Each year Microsoft Research hosts hundreds of influential speakers from around the world including leading scientists, renowned experts in technology, book authors, and leading academics, and makes videos of these lectures freely available.

2013 © Microsoft Corporation. All rights reserved.
Learning to Interact

John Langford @ Microsoft Research (with help from many)

Slides at: http://hunch.net/~jl/interact.pdf

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

“Whoa—way too much information.”
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.
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.

"Whoa—way too much information."
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.
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!
Use past data to learn a reward predictor \( \hat{r}(x, a) \), and act according to \( \arg \max_a \hat{r}(x, a) \).
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.
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>
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>.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></th>
<th>$a_1$</th>
<th>$a_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.8/0.8/0.8</td>
<td>?/0.514/1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.3/0.3/0.3</td>
<td>0.2/0.014/0.2</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/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 1: Generalization alone is not sufficient.
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/.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. Missing Probabilities
   5. 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:

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

$x = \text{context}$

$a = \text{action}$

$r_a = \text{reward for action}$

$p_a = \text{probability of action } a$

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 $\pi$:

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

with deviations bounded by

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

Proof: [Part 1] $E_{a \sim p} \left[ \frac{r_{a}1(\pi(x)=a)}{p_{a}} \right] = \sum_{a} p_{a} \frac{r_{a}1(\pi(x)=a)}{p_{a}} = r_{\pi}(x)$
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 $\pi$:

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

with deviations bounded by

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

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

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”.

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

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

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

Define new estimator: $\hat{V}(\pi) = \hat{E}_{x, a, r_a} \left[ \frac{r_a I(\pi(x)=a)}{\max\{\tau, \hat{p}(a|x)\}} \right]$ where $\tau = 0$ is a 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 $\text{reg}(\hat{p}) = E_{x \sim D, a \sim p(a|x)}[(p(a|x) - \hat{p}(a|x))^2] = \text{squared loss regret.}$
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”.

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

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

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

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

**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 $\text{reg}(\hat{p}) = E_{x \sim D, a \sim p(a|x)}[(p(a|x) - \hat{p}(a|x))^2] = \text{squared loss regret}$.
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.”
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:

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

**Theorem**

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

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

with deviations bounded by

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

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

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

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

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 $\text{reg}(\hat{p}) = E_{x \sim D, a \sim p(a|x)}[(p(a|x) - \hat{p}(a|x))^2] = \text{squared loss regret}$.
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?

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

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

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

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

Theorem

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

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

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.

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 acquire 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 $a' \in \{1, \ldots, K\}$:

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

3. Learn $\pi$ using a cost-sensitive classifier. We’ll use Vowpal Wabbit: [http://hunch.net/~vw](http://hunch.net/~vw)
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 acquire form prospect period interview regist toront resourc barrick ontario qualif bln prospectus convertibil vinc borg arequip ...
```
1. Learn $\hat{r}(a, x)$.

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

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

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

1. Learn \( \hat{r}(a, x) \).
2. Compute for each \( x \) the double-robust estimate for each \( a' \in \{1, \ldots, K\} \):
   \[
   \frac{(r - \hat{r}(a, x))1(a' = a)}{p(a|x)} + \hat{r}(a', x)
   \]
3. Learn \( \pi \) using a cost-sensitive classifier. We’ll use Vowpal Wabbit:
   
   \texttt{http://hunch.net/~vw} 

   \texttt{vw -cb 2 -cb_type dr rcv1.train.txt.gz -c -ngram 2 -skips 4 -b 24 -l 0.25}  
   \text{Progressive 0/1 loss: 0.04582} 

   \texttt{vw -cb 2 -cb_type ips rcv1.train.txt.gz -c -ngram 2 -skips 4 -b 24 -l 0.125}  
   \text{Progressive 0/1 loss: 0.05065} 

   \texttt{vw -cb 2 -cb_type dm rcv1.train.txt.gz -c -ngram 2 -skips 4 -b 24 -l 0.125}  
   \text{Progressive 0/1 loss: 0.04679}
```
initial_t = 0
power_t = 0.5
using cache_file = rcv1_train.cb_vw.cache
ignoring text input in favor of cache input
num sources = 1

<table>
<thead>
<tr>
<th>average</th>
<th>since</th>
<th>example</th>
<th>example</th>
<th>current</th>
<th>current</th>
<th>current</th>
<th>avglossreg</th>
<th>last pred</th>
<th>last correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>loss</td>
<td>last</td>
<td>counter</td>
<td>weight</td>
<td>label</td>
<td>predict</td>
<td>features</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0.666667 0.666667 3 3.0 known 2 316 0.334247 0.041716 0.000000
0.333333 0.000000 6 6.0 known 2 160 0.328435 0.016708 1.000000
0.365390 0.403858 11 11.0 known 2 202 0.354719 0.040916 0.000000
0.363327 0.361265 22 22.0 known 2 502 0.344410 0.049526 0.000000
0.370952 0.378576 44 44.0 known 2 370 0.405983 0.078159 0.000000
0.288965 0.205072 87 87.0 known 1 340 0.356304 0.100344 1.000000
0.293865 0.298764 174 174.0 known 2 130 0.322963 0.083125 0.000000
0.198690 0.103516 348 348.0 known 2 262 0.297750 0.357253 1.000000
0.158162 0.117633 696 696.0 known 2 124 0.249183 0.082325 0.000000
0.123245 0.088328 1392 1392.0 known 2 1066 0.215804 0.583740 0.000000
0.111740 0.100234 2784 2784.0 known 1 280 0.176151 0.247207 1.000000
0.092496 0.073252 5568 5568.0 known 1 514 0.143719 0.203254 0.000000
0.082852 0.073207 11135 11135.0 known 2 352 0.121448 1.058181 1.000000
0.072335 0.061816 22269 22269.0 known 2 820 0.101361 0.076899 0.000000
0.064118 0.055902 44537 44537.0 known 2 226 0.086304 -0.138273 0.000000
0.059023 0.053927 89073 89073.0 known 1 142 0.074598 1.061901 1.000000
0.054813 0.050603 178146 178146.0 known 2 274 0.065937 1.007291 1.000000
0.050256 0.045699 356291 356291.0 known 1 580 0.059258 1.076878 1.000000
0.046211 0.042166 712582 712582.0 known 1 394 0.053942 0.008066 0.000000

