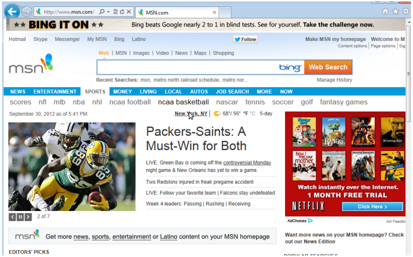
# Using Exploration

John Langford @ Microsoft Research (with help from many)

Large Scale Learning Class, April 9, 2013

(Post Presentation Version)

# Examples of Interactive Learning




Repeatedly:

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

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

# Another Example: Clinical Decision Making

"I stopped taking the medicine because I prefer the original disease to the side effects."

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 Bandit Setting

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

2. The learner chooses an action $a \in A$
3. The world reacts with reward $r_a \in [0, 1]$

Goal:

# The Bandit Setting

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

2. The learner chooses an action $a \in A$
3. The world reacts with reward $r_a \in [0, 1]$

Goal: Learn a good policy for choosing actions.

What does learning mean?

# The Bandit Setting

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

2. The learner chooses an action $a \in A$
3. The world reacts with reward $r_a \in [0, 1]$

Goal: Learn a good policy for choosing actions.

What does learning mean?
Competing with the set of actions $A$.

$$\text{Regret} = \max_{a' \in A} \text{average}_t (r_{a'} - r_a)$$

# The Contextual Bandit Setting

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

1. The world produces some context $x \in X$
2. The learner chooses an action $a \in A$
3. The world reacts with reward $r_a \in [0, 1]$

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

What does learning mean?
Competing a set of policies $\Pi = \{\pi : X \to A\}$:

$$\text{Regret} = \max_{\pi \in \Pi} \text{average}_t (r_{\pi(x)} - r_a)$$

# The Contextual Bandit Setting

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

1. The world produces some context $x \in X$
2. The learner chooses an action $a \in A$
3. The world reacts with reward $r_a \in [0, 1]$

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

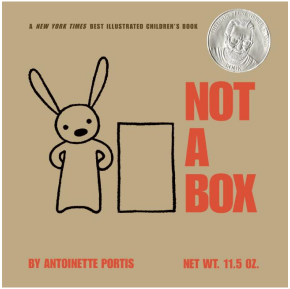
What does learning mean?
Competing a set of policies $\Pi = \{\pi : X \to A\}$:

$$\text{Regret} = \max_{\pi \in \Pi} \text{average}_t (r_{\pi(x)} - r_a)$$

Examples of $\Pi$:

- Context-free policies prescribing the same treatment to all.
- A machine learning system (e.g., all linear predictors)
- A discrete set based on domain-specific hunches or hypotheses

# 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.
- Many bandit algorithms can not be effectively applied.

# The Evaluation Problem

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

# The Evaluation Problem

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

Method 1: Deploy algorithm in the world.

Very Expensive!

# Method 2: The "Direct Method"

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

# Method 2: The "Direct Method"

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

One answer: Build a reward predictor $\hat{r}(x, a)$ from past data and evaluate on set of samples $x$.

$$\text{Value}(\pi) = \text{Average}\left(\hat{r}(x, \pi(x))\right)$$

# Method 2: The "Direct Method"

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

One answer: Build a reward predictor $\hat{r}(x, a)$ from past data and evaluate on set of samples $x$.

$$\text{Value}(\pi) = \text{Average}\left(\hat{r}(x, \pi(x))\right)$$

This can mislead badly. What if $\pi(x)$ always chooses actions which $\hat{r}(x, a)$ was not trained on? See Leon's Ad example in last lecture.

# Method 3: The Importance Weighting Trick

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

# Method 3: The Importance Weighting Trick

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

One answer: Collect $T$ exploration samples of the form

$$(x, a, r_a, p_a),$$

where

$x =$ context

$a =$ action

$r_a =$ reward for action

$p_a =$ probability of action $a$

then evaluate:

$$\text{Value}(\pi) = \text{ Average}\left(\frac{r_a \mathbf{1}(\pi(x) = a)}{p_a}\right)$$

# The Importance Weighting Trick

### Theorem

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

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

with deviations bounded by

$$O\left(\frac{1}{\sqrt{T \min_x p_{\pi(x)}}}\right)$$

Proof: [Part 1] $\mathbf{E}_{a\sim p}\left[\frac{r_a \mathbf{1}(\pi(x)=a)}{p_a}\right] = \sum_a p_a \frac{r_a \mathbf{1}(\pi(x)=a)}{p_a} = r_{\pi(x)}$

# What if you don't know probabilities?

Suppose $p$ was:

1. misrecorded "We randomized some actions, but then the Business Logic did something else."
2. not recorded "We randomized some scores which had an unclear impact on actions".
3. nonexistent "On Tuesday we did A and on Wednesday B".

Protip Leon: If you control the random process log the PRNG seed.

# What if you don't know probabilities?

Suppose $p$ was:

1. misrecorded "We randomized some actions, but then the Business Logic did something else."
2. not recorded "We randomized some scores which had an unclear impact on actions".
3. nonexistent "On Tuesday we did A and on Wednesday B".

Protip Leon: If you control the random process log the PRNG seed.

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

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

# What if you don't know probabilities?

Suppose $p$ was:

1. misrecorded "We randomized some actions, but then the Business Logic did something else."
2. not recorded "We randomized some scores which had an unclear impact on actions".
3. nonexistent "On Tuesday we did A and on Wednesday B".

