Learning to Interact

John Langford @ Microsoft Research (with help from many)

For demo:
Raw RCV1 CCAT-or-not:
http://hunch.net/~jl/VW_raw.tar.gz
Vowpal Wabbit for learning: http://hunch.net/~vw

Workshop on Log-based Personalization, February 28
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

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 $x \in X$
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.
The Evaluation Problem

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

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

Method 1: Deploy algorithm in the world.

Very Expensive!
The “Direct method”

Use past data to learn a reward predictor $\hat{r}(x, a)$, and act according to $\arg \max_a \hat{r}(x, a)$. 
The “Direct method”

Use past data to learn a reward predictor $\hat{r}(x, a)$, and act according to $\arg\max_a \hat{r}(x, a)$.

**Example:** Deployed policy always takes $a_1$ on $x_1$ and $a_2$ on $x_2$.

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The “Direct method”

Use past data to learn a reward predictor \( \hat{r}(x, a) \), and act according to \( \arg\max_a \hat{r}(x, a) \).

**Example:** Deployed policy always takes \( a_1 \) on \( x_1 \) and \( a_2 \) on \( x_2 \).

<table>
<thead>
<tr>
<th></th>
<th>( a_1 )</th>
<th>( a_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>.8</td>
<td>?</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>?</td>
<td>.2</td>
</tr>
</tbody>
</table>

Observed
The “Direct method”

Use past data to learn a reward predictor $\hat{r}(x, a)$, and act according to $\arg \max_a \hat{r}(x, a)$.

**Example:** Deployed policy always takes $a_1$ on $x_1$ and $a_2$ on $x_2$.

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>.8/.8</td>
<td>?/.5</td>
</tr>
<tr>
<td>$x_2$</td>
<td>?/.5</td>
<td>.2/.2</td>
</tr>
</tbody>
</table>
The “Direct method”

Use past data to learn a reward predictor $\hat{r}(x, a)$, and act according to $\arg \max_a \hat{r}(x, a)$.

**Example:** Deployed policy always takes $a_1$ on $x_1$ and $a_2$ on $x_2$.

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>.8/.8</td>
<td>?/.5</td>
</tr>
<tr>
<td>$x_2$</td>
<td>.3/.5</td>
<td>.2/.2</td>
</tr>
</tbody>
</table>
The “Direct method”

Use past data to learn a reward predictor $\hat{r}(x, a)$, and act according to $\arg\max_a \hat{r}(x, a)$.

**Example:** Deployed policy always takes $a_1$ on $x_1$ and $a_2$ on $x_2$.

<table>
<thead>
<tr>
<th>Observed/Estimated</th>
<th>$a_1$</th>
<th>$a_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.8/0.8</td>
<td>?/0.514</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.3/0.3</td>
<td>0.2/0.014</td>
</tr>
</tbody>
</table>
The “Direct method”

Use past data to learn a reward predictor \( \hat{r}(x, a) \), and act according to \( \arg \max_a \hat{r}(x, a) \).

**Example:** Deployed policy always takes \( a_1 \) on \( x_1 \) and \( a_2 \) on \( x_2 \).

<table>
<thead>
<tr>
<th>Observed/Estimated/True</th>
<th>( a_1 )</th>
<th>( a_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>.8/.8/.8</td>
<td>?/.514/1</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>.3/.3/.3</td>
<td>.2/.014/.2</td>
</tr>
</tbody>
</table>
Use past data to learn a reward predictor \( \hat{r}(x, a) \), and act according to \( \text{arg max}_a \hat{r}(x, a) \).

Example: Deployed policy always takes \( a_1 \) on \( x_1 \) and \( a_2 \) on \( x_2 \).

<table>
<thead>
<tr>
<th></th>
<th>Observed/Estimated/True</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>.8/.8/.8</td>
<td>( ?/.514/1 )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>.3/.3/.3</td>
<td>.2/.014/.2</td>
</tr>
</tbody>
</table>

Basic observation 1: Generalization alone is not sufficient.
Use past data to learn a reward predictor $\hat{r}(x,a)$, and act according to $\arg \max_a \hat{r}(x,a)$.

Example: Deployed policy always takes $a_1$ on $x_1$ and $a_2$ on $x_2$.

