimized bias towards particular methods.
2. Has a convergent quality estimator.

Can the same be done for contextual bandits?

The Evaluation Problem

Given data of the form $(x, a, r, p)^*$, how do we evaluate a contextual bandit solving algorithm?

The Evaluation Problem

Given data of the form $(x, a, r, p)^*$, how do we evaluate a contextual bandit solving algorithm?

Method 1: Deploy algorithm in the world.

The Evaluation Problem

Given data of the form $(x, a, r, p)^*$, how do we evaluate a contextual bandit solving algorithm?

Method 1: Deploy algorithm in the world.

1. Found company.
2. Get lots of business.
3. Deploy algorithm.

VERY expensive and VERY noisy.

How do we measure a Static Policy?

Let $\pi : X \rightarrow A$ be a policy mapping features to actions. How do we evaluate it?

How do we measure a Static Policy?

Let $\pi : X \to A$ be a policy mapping features to actions. How do we evaluate it?

Answer: Collect T samples of the form (x, a, p_a, r_a) where $p_a = p(a|x)$ is the probability of choosing action a, then evaluate:

$$\text{Value}(\pi) = \frac{1}{T} \sum_{(x,a,p_a,r_a)} \frac{r_a I(\pi(x) = a)}{p_a}$$

How do we measure a Static Policy?

Let $\pi : X \to A$ be a policy mapping features to actions. How do we evaluate it?

Answer: Collect T samples of the form (x, a, p_a, r_a) where $p_a = p(a|x)$ is the probability of choosing action a, then evaluate:

$$\text{Value}(\pi) = \frac{1}{T} \sum_{(x,a,p_a,r_a)} \frac{r_a I(\pi(x) = a)}{p_a}$$

Theorem: For all policies π, for all IID data distributions D, $\text{Value}(\pi)$ is an unbiased estimate of the expected reward of π:

$$E_{(x,\vec{r})\sim D}\left[r_{\pi(x)}\right] = E\text{Value}(\pi)$$

with deviations bounded by [Kearns et al. '00, adapted]:

$$O\left(\frac{1}{\sqrt{T \min p_a}}\right)$$

Proof: [Part 1] $\forall \pi, x, p(a), r_a$:

$E_{a\sim p}\left[\frac{r_a I(\pi(x)=a)}{p(a)}\right] = \sum_a p(a)\frac{r_a I(\pi(x)=a)}{p(a)} = r_{\pi(x)}$

How do we measure a Dynamic Policy?

For a dynamic policy π is dependent on the history. How do we know how good π is?

How do we measure a Dynamic Policy?

For a dynamic policy π is dependent on the history. How do we know how good π is?

Progressive_Validator(policy π, input $(x, a, r, p)^T$)

Let $h = \emptyset$ a history, $R = 0$. For each event (x, a, r, p)

1. If $\pi(h, x) = a$
2. then $R \leftarrow R + r/p$
3. $h \leftarrow h \cup (x, a, r, p)$

Return R/T

How do we measure a Dynamic Policy?

For a dynamic policy π is dependent on the history. How do we know how good π is?

Progressive_Validator(policy π, input $(x, a, r, p)^T$)

Let $h = \emptyset$ a history, $R = 0$. For each event (x, a, r, p)

1. If $\pi(h, x) = a$
2. then $R \leftarrow R + r/p$
3. $h \leftarrow h \cup (x, a, r, p)$

Return R/T

A bit strange: requires π to use externally chosen a and p.

How do we measure a Dynamic Policy?

For a dynamic policy π is dependent on the history. How do we know how good π is?

Progressive_Validator(policy π, input $(x, a, r, p)^T$)

Let $h = \emptyset$ a history, $R = 0$. For each event (x, a, r, p)

1. If $\pi(h, x) = a$
2. then $R \leftarrow R + r/p$
3. $h \leftarrow h \cup (x, a, r, p)$

Return R/T

A bit strange: requires π to use externally chosen a and p.

Theorem [Blum et al. '99, Cesa-Bianchi et al '04, Cesa-Bianchi & Gentile '08]: For all data distributions D, for all adaptively chosen sequences of policies $\pi_1, \pi_2, ...$ with high probability:

$$\left| \frac{1}{T} \sum_{(x,a,p_a,r_a)} \frac{r_a I(\pi_t(x) = a)}{p_a} - \frac{1}{T} \sum_t E_{(x,\vec{r})\sim D}[r_{\pi_t(x)}] \right| \leq O\left(\frac{1}{\sqrt{T \min p_a}} \right)$$

How do we measure a Dynamic Policy?