finished run
number of examples = 781265
weighted example sum = 7.813e+05
weighted label sum = 0
average loss = 0.04582
best constant = 0
total feature number = 343993166
```
Generating 2-grams for all namespaces.
Generating 4-skips for all namespaces.
Num weight bits = 24
learning rate = 0.25
initial_t = 0
power_t = 0.5
using cache_file = rcv1_train.cb_vw.cache
ignoring text input in favor of cache input
num sources = 1

<table>
<thead>
<tr>
<th>average</th>
<th>since</th>
<th>example</th>
<th>example</th>
<th>current</th>
<th>current</th>
<th>current</th>
<th>current</th>
<th>predict</th>
<th>features</th>
<th>avglossreg</th>
<th>last_pred</th>
<th>last correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.666667</td>
<td>0.666667</td>
<td>3</td>
<td>3.0</td>
<td>known</td>
<td>2</td>
<td>316</td>
<td>0.334247</td>
<td>0.041716</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.333333</td>
<td>0.000000</td>
<td>6</td>
<td>6.0</td>
<td>known</td>
<td>2</td>
<td>160</td>
<td>0.328435</td>
<td>0.016708</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.365390</td>
<td>0.403858</td>
<td>11</td>
<td>11.0</td>
<td>known</td>
<td>2</td>
<td>202</td>
<td>0.354719</td>
<td>0.049916</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.363327</td>
<td>0.361265</td>
<td>22</td>
<td>22.0</td>
<td>known</td>
<td>2</td>
<td>502</td>
<td>0.344410</td>
<td>0.049526</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.370952</td>
<td>0.378576</td>
<td>44</td>
<td>44.0</td>
<td>known</td>
<td>2</td>
<td>370</td>
<td>0.405983</td>
<td>0.078159</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.288965</td>
<td>0.205072</td>
<td>87</td>
<td>87.0</td>
<td>known</td>
<td>1</td>
<td>340</td>
<td>0.356304</td>
<td>0.100344</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.293865</td>
<td>0.298764</td>
<td>174</td>
<td>174.0</td>
<td>known</td>
<td>2</td>
<td>130</td>
<td>0.322963</td>
<td>0.083125</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.198690</td>
<td>0.103516</td>
<td>348</td>
<td>348.0</td>
<td>known</td>
<td>2</td>
<td>262</td>
<td>0.297750</td>
<td>0.357253</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.158162</td>
<td>0.117633</td>
<td>696</td>
<td>696.0</td>
<td>known</td>
<td>2</td>
<td>124</td>
<td>0.249183</td>
<td>0.082325</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.123245</td>
<td>0.088328</td>
<td>1392</td>
<td>1392.0</td>
<td>known</td>
<td>2</td>
<td>1066</td>
<td>0.215804</td>
<td>0.583740</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.111740</td>
<td>0.100234</td>
<td>2784</td>
<td>2784.0</td>
<td>known</td>
<td>1</td>
<td>280</td>
<td>0.176151</td>
<td>0.247207</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
6:15PM 1-of-3-7: ls -l rcv1_train.cb_vw  
-rw-r--r-- 1 jl jl 375458083 Dec 5 13:03 rcv1_train.cb_vw
7:06PM 1-of-3-8: less rcv1_train.cb_vw  
~/.presentations/nips_2013 [jl/ttypts/0]
7:07PM 1-of-3-9: vw --cb 2 --cb_type dr --ngram 2 --skips 4 -b 24 -l 0.25 rcv1_train.cb_vw -c
Generating 2-grams for all namespaces.
Generating 4-skips for all namespaces.
Num weight bits = 24
learning rate = 0.25
initial_t = 0
power_t = 0.5
using cache_file = rcv1_train.cb_vw.cache
ignoring text input in favor of cache input
num sources = 1

<table>
<thead>
<tr>
<th>average</th>
<th>since</th>
<th>example</th>
<th>example</th>
<th>current</th>
<th>current</th>
<th>current</th>
<th>current</th>
<th>avglossreg</th>
<th>last pred</th>
<th>last correc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.666667</td>
<td>0.666667</td>
<td>3</td>
<td>3.0</td>
<td>known</td>
<td>2</td>
<td>316</td>
<td>0.334247</td>
<td>0.041716</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>0.333333</td>
<td>0.000000</td>
<td>6</td>
<td>6.0</td>
<td>known</td>
<td>2</td>
<td>160</td>
<td>0.328435</td>
<td>0.016708</td>
<td>1.000000</td>
<td></td>
</tr>
<tr>
<td>0.365390</td>
<td>0.403858</td>
<td>11</td>
<td>11.0</td>
<td>known</td>
<td>2</td>
<td>202</td>
<td>0.354719</td>
<td>0.049016</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>0.363327</td>
<td>0.361265</td>
<td>22</td>
<td>22.0</td>
<td>known</td>
<td>2</td>
<td>502</td>
<td>0.344410</td>
<td>0.049526</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>0.370952</td>
<td>0.378576</td>
<td>44</td>
<td>44.0</td>
<td>known</td>
<td>2</td>
<td>370</td>
<td>0.405983</td>
<td>0.078159</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>0.288965</td>
<td>0.205072</td>
<td>87</td>
<td>87.0</td>
<td>known</td>
<td>1</td>
<td>340</td>
<td>0.356304</td>
<td>0.100344</td>
<td>1.000000</td>
<td></td>
</tr>
<tr>
<td>0.293865</td>
<td>0.298764</td>
<td>174</td>
<td>174.0</td>
<td>known</td>
<td>2</td>
<td>130</td>
<td>0.322963</td>
<td>0.083125</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>0.198690</td>
<td>0.135183</td>
<td>348</td>
<td>348.0</td>
<td>known</td>
<td>2</td>
<td>262</td>
<td>0.297750</td>
<td>0.357253</td>
<td>1.000000</td>
<td></td>
</tr>
<tr>
<td>0.158162</td>
<td>0.117633</td>
<td>696</td>
<td>696.0</td>
<td>known</td>
<td>2</td>
<td>124</td>
<td>0.249183</td>
<td>0.082325</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>0.123245</td>
<td>0.088328</td>
<td>1392</td>
<td>1392.0</td>
<td>known</td>
<td>2</td>
<td>1066</td>
<td>0.215804</td>
<td>0.583740</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>0.111740</td>
<td>0.100234</td>
<td>2784</td>
<td>2784.0</td>
<td>known</td>
<td>1</td>
<td>280</td>
<td>0.176151</td>
<td>0.247207</td>
<td>1.000000</td>
<td></td>
</tr>
<tr>
<td>0.092496</td>
<td>0.073252</td>
<td>5568</td>
<td>5568.0</td>
<td>known</td>
<td>1</td>
<td>514</td>
<td>0.143719</td>
<td>0.203254</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>0.082852</td>
<td>0.073207</td>
<td>11135</td>
<td>11135.0</td>