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>.8/.8/.8</td>
<td>?/.514/1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>.3/.3/.3</td>
<td>.2/.014/.2</td>
</tr>
</tbody>
</table>

Basic observation 2: Exploration is required to succeed.
The “Direct method”

Use past data to learn a reward predictor \( \hat{r}(x, a) \), and act according to \( \arg \max_a \hat{r}(x, a) \).

Example: Deployed policy always takes \( a_1 \) on \( x_1 \) and \( a_2 \) on \( x_2 \).

<table>
<thead>
<tr>
<th>Observed/Estimated/True</th>
<th>( a_1 )</th>
<th>( a_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>.8/.8/.8</td>
<td>(?/.514/1</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>.3/.3/.3</td>
<td>.2/.014/.2</td>
</tr>
</tbody>
</table>

Basic observation 3: Prediction errors not controlled exploration.
Outline

1 Using Exploration
   1 Problem Definition
   2 Direct Method fails
   3 Importance Weighting
   4 Doubly Robust

2 Doing Exploration
Method 3: The Importance Weighting Trick

Let $\pi : X \rightarrow A$ be a policy mapping features to actions. How do we evaluate it?
Method 3: The Importance Weighting Trick

Let $\pi : X \rightarrow A$ 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:

$$\text{Value}(\pi) = \text{Average} \left( \frac{r_a 1(\pi(x) = a)}{p_a} \right)$$
The Importance Weighting Trick

Theorem

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

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

Proof:

$$E_{a \sim p} \left[ \frac{r_a 1(\pi(x) = a)}{p_a} \right] = \sum_a p_a \frac{r_a 1(\pi(x) = a)}{p_a} = r_{\pi}(x)$$

Example:

<table>
<thead>
<tr>
<th>Action</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>Probability</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{3}{4}$</td>
</tr>
<tr>
<td>Estimate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Importance Weighting Trick

Theorem

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

\[
E_{(x, \tilde{r}) \sim D} [r_{\pi(x)}] = \mathbf{E}[\text{Value}(\pi)]
\]

Proof: \( E_{a \sim p} \left[ \frac{r_a 1(\pi(x) = a)}{p_a} \right] = \sum_a p_a \frac{r_a 1(\pi(x) = a)}{p_a} = r_{\pi(x)} \)

Example:

<table>
<thead>
<tr>
<th>Action</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>Probability</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{3}{4} )</td>
</tr>
<tr>
<td>Estimate</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
The Importance Weighting Trick

**Theorem**

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

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

**Proof:**

$$E_{a \sim p}\left[\frac{r_a \mathbf{1}(\pi(x) = a)}{p_a}\right] = \sum_a p_a \frac{r_a \mathbf{1}(\pi(x) = a)}{p_a} = r_{\pi}(x)$$

**Example:**

<table>
<thead>
<tr>
<th>Action</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>Probability</td>
<td>1/4</td>
<td>3/4</td>
</tr>
<tr>
<td>Estimate</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
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?

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

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

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

Why does this work?
Can we do better?

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

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

Why does this work?

\[
E_{a \sim p} \left( \frac{\hat{r}(a, x)1(\pi(x) = a)}{p_a} \right) = \hat{r}(\pi(x), x)
\]

so

\[
0 = E_{a \sim p} \left( \frac{-\hat{r}(a, x)1(\pi(x) = a)}{p_a} + \hat{r}(\pi(x), x) \right)
\]
Can we do better?

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

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

Why does this work?

\[
\mathbb{E}_{a \sim p} \left( \frac{\hat{r}(a, x)1(\pi(x) = a)}{p_a} \right) = \hat{r}(\pi(x), x)
\]

so

\[
0 = \mathbb{E}_{a \sim p} \left( \frac{-\hat{r}(a, x)1(\pi(x) = a)}{p_a} + \hat{r}(\pi(x), x) \right)
\]

So it’s “safe”. It helps, because $r_a - \hat{r}(a, x)$ small reduces variance.
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.
3. 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 ear million short company vehicle plan corporation subsid credit issue debt pay gold bureau preliminary billion telephone time draw basic related speaking esem reuter... 
   ```
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.