For a dynamic policy π is dependent on the history. How do we know how good π is?

Progressive_Validator(policy π, input $(x, a, r, p)^T$)

Let $h = \emptyset$ a history, $R = 0$. For each event (x, a, r, p)

1. If $\pi(h, x) = a$
2. then $R \leftarrow R + r/p$
3. $h \leftarrow h \cup (x, a, r, p)$

Return R/T

A bit strange: requires π to use externally chosen a and p.

Theorem [Blum et al. '99, Cesa-Bianchi et al '04, Cesa-Bianchi & Gentile '08]: For all data distributions D, for all adaptively chosen sequences of policies $\pi_1, \pi_2, ...$ with high probability:

$$\left| \frac{1}{T} \sum_{(x,a,p_a,r_a)} \frac{r_a I(\pi_t(x) = a)}{p_a} - \frac{1}{T} \sum_t E_{(x,\vec{r})\sim D}[r_{\pi_t(x)}] \right| \leq O\left(\frac{1}{\sqrt{T \min p_a}} \right)$$

Is this adequate?

How do we measure a Dynamic Policy?

For a dynamic policy π is dependent on the history. How do we know how good π is?

Progressive_Validator(policy π, input $(x, a, r, p)^T$)

Let $h = \emptyset$ a history, $R = 0$. For each event (x, a, r, p)

1. If $\pi(h, x) = a$
2. then $R \leftarrow R + r/p$
3. $h \leftarrow h \cup (x, a, r, p)$

Return R/T

A bit strange: requires π to use externally chosen a and p.

Theorem [Blum et al. '99, Cesa-Bianchi et al '04, Cesa-Bianchi & Gentile '08]: For all data distributions D, for all adaptively chosen sequences of policies $\pi_1, \pi_2, ...$ with high probability:

$$\left| \frac{1}{T} \sum_{(x,a,p_a,r_a)} \frac{r_a I(\pi_t(x) = a)}{p_a} - \frac{1}{T} \sum_t E_{(x,\vec{r})\sim D}[r_{\pi_t(x)}] \right| \leq O\left(\frac{1}{\sqrt{T \min p_a}} \right)$$

Is this adequate? No. This doesn't show policy convergence.

How do we measure a Dynamic Policy? Idea 2

Policy_Evaluator(policy π, input $(x, a, r)^T$ where a chosen uniform at random)

Let $h = \emptyset$ a history, $R = 0$

For each event (x, a, r)

1. If $\pi(h, x) = a$
2. then $h \leftarrow h \cup (x, a, r)$, $R \leftarrow R + r$

Return $R/|h|$

How do we measure a Dynamic Policy? Idea 2

Policy_Evaluator(policy π, input $(x, a, r)^T$ where a chosen uniform at random)

Let $h = \emptyset$ a history, $R = 0$

For each event (x, a, r)

1. If $\pi(h, x) = a$
2. then $h \leftarrow h \cup (x, a, r)$, $R \leftarrow R + r$

Return $R/|h|$

Theorem: [Li et al. '10] For all history lengths T, For all dynamic policies π, and all IID worlds D, the probability of a simulated history of length T = the probability of the same history of length T in the real world.

How do we measure a Dynamic Policy? Idea 2

Policy_Evaluator(policy π, input $(x, a, r)^T$ where a chosen uniform at random)

Let $h = \emptyset$ a history, $R = 0$

For each event (x, a, r)

1. If $\pi(h, x) = a$
2. then $h \leftarrow h \cup (x, a, r)$, $R \leftarrow R + r$

Return $R/|h|$

Theorem: [Li et al. '10] For all history lengths T, For all dynamic policies π, and all IID worlds D, the probability of a simulated history of length T = the probability of the same history of length T in the real world.

Easy proof by induction on history length.