<td>known</td>
<td>2</td>
<td>352</td>
<td>0.121448</td>
<td>1.058181</td>
<td>1.000000</td>
<td></td>
</tr>
<tr>
<td>0.072335</td>
<td>0.061816</td>
<td>22269</td>
<td>22269.0</td>
<td>known</td>
<td>2</td>
<td>820</td>
<td>0.101361</td>
<td>0.076899</td>
<td>0.000000</td>
<td></td>
</tr>
</tbody>
</table>
```
6:15PM 1-of-3: ls -l rcv1_train.cb_vw
7:06PM 1-of-3: less rcv1_train.cb_vw
7:07PM 1-of-3: vw --cb 2 --cb_type dr --ngram 2 --skips 4 -b 24 -l 0.25 rcv1_train.cb_vw -c
Generating 2-grams for all namespaces.
Generating 4-skips for all namespaces.
Num weight bits = 24
learning rate = 0.25
initial_t = 0
power_t = 0.5
using cache_file = rcv1_train.cb_vw.cache
ignoring text input in favor of cache input
num sources = 1
<table>
<thead>
<tr>
<th>average</th>
<th>since</th>
<th>last</th>
<th>example</th>
<th>counter</th>
<th>example</th>
<th>current</th>
<th>weight</th>
<th>label</th>
<th>current</th>
<th>predict</th>
<th>features</th>
<th>avglossreg</th>
<th>last pred</th>
<th>last correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.666667</td>
<td>0.666667</td>
<td>3</td>
<td>3.0</td>
<td>known</td>
<td>2</td>
<td>316</td>
<td>0.334247</td>
<td>0.041716</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.333333</td>
<td>0.000000</td>
<td>6</td>
<td>6.0</td>
<td>known</td>
<td>2</td>
<td>160</td>
<td>0.328435</td>
<td>0.016708</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.365390</td>
<td>0.403858</td>
<td>11</td>
<td>11.0</td>
<td>known</td>
<td>2</td>
<td>202</td>
<td>0.354719</td>
<td>0.040916</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.363327</td>
<td>0.361265</td>
<td>22</td>
<td>22.0</td>
<td>known</td>
<td>2</td>
<td>502</td>
<td>0.344410</td>
<td>0.049526</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.370952</td>
<td>0.378576</td>
<td>44</td>
<td>44.0</td>
<td>known</td>
<td>2</td>
<td>370</td>
<td>0.405983</td>
<td>0.078159</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.288965</td>
<td>0.205072</td>
<td>87</td>
<td>87.0</td>
<td>known</td>
<td>1</td>
<td>340</td>
<td>0.356304</td>
<td>0.100344</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.293865</td>
<td>0.298764</td>
<td>174</td>
<td>174.0</td>
<td>known</td>
<td>2</td>
<td>130</td>
<td>0.322963</td>
<td>0.083125</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.198690</td>
<td>0.103516</td>
<td>348</td>
<td>348.0</td>
<td>known</td>
<td>2</td>
<td>262</td>
<td>0.297750</td>
<td>0.357253</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.158162</td>
<td>0.117633</td>
<td>696</td>
<td>696.0</td>
<td>known</td>
<td>2</td>
<td>124</td>
<td>0.249183</td>
<td>0.082325</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.123245</td>
<td>0.088328</td>
<td>1392</td>
<td>1392.0</td>
<td>known</td>
<td>2</td>
<td>1066</td>
<td>0.215804</td>
<td>0.583740</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.111740</td>
<td>0.100234</td>
<td>2784</td>
<td>2784.0</td>
<td>known</td>
<td>1</td>
<td>280</td>
<td>0.176151</td>
<td>0.247207</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.092246</td>
<td>0.073252</td>
<td>5568</td>
<td>5568.0</td>
<td>known</td>
<td>1</td>
<td>514</td>
<td>0.143719</td>
<td>0.203254</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.082852</td>
<td>0.073207</td>
<td>11135</td>
<td>11135.0</td>
<td>known</td>
<td>2</td>
<td>352</td>
<td>0.121448</td>
<td>1.058181</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.072335</td>
<td>0.061816</td>
<td>22269</td>
<td>22269.0</td>
<td>known</td>
<td>2</td>
<td>820</td>
<td>0.101361</td>
<td>0.076899</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.064118</td>
<td>0.055902</td>
<td>44537</td>
<td>44537.0</td>
<td>known</td>
<td>2</td>
<td>226</td>
<td>0.086304</td>
<td>-0.138273</td>
<td>0.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.059023</td>
<td>0.053927</td>
<td>89073</td>
<td>89073.0</td>
<td>known</td>
<td>1</td>
<td>142</td>
<td>0.074598</td>
<td>1.061901</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.054813</td>
<td>0.050603</td>
<td>178146</td>
<td>178146.0</td>
<td>known</td>
<td>2</td>
<td>274</td>
<td>0.065937</td>
<td>1.007291</td>
<td>1.000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
initial_t = 0
power_t = 0.5
using cache_file = rcv1_train.cb_vw.cache
ignoring text input in favor of cache input
num sources = 1
average since example example current current current
t loss last counter weight label predict features avglossreg last pred last correct
0.666667 0.666667 3 3.0 known 2 316 0.334247 0.041716 0.000000
0.333333 0.000000 6 6.0 known 2 160 0.328435 0.016708 1.000000
0.365390 0.403858 11 11.0 known 2 202 0.354719 0.040916 0.000000
0.363327 0.361265 22 22.0 known 2 502 0.344410 0.049526 0.000000
0.370952 0.378576 44 44.0 known 2 370 0.405983 0.078159 0.000000
0.289965 0.205072 87 87.0 known 1 340 0.356304 0.100344 1.000000
0.293865 0.298764 174 174.0 known 2 130 0.322963 0.083125 0.000000
0.198690 0.103516 348 348.0 known 2 262 0.297750 0.357253 1.000000
0.158162 0.117633 696 696.0 known 2 124 0.249183 0.082325 0.000000
0.123245 0.088328 1392 1392.0 known 2 1066 0.215804 0.583740 0.000000
0.111740 0.100234 2784 2784.0 known 1 280 0.176151 0.247207 1.000000
0.092496 0.073252 5568 5568.0 known 1 514 0.143719 0.203254 0.000000
0.082852 0.073207 11135 11135.0 known 2 352 0.121448 1.058181 1.000000
0.072335 0.061816 22269 22269.0 known 2 820 0.101361 0.076899 0.000000
0.064118 0.055902 44537 44537.0 known 2 226 0.086304 0.138273 0.000000
0.059023 0.053927 89073 89073.0 known 1 142 0.074598 1.061901 1.000000
0.054813 0.050603 178146 178146.0 known 2 274 0.065937 1.007291 1.000000
0.050256 0.045699 356291 356291.0 known 1 580 0.059258 1.076878 1.000000
0.046211 0.042166 712582 712582.0 known 1 394 0.053942 0.008066 0.000000