```bash
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 ear million short company vehicle statistic exchange plan corporate subsid credit issuance pay gold bureau preliminary billion telephone time draw basic relation speech spok esm reuters secur acquire form prospect period interview register to front resource barricade ontario qualify billion prospect convertible bond approach requirement ...
```
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:

\[ \text{action:cost:probability | features} \]

Example:

\[ 1:1:0.5 \mid \text{tuesday year million short compan vehicl line stat financ commit exchang plan corp subsid credit issu debt pay gold bureau prelimin refin billion telephon time draw basic relat file spokesm reut secur acquir form prospect period interview regist toront resourc barrick ontario qualif bln prospectus convertibl vinc borg arequip} \]
How do you train?

1. Learn $\hat{r}(a, x)$.

2. Compute for each $x$ the double-robust estimate for each $a' \in \{1, \ldots, K\}$:

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

3. Learn $\pi$ using a cost-sensitive classifier. We’ll use Vowpal Wabbit: [http://hunch.net/~vw](http://hunch.net/~vw)
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. We’ll use Vowpal Wabbit: [http://hunch.net/~vw](http://hunch.net/~vw)

   - `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
Outline

1 Using Exploration
   1 Problem Definition
   2 Direct Method fails
   3 Importance Weighting
   4 Doubly Robust

2 Doing Exploration
   1 Exploration First
   2 $\epsilon$-Greedy
   3 Thompson Sampling
   4 Covering
Reminder: 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? Efficiently competing with some large reference class of policies $\Pi = \{\pi : X \rightarrow A\}$:

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

Let $Q_1 =$ uniform distribution
For $t = 1, \ldots, T$: 

1. The world produces some context $x \in X$
2. Sample $a \in A$ from $Q_t$
3. The world reacts with reward $r_a \in [0, 1]$
4. Let $Q_{t+1}$ cover the good subset of policies.
A Covering Algorithm

Let $Q_1 =$ uniform distribution
For $t = 1, \ldots, T$:

1. The world produces some context $x \in X$
2. Sample $a \in A$ from $Q_t$
3. The world reacts with reward $r_a \in [0, 1]$
4. Let $Q_{t+1}$ cover the good subset of policies.

What is a good cover?
A Covering Algorithm

Let $Q_1 = \text{uniform distribution}$
For $t = 1, \ldots, T$:

1. The world produces some context $x \in X$
2. Sample $a \in A$ from $Q_t$
3. The world reacts with reward $r_a \in [0, 1]$
4. Let $Q_{t+1}$ cover the good subset of policies.

What is a good cover?

Let $\mu_t = \sqrt{\frac{\ln |\Pi|}{t|A|}}$

A distribution over policies $\pi$ satisfying

$$\forall \pi \in \Pi : \mathbb{E}_x \left[ \frac{1}{Q_t(\pi(x)|x)} \right] \leq \frac{\text{Empirical regret of } \pi}{\mu_t}$$
Theorem: Optimal in all ways

Regret: $\tilde{O}\left(\sqrt{\frac{|A| \ln |\Pi|}{T}}\right)$
Theorem: Optimal in all ways

Regret: $\tilde{O}\left(\sqrt{\frac{|A| \ln |\Pi|}{T}}\right)$

Calls to Cost sensitive classification oracle: $\tilde{O}\left(T^{0.5}\right)$

Lower bound: $\Omega\left(T^{0.5}\right)$ calls to oracle
Theorem: Optimal in all ways

Regret: \( \tilde{O} \left( \sqrt{\frac{|A| \ln |\Pi|}{T}} \right) \)

Calls to Cost sensitive classification oracle: \( \tilde{O} \left( T^{0.5} \right) \)

Lower bound: \( \Omega \left( T^{0.5} \right) \) calls to oracle

Running time: \( \tilde{O} \left( T^{1.5} \right) \)
How well does this work?

losses on CCAT RCV1 problem

- eps-greedy
- tau-first
- LinUCB*
- Cover
How long does this take?

Running times on CCAT RCV1 problem

- eps-greedy
- tau-first
- LinUCB*
- Cover

- seconds

- 1e+06
- 100000
- 10000
- 1000
- 100
- 10
- 1
Also see NIPS tutorial or NYU large scale learning class for much more detail.

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