Outline

1. How can we Learn?
   - online, stochastic
   - online, non-stochastic
2. Can we reuse Supervised Learning?
   - Argmax Regression
   - Importance Weighted
   - Offset Tree
3. How can we Evaluate?
   - A static policy
   - A dynamic policy
4. Setting Extensions
   - Missing *p*s.
   - Double Robust Policy Estimation
   - Linear Settings
   - Nearest Neighbor Settings

The Problem

Given logged data of the form $(x, a, r)^*$ where

1. x = features
2. a = a chosen action
3. r = the observed reward for the chosen action.

find a policy $\pi : x \to a$
maximizing $\text{Value}(\pi) = E_{(x,\vec{r})\sim D}\left[r_{\pi(x)}\right]$

The Problem

Given logged data of the form $(x, a, r)^*$ where

1. x = features
2. a = a chosen action
3. r = the observed reward for the chosen action.

find a policy $\pi : x \to a$
maximizing $\text{Value}(\pi) = E_{(x,\vec{r})\sim D}\left[r_{\pi(x)}\right]$
There is no $p(a|x)$!

The Problem

Given logged data of the form $(x, a, r)^*$ where

1. x = features
2. a = a chosen action
3. r = the observed reward for the chosen action.

find a policy $\pi : x \rightarrow a$
maximizing $\text{Value}(\pi) = E_{(x,\vec{r})\sim D}\left[r_{\pi(x)}\right]$
There is no $p(a|x)$!
Examples:

1. A fraction of all users were served by policy π_1 and the rest by policy π_2.

The Problem

Given logged data of the form $(x, a, r)^*$ where

1. x = features
2. a = a chosen action
3. r = the observed reward for the chosen action.

find a policy $\pi : x \to a$
maximizing $\text{Value}(\pi) = E_{(x,\vec{r})\sim D}\left[r_{\pi(x)}\right]$
There is no $p(a|x)$!
Examples:

1. A fraction of all users were served by policy π_1 and the rest by policy π_2.
2. Doctors in the US prescribe antibiotics more than doctors in Europe.

The Problem

Given logged data of the form $(x, a, r)^*$ where

1. x = features
2. a = a chosen action
3. r = the observed reward for the chosen action.

find a policy $\pi : x \to a$
maximizing $\text{Value}(\pi) = E_{(x,\vec{r})\sim D}\left[r_{\pi(x)}\right]$
There is no $p(a|x)$!
Examples:

1. A fraction of all users were served by policy π_1 and the rest by policy π_2.
2. Doctors in the US prescribe antibiotics more than doctors in Europe.
3. An ad runs out of budget and is removed from consideration.

Main Result

Define: $p(a|x) = \Pr_{t \sim U(1 \ldots T)}(\pi_t(x) = a)$.

Main Result

Define: $p(a|x) = \Pr_{t \sim U(1 \ldots T)}(\pi_t(x) = a)$.

Learn predictor $\hat{p}(a|x)$ of $p(a|x)$ on $(x, a)^*$ data.

Main Result

Define: $p(a|x) = \Pr_{t \sim U(1 \ldots T)}(\pi_t(x) = a)$.

Learn predictor $\hat{p}(a|x)$ of $p(a|x)$ on $(x, a)^*$ data.

Define: $\hat{V}(\pi) = \hat{E}_{x,a,r_a}\left[\frac{r_a I(h(x)=a)}{\max\{\tau, \hat{p}(a|x)\}}\right]$ where $\tau =$ small number.

Main Result

Define: $p(a|x) = \Pr_{t \sim U(1 \ldots T)}(\pi_t(x) = a)$.

Learn predictor $\hat{p}(a|x)$ of $p(a|x)$ on $(x, a)^*$ data.

Define: $\hat{V}(\pi) = \hat{E}_{x,a,r_a}\left[\frac{r_a I(h(x)=a)}{\max\{\tau, \hat{p}(a|x)\}}\right]$ where $\tau =$ small number.

Theorem [Strehl et al. '10]: For all IID D, for all logging policy sequences $\pi_1, \ldots, \pi_T$, for all policies π with $p(a|x) \geq \tau$:

$$\text{Value}(\pi) - \frac{\sqrt{\text{reg}(\hat{p})}}{\tau} \leq E\hat{V}(\pi) \leq \text{Value}(\pi) - \frac{\sqrt{\text{reg}(\hat{p})}}{\tau}$$

where $\text{reg}(\hat{p}) = E_x(p(a|x) - \hat{p}(a|x))^2 =$ squared loss regret.

Main Result

Define: $p(a|x) = \Pr_{t \sim U(1...T)}(\pi_t(x) = a)$.

Learn predictor $\hat{p}(a|x)$ of $p(a|x)$ on $(x, a)^*$ data.

Define: $\hat{V}(\pi) = \hat{E}_{x,a,r_a}\left[\frac{r_a I(h(x)=a)}{\max\{\tau, \hat{p}(a|x)\}}\right]$ where $\tau =$ small number.