finished run
number of examples = 781265
weighted example sum = 7.813e+05
weighted label sum = 0
average loss = 0.04582
best constant = 0
total feature number = 343993166
```
finished run
number of examples = 781265
weighted example sum = 7.813e+05
weighted label sum = 0
average loss = 0.04582
best constant = 0
total feature number = 343993166
7:09PM 1-of-3-10: vw --cb 2 --cb_type ips --ngram 2 --skips 4 -b 24 -l 0.25 rcv1_train.cb_vw.c
Generating 2-grams for all namespaces.
Generating 4-skips for all namespaces.
Num weight bits = 24
learning rate = 0.25
initial_t = 0
power_t = 0.5
using cache_file = rcv1_train.cb_vw.cache
ignoring text input in favor of cache input
num sources = 1

<table>
<thead>
<tr>
<th>average loss</th>
<th>since</th>
<th>example counter</th>
<th>example weight</th>
<th>current label</th>
<th>current predict</th>
<th>current features</th>
<th>avglossreg</th>
<th>last pred</th>
<th>last correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.666667</td>
<td>0.666667</td>
<td>3</td>
<td>3.0</td>
<td>known</td>
<td>2</td>
<td>316</td>
<td>0.3333333</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.333333</td>
<td>0.000000</td>
<td>6</td>
<td>6.0</td>
<td>known</td>
<td>2</td>
<td>160</td>
<td>0.3333333</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.363636</td>
<td>0.400000</td>
<td>11</td>
<td>11.0</td>
<td>known</td>
<td>2</td>
<td>202</td>
<td>0.363636</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.454545</td>
<td>0.545455</td>
<td>22</td>
<td>22.0</td>
<td>known</td>
<td>2</td>
<td>502</td>
<td>0.363636</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.363636</td>
<td>0.272727</td>
<td>44</td>
<td>44.0</td>
<td>known</td>
<td>2</td>
<td>370</td>
<td>0.477277</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.275862</td>
<td>0.186047</td>
<td>87</td>
<td>87.0</td>
<td>known</td>
<td>1</td>
<td>340</td>
<td>0.471264</td>
<td>0.000000</td>
<td>1.000000</td>
</tr>
<tr>
<td>0.298851</td>
<td>0.321839</td>
<td>174</td>
<td>174.0</td>
<td>known</td>
<td>2</td>
<td>130</td>
<td>0.459770</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.212644</td>
<td>0.126437</td>
<td>348</td>
<td>348.0</td>
<td>known</td>
<td>2</td>
<td>262</td>
<td>0.465517</td>
<td>0.000000</td>
<td>1.000000</td>
</tr>
<tr>
<td>0.186782</td>
<td>0.160920</td>
<td>696</td>
<td>696.0</td>
<td>known</td>
<td>2</td>
<td>124</td>
<td>0.472701</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.150862</td>
<td>0.114943</td>
<td>1392</td>
<td>1392.0</td>
<td>known</td>
<td>1</td>
<td>1066</td>
<td>0.481322</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.128592</td>
<td>0.106322</td>
<td>2784</td>
<td>2784.0</td>
<td>known</td>
<td>1</td>
<td>280</td>
<td>0.497845</td>
<td>0.000000</td>
<td>1.000000</td>
</tr>
<tr>
<td>0.108836</td>
<td>0.089080</td>
<td>5568</td>
<td>5568.0</td>
<td>known</td>
<td>1</td>
<td>514</td>
<td>0.497306</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.094836</td>
<td>0.080833</td>
<td>11135</td>
<td>11135.0</td>
<td>known</td>
<td>2</td>
<td>352</td>
<td>0.502829</td>
<td>0.000000</td>
<td>1.000000</td>
</tr>
</tbody>
</table>
weighted label sum = 0
average loss = 0.04582
best constant = 0

total feature number = 343993166

7:09PM 1-of-3:10: vw --cb 2 --cb_type ips --ngram 2 --skips 4 -b 24 -l 0.25 rcv1_train.cb_vw -c
Generating 2-grams for all namespaces.
Generating 4-skips for all namespaces.
Num weight bits = 24
learning rate = 0.25
initial_t = 0
power_t = 0.5
using cache_file = rcv1_train.cb_vw.cache
ignoring text input in favor of cache input
num sources = 1

<table>
<thead>
<tr>
<th>平均损失</th>
<th>估计</th>
<th>最近</th>
<th>例子</th>
<th>例子</th>
<th>当前</th>
<th>当前</th>
<th>当前</th>
<th>预测</th>
<th>特征</th>
<th>平均损失</th>
<th>最后预测</th>
<th>最后正确</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.666667</td>
<td>0.666667</td>
<td>3</td>
<td>3.0</td>
<td>known</td>
<td>2</td>
<td>316</td>
<td>0.333333</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.666667</td>
<td>0.666667</td>
<td></td>
</tr>
<tr>
<td>0.333333</td>
<td>0.000000</td>
<td>6</td>
<td>6.0</td>
<td>known</td>
<td>2</td>
<td>160</td>
<td>0.333333</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.363636</td>
<td>0.400000</td>
<td>11</td>
<td>11.0</td>
<td>known</td>
<td>2</td>
<td>202</td>
<td>0.363636</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.454545</td>
<td>0.545455</td>
<td>22</td>
<td>22.0</td>
<td>known</td>
<td>2</td>
<td>502</td>
<td>0.363636</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.363636</td>
<td>0.272727</td>
<td>44</td>
<td>44.0</td>
<td>known</td>
<td>2</td>
<td>370</td>
<td>0.477273</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.275862</td>
<td>0.186047</td>
<td>87</td>
<td>87.0</td>
<td>known</td>
<td>1</td>
<td>340</td>
<td>0.471264</td>
<td>0.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.298851</td>
<td>0.321839</td>
<td>174</td>
<td>174.0</td>
<td>known</td>
<td>2</td>
<td>130</td>
<td>0.459770</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.212644</td>
<td>0.126437</td>
<td>348</td>
<td>348.0</td>
<td>known</td>
<td>2</td>
<td>262</td>
<td>0.465427</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.186782</td>
<td>0.160920</td>
<td>696</td>
<td>696.0</td>
<td>known</td>
<td>2</td>
<td>124</td>
<td>0.472701</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.150862</td>
<td>0.114943</td>
<td>1392</td>
<td>1392.0</td>
<td>known</td>
<td>1</td>
<td>1866</td>
<td>0.481322</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.128592</td>
<td>0.106322</td>
<td>2784</td>
<td>2784.0</td>
<td>known</td>
<td>1</td>
<td>280</td>
<td>0.497845</td>
<td>0.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.108838</td>
<td>0.089080</td>
<td>5568</td>
<td>5568.0</td>
<td>known</td>
<td>1</td>
<td>514</td>
<td>0.497306</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.094836</td>
<td>0.080833</td>
<td>11135</td>
<td>11135.0</td>
<td>known</td>
<td>2</td>
<td>352</td>
<td>0.502829</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.085231</td>
<td>0.075624</td>
<td>22269</td>
<td>22269.0</td>
<td>known</td>
<td>2</td>
<td>820</td>
<td>0.504109</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.075488</td>
<td>0.065745</td>
<td>44537</td>
<td>44537.0</td>
<td>known</td>
<td>2</td>
<td>226</td>
<td>0.502997</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.069067</td>
<td>0.062646</td>
<td>89073</td>
<td>89073.0</td>
<td>known</td>
<td>1</td>
<td>142</td>
<td>0.502565</td>
<td>0.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.063094</td>
<td>0.057122</td>
<td>178146</td>
<td>178146.0</td>
<td>known</td>
<td>2</td>
<td>274</td>
<td>0.502107</td>
<td>0.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.058233</td>
<td>0.053372</td>
<td>356291</td>
<td>356291.0</td>
<td>known</td>
<td>1</td>
<td>580</td>
<td>0.500931</td>
<td>0.000000</td>
<td>1.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>
How do you train?

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

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

$$\frac{(r - \hat{r}(a, x))1(a' = a)}{p(a|x)} + \hat{r}(a', x)$$

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
```
Experimental Results