Protip Leon: If you control the random process log the PRNG seed.

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

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

Theorem: For all IID $D$, for all policies $\pi$ with $p(a|x) > \tau$

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

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

# Can we do better?

Suppose we have a (possibly bad) reward estimator $\hat{r}(a, x)$. How can we use it?

# Can we do better?

Suppose we have a (possibly bad) reward estimator $\hat{r}(a, x)$. How can we use it?

$$\text{Value'}(\pi) = \text{Average}\left(\frac{(r_a - \hat{r}(a, x))\mathbf{1}(\pi(x) = a)}{p_a} + \hat{r}(\pi(x), x)\right)$$

# Can we do better?

Suppose we have a (possibly bad) reward estimator $\hat{r}(a, x)$. How can we use it?

$$\text{Value}'(\pi) = \text{Average}\left(\frac{(r_a - \hat{r}(a,x))\mathbf{1}(\pi(x) = a)}{p_a} + \hat{r}(\pi(x), x)\right)$$

Let $\Delta(a, x) = \hat{r}(a, x) - E_{\vec{r}|x} r_a$ = reward deviation
Let $\delta(a, x) = 1 - \frac{p_a}{\hat{p}_a}$ = probability deviation

**Theorem**

For all policies $\pi$ and all $(x, \vec{r})$:

$$|\text{Value}'(\pi) - E_{\vec{r}|x}[r_{\pi(x)}]| \leq |\Delta(\pi(x), x)\delta(\pi(x), x)|$$

The deviations multiply, so deviations $< 1$ means we win!

# How do you test things?

Contextual Bandit datasets tend to be highly proprietary. What can you do?

# How do you test things?

Contextual Bandit datasets tend to be highly proprietary. What can you do?

1. Pick classification dataset.
2. Generate $(x, a, r, p)$ quads via uniform random exploration of actions

# How do you test things?

Contextual Bandit datasets tend to be highly proprietary. What can you do?

1. Pick classification dataset.
2. Generate $(x, a, r, p)$ quads via uniform random exploration of actions

Apply transform to RCV1 dataset.
wget http://hunch.net/~jl/VW_raw.tar.gz
wget http://hunch.net/~jl/cbify.cc
Output format is:
action:cost:probability | features
Example:
1:1:0.5 | tuesday year million short compan vehicl line stat financ commit exchang plan corp subsid credit issu debt pay gold bureau prelimin refin billion telephon time draw basic relat file spokesm reut secur acquir form prospect period interview regist toront resourc barrick ontario qualif bln prospectus convertibl vinc borg arequip
...

# How do you train?

1. Learn $\hat{r}(a, x)$.
2. Compute for each $x$ the double-robust estimate for each $a' \in \{1, ..., K\}$:

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

3. Learn $\pi$ using a cost-sensitive classifier.

# How do you train?

1. Learn $\hat{r}(a, x)$.
2. Compute for each $x$ the double-robust estimate for each $a' \in \{1, ..., K\}$:

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

3. Learn $\pi$ using a cost-sensitive classifier.

vw –cb 2 –cb_type dr rcv1.train.txt.gz -c –ngram 2 –skips 4 -b 24 -l 0.25

Progressive 0/1 loss: 0.04582

vw –cb 2 –cb_type ips rcv1.train.txt.gz -c –ngram 2 –skips 4 -b 24 -l 0.125

Progressive 0/1 loss: 0.05065

vw –cb 2 –cb_type dm rcv1.train.txt.gz -c –ngram 2 –skips 4 -b 24 -l 0.125

Progressive 0/1 loss: 0.04679

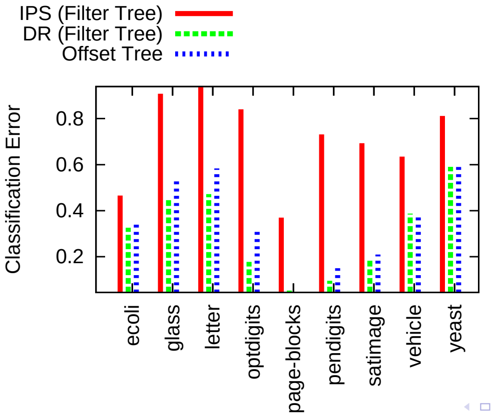
# Experimental Results

IPS = $\hat{r}(a,x) = 0$

DR = $\hat{r}(a,x) = w_a \cdot x$

Filter Tree = Cost Sensitive Multiclass classifier

Offset Tree = Earlier method for CB learning with same representation



# Summary of methods

1. Deployment. Aka A/B testing. Gold standard for measurement and cost.
2. Direct Method. Often used by people who don't know what they are doing. Some value when used in conjunction with careful exploration.
3. Inverse probability. Unbiased, but possibly high variance.
4. Inverse propensity score. For when you don't know or don't trust recorded probabilities. Debugging tool that gives hints, but caution is in order.
5. Double robust. Best known offline method. Unbiased + reduced variance.

# Bibliography: Exploration


Inverse An old technique, not sure where it was first used.

Nonrand J. Langford, A. Strehl, and J. Wortman Exploration Scavenging ICML 2008.

Offset A. Beygelzimer and J. Langford, The Offset Tree for Learning with Partial Labels KDD 2009.

Implicit A. Strehl, J. Langford, S. Kakade, and L. Li Learning from Logged Implicit Exploration Data NIPS 2010.

DRobust M. Dudik, J. Langford and L. Li, Doubly Robust Policy Evaluation and Learning, ICML 2011.