Theorem [Strehl et al. '10]: For all IID D, for all logging policy sequences $\pi_1, ..., \pi_T$, for all policies π with $p(a|x) \geq \tau$:

$$\text{Value}(\pi) - \frac{\sqrt{\text{reg}(\hat{p})}}{\tau} \leq E\hat{V}(\pi) \leq \text{Value}(\pi) - \frac{\sqrt{\text{reg}(\hat{p})}}{\tau}$$

where $\text{reg}(\hat{p}) = E_x(p(a|x) - \hat{p}(a|x))^2 =$ squared loss regret.

What happens when $p(a|x) < \tau$?

Main Result

Define: $p(a|x) = \Pr_{t \sim U(1...T)}(\pi_t(x) = a)$.

Learn predictor $\hat{p}(a|x)$ of $p(a|x)$ on $(x, a)^*$ data.

Define: $\hat{V}(\pi) = \hat{E}_{x,a,r_a}\left[\frac{r_a I(h(x)=a)}{\max\{\tau, \hat{p}(a|x)\}}\right]$ where $\tau =$ small number.

Theorem [Strehl et al. '10]: For all IID D, for all logging policy sequences $\pi_1, ..., \pi_T$, for all policies π with $p(a|x) \geq \tau$:

$$\text{Value}(\pi) - \frac{\sqrt{\text{reg}(\hat{p})}}{\tau} \leq E\hat{V}(\pi) \leq \text{Value}(\pi) - \frac{\sqrt{\text{reg}(\hat{p})}}{\tau}$$

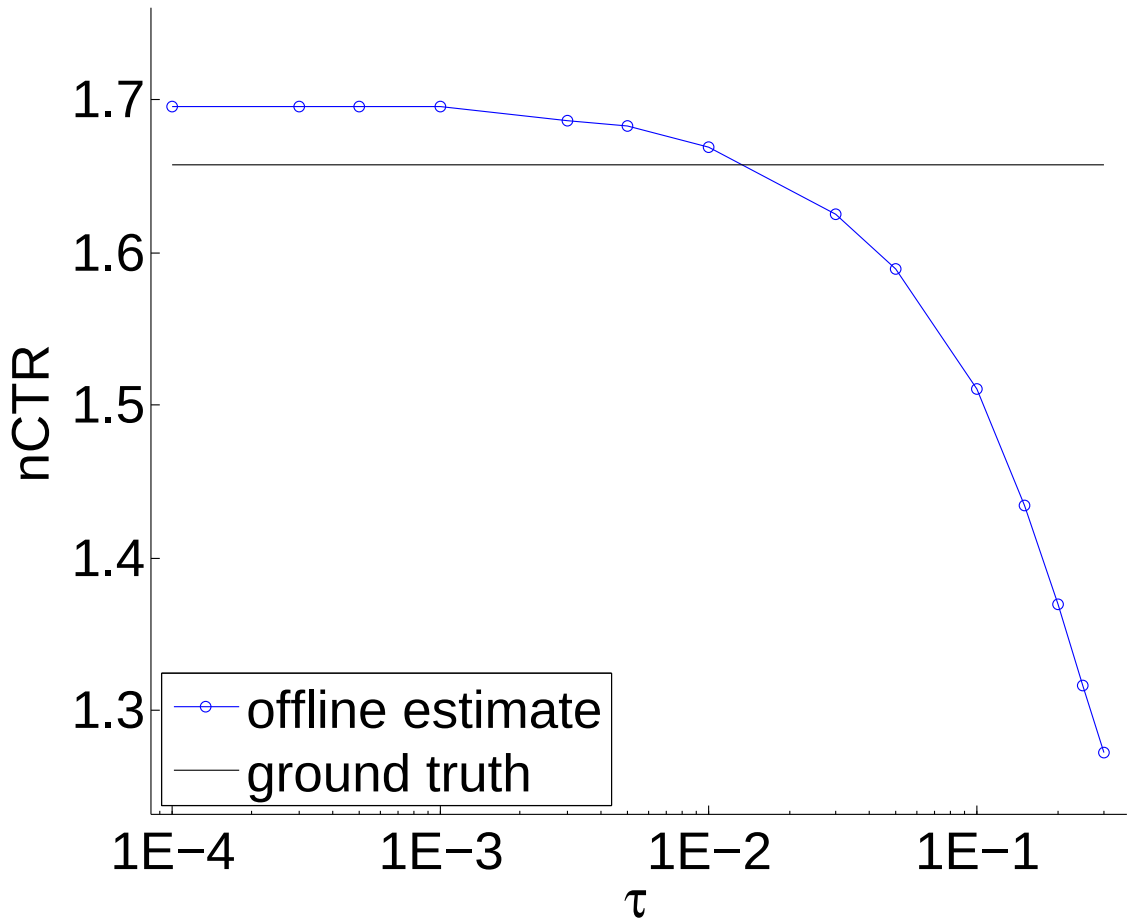
where $\text{reg}(\hat{p}) = E_x(p(a|x) - \hat{p}(a|x))^2 =$ squared loss regret.