IPS = Inverse probability
DR = Doubly Robust, with $\hat{r}(a, x) = w_a \cdot x$
Filter Tree = Cost Sensitive Multiclass classifier
Offset Tree = Earlier method for CB learning with same representation

![Graph showing classification error for different datasets](image)
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. **Offset Tree.** (not discussed) Only known logarithmic time method.

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 $\Pi = \{\pi : X \to A\}$:

$$\text{Regret} = \max_{\pi \in \Pi} \text{average}_t (r_{\pi(x)} - r_a)$$
**Exploration** = Choosing not-obviously best actions to gather information for better performance in the future.
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 $\Pi = \{\pi : X \to A\}$:

$$\text{Regret} = \max_{\pi \in \Pi} \text{average}_t (r_{\pi(x)} - r_a)$$
Exploration = Choosing not-obviously best actions to gather information for better performance in the future.
**Exploration** = Choosing not-obviously best actions to gather information for better performance in the future.

There are two kinds:

1. **Deterministic.** Choose action $A$, then $B$, then $C$, then $A$, then $B$, ...

2. **Randomized.** Choose random actions according to some distribution over actions.
**What is exploration?**

**Exploration** = Choosing not-obviously best actions to gather information for better performance in the future. There are two kinds:

1. **Deterministic.** Choose action $A$, then $B$, then $C$, then $A$, then $B$, ...
2. **Randomized.** Choose random actions according to some distribution over actions.

We discuss **Randomized** here.

1. There are no good deterministic exploration algorithms in this setting.
2. Supports off-policy evaluation. (See first half.)
3. Randomize = robust to delayed updates, which are very common in practice.
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 $\Pi = \{\pi : X \rightarrow A\}$:

$$\text{Regret} = \max_{\pi \in \Pi} \text{average}_t (r_{\pi(x)} - r_a)$$
Exploration = Choosing not-obviously best actions to gather information for better performance in the future. There are two kinds:

1. Deterministic. Choose action $A$, then $B$, then $C$, then $A$, then $B$, ...

2. Randomized. Choose random actions according to some distribution over actions.
What is exploration?

Exploration = Choosing not-obviously best actions to gather information for better performance in the future. There are two kinds:

1. **Deterministic.** Choose action $A$, then $B$, then $C$, then $A$, then $B$, ...

2. **Randomized.** Choose random actions according to some distribution over actions.

We discuss Randomized here.

1. There are no good deterministic exploration algorithms in this setting.

2. Supports off-policy evaluation. (See first half.)

3. Randomize = robust to delayed updates, which are very common in practice.
Outline

1. Using Exploration
   1. Problem Definition
   2. Direct Method fails
   3. Importance Weighting
   4. Missing Probabilities
   5. Doubly Robust

2. Doing Exploration
   1. Exploration First
   2. $\epsilon$-Greedy
   3. epoch Greedy
   4. Policy Elimination
   5. Thompson Sampling
Explore \( \tau \) then Follow the Leader (Explore-\( \tau \))
Explore $\tau$ then Follow the Leader (Explore-$\tau$)

Initially, $h = \emptyset$

For the first $\tau$ rounds

1. Observe $x$.
2. Choose $a$ uniform randomly.
3. Observe $r$, and add $(x, a, r)$ to $h$.

For the next $T$ rounds, use empirical best.
Explore $\tau$ then Follow the Leader (Explore-$\tau$)

Initially, $h = \emptyset$

For the first $\tau$ rounds

1. Observe $x$.
2. Choose $a$ uniform randomly.
3. Observe $r$, and add $(x, a, r)$ to $h$.

For the next $T$ rounds, use empirical best.

Suppose all examples are drawn from a fixed distribution $D(x, r)$.

Theorem: For all $D, \Pi, \text{Explore-$\tau$}$ has regret $O \left( \frac{\tau}{T} + \sqrt{\frac{|A| \ln |\Pi|}{\tau}} \right)$

with high probability.
Explore $\tau$ then Follow the Leader (Explore-$\tau$)

Initially, $h = \emptyset$
For the first $\tau$ rounds
\begin{enumerate}
\item Observe $x$.
\item Choose $a$ uniform randomly.
\item Observe $r$, and add $(x, a, r)$ to $h$.
\end{enumerate}
For the next $T$ rounds, use empirical best.

Suppose all examples are drawn from a fixed distribution $D(x, \bar{r})$.
Theorem: For all $D$, $\Pi$, Explore-$\tau$ has regret $O\left(\frac{\tau}{T} + \sqrt{\frac{|A| \ln (|\Pi|/\delta)}{\tau}}\right)$
with high probability.

Proof: After $\tau$ rounds, a large deviation bound implies empirical average value of a policy deviates from expectation $E_{(x, \bar{r}) \sim D}[r_\pi(x)]$
by at most $\sqrt{\frac{|A| \ln (|\Pi|/\delta)}{\tau}}$, so regret is bounded by
\[
\frac{\tau}{T} + \frac{T}{T} \sqrt{\frac{|A| \ln (|\Pi|/\delta)}{\tau}}.
\]
What does this mean?

1. **Easiest approach**: offline prerecorded exploration can feed into any learning algorithm. See first half.
2. **Doesn’t adapt** when world changes.
3. **Underexploration common**. Think of clinical trials.
Explore $\tau$ then Follow the Leader (Explore-$\tau$)

Initially, $h = \emptyset$

For the first $\tau$ rounds

1. Observe $x$.
2. Choose $a$ uniform randomly.
3. Observe $r$, and add $(x, a, r)$ to $h$.

For the next $T$ rounds, use empirical best.

Suppose all examples are drawn from a fixed distribution $D(x, \bar{r})$.

Theorem: For all $D, \Pi$, Explore-$\tau$ has regret $O \left( \frac{\tau}{T} + \sqrt{\frac{|A| \ln |\Pi|}{\tau}} \right)$

with high probability.

Proof: After $\tau$ rounds, a large deviation bound implies empirical average value of a policy deviates from expectation $E_{(x, \bar{r}) \sim D} [r_\pi(x)]$ by at most $\sqrt{\frac{|A| \ln(|\Pi|/\delta)}{\tau}}$, so regret is bounded by

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

At optimal $\tau$? $O \left( \left( \frac{|A| \ln |\Pi|}{T} \right)^{1/3} \right)$
What does this mean?

1. **Easiest approach**: offline prerecorded exploration can feed into any learning algorithm. See first half.
2. - **Doesn’t adapt** when world changes.
3. - **Underexploration common**. Think of clinical trials.
$\epsilon$-Greedy

1. Observe $x$.
2. With probability $1 - \epsilon$
   1. Choose learned $a$
   2. Observe $r$, and learn with $(x, a, r, 1 - \epsilon)$. 
**ε-Greedy**

1. Observe $x$.

2. With probability $1 - \varepsilon$
   
   1. Choose learned $a$
   2. Observe $r$, and learn with $(x, a, r, 1 - \varepsilon)$.

   With probability $\varepsilon$

   1. Choose Uniform random other $a$
   2. Observe $r$, and learn with $(x, a, r, \varepsilon/(|A| - 1))$. 
**ε-Greedy**

1. Observe $x$.
2. With probability $1 - \varepsilon$
   1. Choose learned $a$
   2. Observe $r$, and learn with $(x, a, r, 1 - \varepsilon)$.
   
   With probability $\varepsilon$
   1. Choose Uniform random other $a$
   2. Observe $r$, and learn with $(x, a, r, \varepsilon/(|A| - 1))$.

Theorem: $\varepsilon$-Greedy has regret $O\left(\varepsilon + \sqrt{\frac{|A| \ln |\Pi|}{T \varepsilon}}\right)$
What does this mean?

1. - **Harder Approach**: Need online learning algorithm to use.
2. + **Adapts** when world changes.
3. - **Overexploration common**. Bad possibilities keep being explored.
What does this mean?

1. **Harder Approach**: Need online learning algorithm to use.
2. **Adapts** when world changes.
3. **Overexploration common**: Bad possibilities keep being explored.

Can we do better?
At every timestep $t$, the learned policy has an empirical performance known up to some precision $\epsilon_t$ which can be estimated.
Epoch Greedy

At every timestep $t$, the learned policy has an empirical performance known up to some precision $\epsilon_t$ which can be estimated.

1. Observe $x$.
2. With probability $1 - \epsilon_t$
   1. Choose learned $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, 1 - \epsilon_t)$. 
Epoch Greedy

At every timestep $t$, the learned policy has an empirical performance known up to some precision $\epsilon_t$ which can be estimated.

1. Observe $x$.
2. With probability $1 - \epsilon_t$
   1. Choose learned $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, 1 - \epsilon_t)$.

With probability $\epsilon_t$

1. Choose Uniform random other $a$
2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, \epsilon_t/(|A| - 1))$. 
Epoch Greedy

At every timestep $t$, the learned policy has an empirical performance known up to some precision $\epsilon_t$ which can be estimated.