What happens when $p(a|x) < \tau$? Bias growing with gap.

Experimental Results I

Dataset = $64.7M$ (x, a, r) triples with a uniform at random.
Actions a range over 20 choices.

1. run deterministic exploration algorithm ("LinUCB") with Policy_evaluator.
2. Define $\hat{p}$ and evaluate new policy.



Experimental Results II

Dataset = $35M$ (x, a, r) triples + $19M$ triples in test set. Ads a range over $880K$ choices.

Method	τ	Estimate	Interval
Learned	0.01	0.0193	[0.0187,0.0206]
Random	0.01	0.0154	[0.0149,0.0166]
Learned	0.05	0.0132	[0.0129,0.0137]
Random	0.05	0.0111	[0.0109,0.0116]
Naive	0.05	0.0	[0,0.0071]

Learned = optmizing $\hat{V}$ over ads with $\hat{p}(a|x) > 0$ using a linear regressor.
Random = choosing randomly amongst ads with $\hat{p}(a|x) > 0$.
Naive = supervised learning approach.

Double Robust Policy Evaluation

Basic question: Can we reduce the variance of a policy estimate? Suppose we have an estimate $\hat{r}(a, x)$, then we can form an estimator according to:

$$\frac{(r - \hat{r}(a, x))I(\pi(x) = a)}{p(a|x)} + \hat{r}(\pi(x), x)$$

Or even:

$$\frac{(r - \hat{r}(a, x))I(\pi(x) = a)}{\hat{p}(a|x)} + \hat{r}(\pi(x), x)$$

Theorem: If $\hat{p}(a|x) = p(a|x)$ or $\hat{r}(a, x) = E_{\vec{r} \sim D|x}[r_a]$ then $E\text{estimator} = \text{truth}$.
(And if $\hat{r}(a, x) = E_{\vec{r} \sim D|x}[r_a]$ it converges faster than the old estimator. Much faster when conditional variance is small.)

The Linear Setting

In the basic linear setting, assume:
$\exists w : \langle w, x_a \rangle = E_{(x,\vec{r})\sim D} r_a$ where x_a is a d dimension space.

[Auer 2002, Dani Hayes Kakade 2008, Lugosi & Cesa-Bianchi 2009] Theorem: For all true w with $||w|| < C$, with probability $1 - \delta$ the regret of the algorithm is

$$\tilde{O}(\mathrm{poly}(d)\sqrt{T})$$

Observation: realizable $\Rightarrow$ all updates useful $\Rightarrow$ weak K.

(But algorithms inconsistent when no perfect w exists.)
Another variant: Assume that all actions in a convex set are possible.

The Nearest Neighbor Setting

[Slivkins 09, Lu & Pal[2] 10]
Assume $\exists$ known $s(\cdot, \cdot)$ satisfying
$\forall x, a, x', a' : \; s((x, a), (x', a')) \geq |E[r_a|x] - E[r_{a'}|x']|$.
Theorem: For all problems, the online regret is bounded by

$$\tilde{O}(T^{1-1/(2+d_x+d_y)})$$

where d_x and d_y are measures of dimensionality.

Basic observation: You can first explore actions at course scale and then zoom in on the plausibly interesting areas.

Missing Pieces

1. How do we *efficiently* achieve the optimality of EXP4(P) given an oracle optimizer? Given an oracle that optimizes over policies, how can we use it to efficiently achieve EXP4(P) guarantees, at least in an IID setting? (New approach coming soon.)

Missing Pieces

1. How do we *efficiently* achieve the optimality of EXP4(P) given an oracle optimizer? Given an oracle that optimizes over policies, how can we use it to efficiently achieve EXP4(P) guarantees, at least in an IID setting? (New approach coming soon.)
2. How do we deal with delayed rewards? In the real world, reward information is not instantaneous. (Analysis coming soon.)