1. Observe $x$.
2. With probability $1 - \epsilon_t$
   1. Choose learned $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, 1 - \epsilon_t)$.
   With probability $\epsilon_t$
   1. Choose Uniform random other $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, \epsilon_t/(|A| - 1))$.

Theorem: Epoch Greedy has regret $O\left(\left(\frac{|A| \ln |\Pi|}{T}\right)^{1/3}\right)$ with high probability.
Autotuning!
What does this mean?

1. **Harder Approach**: Need online learning algorithm to use + keeping track of deviation bound.
2. +Adapts when world changes.
3. +Neither under nor over exploration.
What does this mean?

1. **Harder Approach**: Need online learning algorithm to use + keeping track of deviation bound.
2. +Adapts when world changes.
3. +Neither under nor over exploration.

Is it possible to do better?

<table>
<thead>
<tr>
<th></th>
<th>Supervised</th>
<th>$\tau$-first/$\epsilon$-greedy/epoch-greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regret</td>
<td>$O\left(\left(\frac{\ln</td>
<td>\Pi</td>
</tr>
</tbody>
</table>
Epoch Greedy

At every timestep $t$, the learned policy has an empirical performance known up to some precision $\epsilon_t$ which can be estimated.

1. Observe $x$.
2. With probability $1 - \epsilon_t$
   1. Choose learned $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, 1 - \epsilon_t)$.
With probability $\epsilon_t$
   1. Choose Uniform random other $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, \epsilon_t/(|A| - 1))$. 
What does this mean?

1. **Harder Approach**: Need online learning algorithm to use.
2. **Adapts** when world changes.
3. **Overexploration common**. Bad possibilities keep being explored.

Can we do better?
\( \epsilon \)-Greedy

1. Observe \( x \).
2. With probability \( 1 - \epsilon \)
   1. Choose learned \( a \)
   2. Observe \( r \), and learn with \( (x, a, r, 1 - \epsilon) \).

With probability \( \epsilon \)
1. Choose Uniform random other \( a \)
2. Observe \( r \), and learn with \( (x, a, r, \epsilon/(|A| - 1)) \).

Theorem: \( \epsilon \)-Greedy has regret \( O\left( \epsilon + \sqrt{\frac{|A| \ln |\Pi|}{T \epsilon}} \right) \)

For optimal epsilon? \( O\left( \left( \frac{|A| \ln |\Pi|}{T} \right)^{1/3} \right) \)
What does this mean?

1. **Easiest approach**: offline prerecorded exploration can feed into any learning algorithm. See first half.
2. **Doesn’t adapt** when world changes.
3. **Underexploration common**. Think of clinical trials.

Can we do better?
Explore $\tau$ then Follow the Leader (Explore-$\tau$)

Initially, $h = \emptyset$

For the first $\tau$ rounds

1. Observe $x$.
2. Choose $a$ uniform randomly.
3. Observe $r$, and add $(x, a, r)$ to $h$.

For the next $T$ rounds, use empirical best.

Suppose all examples are drawn from a fixed distribution $D(x, r)$.

Theorem: For all $D, \Pi, \text{Explore-}\tau$ has regret $O\left(\frac{\tau}{T} + \sqrt{\frac{|A| \ln |\Pi|}{\tau}}\right)$ with high probability.

Proof: After $\tau$ rounds, a large deviation bound implies empirical average value of a policy deviates from expectation $E_{(x, r) \sim D}[r_{\pi(x)}]$ by at most $\sqrt{\frac{|A| \ln (|\Pi|/\delta)}{\tau}}$, so regret is bounded by

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

At optimal $\tau$? $O\left((\frac{|A| \ln |\Pi|}{T})^{1/3}\right)$
\( \epsilon \)-Greedy

1. Observe \( x \).
2. With probability \( 1 - \epsilon \)
   1. Choose learned \( a \)
   2. Observe \( r \), and learn with \((x, a, r, 1 - \epsilon)\).
With probability \( \epsilon \)
1. Choose Uniform random other \( a \)
2. Observe \( r \), and learn with \((x, a, r, \epsilon/(|A| - 1))\).

Theorem: \( \epsilon \)-Greedy has regret \( O \left( \epsilon + \sqrt{\frac{|A| \ln |\Pi|}{T \epsilon}} \right) \)
At every timestep $t$, the learned policy has an empirical performance known up to some precision $\epsilon_t$ which can be estimated.

1. Observe $x$.
2. With probability $1 - \epsilon_t$
   1. Choose learned $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, 1 - \epsilon_t)$. 
Epoch Greedy

At every timestep $t$, the learned policy has an empirical performance known up to some precision $\epsilon_t$ which can be estimated.

1. Observe $x$.
2. With probability $1 - \epsilon_t$
   1. Choose learned $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, 1 - \epsilon_t)$.
With probability $\epsilon_t$
   1. Choose Uniform random other $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, \epsilon_t/(|A| - 1))$.

Theorem: Epoch Greedy has regret $O \left( \left( \frac{|A| \ln |\Pi|}{T} \right)^{1/3} \right)$ with high probability.
Autotuning!
What does this mean?

1. **Harder Approach**: Need online learning algorithm to use + keeping track of deviation bound.
2. +Adapts when world changes.
3. +Neither under nor over exploration.
What does this mean?

1. **Harder Approach**: Need online learning algorithm to use + keeping track of deviation bound.
2. +Adapts when world changes.
3. +Neither under nor over exploration.

Is it possible to do better?

<table>
<thead>
<tr>
<th></th>
<th>Supervised</th>
<th>$\tau$-first/$\epsilon$-greedy/epoch-greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regret</td>
<td>$O\left(\left(\frac{\ln</td>
<td>\Pi</td>
</tr>
</tbody>
</table>
Policy Elimination

Let $\Pi_0 = \Pi$ and $\mu_t = 1/\sqrt{Kt}$ and $\eta_t(\pi) =$ empirical reward
For each $t = 1, 2, \ldots$

1. Choose distribution $P$ over $\Pi_{t-1}$ s.t. for every remaining policy $\pi$, the expected variance of a value estimate is small.
2. observe $x$
3. Let $p(a) =$ fraction of $P$ choosing $a$ given $x$.
4. Choose $a \sim p$ and observe reward $r$
5. Let $\Pi_t =$ remaining near empirical best policies.