Missing Pieces

1. How do we *efficiently* achieve the optimality of EXP4(P) given an oracle optimizer? Given an oracle that optimizes over policies, how can we use it to efficiently achieve EXP4(P) guarantees, at least in an IID setting? (New approach coming soon.)
2. How do we deal with delayed rewards? In the real world, reward information is not instantaneous. (Analysis coming soon.)
3. How do we cope with priors over policies? Often we have some belief that one is better than another at the outset.

Missing Pieces

1. How do we *efficiently* achieve the optimality of EXP4(P) given an oracle optimizer? Given an oracle that optimizes over policies, how can we use it to efficiently achieve EXP4(P) guarantees, at least in an IID setting? (New approach coming soon.)
2. How do we deal with delayed rewards? In the real world, reward information is not instantaneous. (Analysis coming soon.)
3. How do we cope with priors over policies? Often we have some belief that one is better than another at the outset.

Missing Pieces

1. How do we *efficiently* achieve the optimality of EXP4(P) given an oracle optimizer? Given an oracle that optimizes over policies, how can we use it to efficiently achieve EXP4(P) guarantees, at least in an IID setting? (New approach coming soon.)
2. How do we deal with delayed rewards? In the real world, reward information is not instantaneous. (Analysis coming soon.)
3. How do we cope with priors over policies? Often we have some belief that one is better than another at the outset.

Some further discussion in posts at http://hunch.net

Bibliography (Presentation order)

[Epoch-Greedy] John Langford and Tong Zhang, The Epoch-Greedy Algorithm for Contextual Multi-armed Bandits NIPS 2007.

[EXP4] Peter Auer, Nicol Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The non-stochastic multi-armed bandit problem. SIAM Journal on Computing, 32(1):48-77, 2002.

[EXP4P] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, Robert E. Schapire, An Optimal High Probability Algorithm for the Contextual Bandit Problem, http://arxiv.org/abs/1002.4058

[Offset Tree] Alina Beygelzimer and John Langford, The Offset Tree for Learning with Partial Labels, KDD 2009.

Bibliography II

[Policy Deviations] Michael Kearns, Yishay Mansour, and Andrew Ng, Approximate planning in large POMDPs via reusable trajectories, NIPS 2000

[Progressive Validation] Avrim Blum, Adam Kalai, and John Langford, Beating the Holdout: Bounds for KFold and Progressive Cross-Validation, COLT99.

[Progressive Validation II] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile, On the generalization ability of on-line learning algorithms IEEE Transactions on Information Theory, 50(9):2050-2057, 2004.

[Progressive Validatin III] Nicolo Cesa-Bianchi and Claudio Gentile Improved risk tail bounds for on-line algorithms IEEE Transactions on Information Theory, 54(1)386-390, 2008.

Bibliography III

[Nonstationary Policy Evaluation] Lihong Li, Wei Chu, John Langford, Robert Schapire, A Contextual Bandit Approach to Personalized News Recommendation, WWW 2010.

[Exploration Scavenging] John Langford, Alexander Strehl, and Jennifer Wortman, Exploration Scavenging, ICML 2008.

[Implicit Exploration] Alex Strehl, John Langford, Sham Kakade, Lihong Li, Learning from Logged Implicit Exploration Data, http://arxiv.org/abs/1003.0120

[Double Robust I] Elad Hazan, Satyen Kale, Better Algorithms for Benign Bandits, SODA 2009.

[Double Robust II] David Chan, Rong Ge, Ori Gershony, Tim Hesterberg, Diane Lambert, Evaluating Online Ad Campaigns in a Pipeline: Causal Models at Scale, KDD 2010.

Bibliography IV

[Linear I] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. Journal of machine learning research, pages 397–422, 2002.

[Linear II] Varsha Dani, Thomas Hayes, and Sham Kakade, Stochastic Linear Optimization under Bandit Feedback, COLT 2008.

[Linear III] Nicolo Cesa-Bianchi and Gabor Lugosi, Combinatorial bandits, COLT 2009.

[Contextual Similarity I] Alex Slivkins. Contextual bandits with similarity information, Arxiv 2009.

[Contextual Similarity II] Tyler Lu, David Pal, Martin Pal, Contextual Multi-Armed Bandits, AIStats 2010.