Theorem: With high probability Policy Elimination has regret

$$O\left(\sqrt{\frac{|A| \ln |\Pi|}{T}}\right)$$
Policy Elimination

Let $\Pi_0 = \Pi$ and $\mu_t = 1/\sqrt{K}t$ and $\eta_t(\pi) =$ empirical reward
For each $t = 1, 2, \ldots$

1. Choose distribution $P$ over $\Pi_{t-1}$ s.t. $\forall \pi \in \Pi_{t-1}$:
   $$\mathbb{E}_{x \sim D_x} \left[ \frac{1}{(1-K\mu_t) \Pr_{\pi' \sim P}(\pi'(x) = \pi(x)) + \mu_t} \right] \leq 2K$$
2. observe $x$
3. Let $p(a) =$ fraction of $P$ choosing $a$ given $x$.
4. Choose $a \sim p$ and observe reward $r$
5. Let $\Pi_t =$ remaining near empirical best policies.

Theorem: With high probability Policy Elimination has regret

$$O\left(\sqrt{\frac{|A| \ln |\Pi|}{T}}\right)$$
Better 1: Policy Elimination

Policy Elimination

Let $\Pi_0 = \Pi$ and $\mu_t = 1/\sqrt{Kt}$ and $\eta_t(\pi) =$ empirical reward
For each $t = 1, 2, \ldots$

1. Choose distribution $P$ over $\Pi_{t-1}$ s.t. $\forall \pi \in \Pi_{t-1}$:
   
   $\mathbb{E}_{x \sim D_x} \left[ \frac{1}{(1-K\mu_t) \Pr_{\pi' \sim P}(\pi'(x) = \pi(x)) + \mu_t} \right] \leq 2K$

2. observe $x$

3. Let $p(a) = (1 - K\mu_t) \Pr_{\pi \sim P}(\pi(x) = a) + \mu_t$

4. Choose $a \sim p$ and observe reward $r$

5. Let $\Pi_t = \{ \pi \in \Pi_{t-1} : \eta_t(\pi) \geq \max_{\pi' \in \Pi_{t-1}} \eta_t(\pi') - K\mu_t \}$

Theorem: With high probability Policy Elimination has regret

$O \left( \sqrt{\frac{|A| \ln |\Pi|}{T}} \right)$
What does this mean?

1. Doesn't adapt when world changes.
2. Much more efficient exploration. Only efficient in special cases.
3. Much Harder Approach: Need to keep track of policies, which is often intractable.
Policy Elimination

Let $\Pi_0 = \Pi$ and $\mu_t = 1/\sqrt{Kt}$ and $\eta_t(\pi) =$ empirical reward

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

1. Choose distribution $P$ over $\Pi_{t-1}$ s.t. $\forall \pi \in \Pi_{t-1}$:
   $$E_{x \sim D_x} \left[ \frac{1}{(1-K\mu_t) Pr_{\pi' \sim P} (\pi'(x) = \pi(x)) + \mu_t} \right] \leq 2K$$

2. observe $x$

3. Let $p(a) = (1 - K\mu_t) Pr_{\pi \sim P} (\pi(x) = a) + \mu_t$

4. Choose $a \sim p$ and observe reward $r$

5. Let $\Pi_t = \{ \pi \in \Pi_{t-1} : \eta_t(\pi) \geq \max_{\pi' \in \Pi_{t-1}} \eta_t(\pi') - K\mu_t \}$

Theorem: With high probability Policy Elimination has regret

$$O \left( \sqrt{\frac{|A| \ln |\Pi|}{T}} \right)$$
What does this mean?

1. Doesn’t adapt when world changes.
2. Much more efficient exploration. Only efficient in special cases.
3. Much Harder Approach: Need to keep track of policies, which is often intractable.
What does this mean?

1. Doesn't adapt when world changes.
2. **Much more efficient exploration.** Only efficient in special cases.
3. **- Much Harder Approach:** Need to keep track of policies, which is often intractable.

Adapting algorithms exist (EXP4).
More efficient versions exist (RUCB), but not yet efficient enough.
Can you do better?
What does this mean?

1. Doesn’t adapt when world changes.
2. ++Much more efficient exploration. Only efficient in special cases.
3. - -Much Harder Approach: Need to keep track of policies, which is often intractable.

Adapting algorithms exist (EXP4).
More efficient versions exist (RUCB), but not yet efficient enough.
Better 1: Policy Elimination

Policy Elimination

Let $\Pi_0 = \Pi$ and $\mu_t = 1/\sqrt{Kt}$ and $\eta_t(\pi) =$ empirical reward
For each $t = 1, 2, \ldots$

1. Choose distribution $P$ over $\Pi_{t-1}$ s.t. $\forall \pi \in \Pi_{t-1}$:
   $$\mathbb{E}_{x \sim D_x} \left[ \frac{1}{(1 - K\mu_t) \Pr_{\pi' \sim P}(\pi'(x) = \pi(x)) + \mu_t} \right] \leq 2K$$

2. observe $x$

3. Let $p(a) = (1 - K\mu_t) \Pr_{\pi \sim P}(\pi(x) = a) + \mu_t$

4. Choose $a \sim p$ and observe reward $r$

5. Let $\Pi_t = \{ \pi \in \Pi_{t-1} : \eta_t(\pi) \geq \max_{\pi' \in \Pi_{t-1}} \eta_t(\pi') - K\mu_t \}$

Theorem: With high probability Policy Elimination has regret

$$O\left(\sqrt{\frac{|A| \ln |\Pi|}{T}}\right)$$
What does this mean?

1. Doesn't adapt when world changes.
2. Much more efficient exploration. Only efficient in special cases.
3. Much Harder Approach: Need to keep track of policies, which is often intractable.

Adapting algorithms exist (EXP4).
More efficient versions exist (RUCB), but not yet efficient enough.
Policy Elimination

Let $\Pi_0 = \Pi$ and $\mu_t = 1/\sqrt{Kt}$ and $\eta_t(\pi) =$empirical reward
For each $t = 1, 2, \ldots$

1. Choose distribution $P$ over $\Pi_{t-1}$ s.t. $\forall \pi \in \Pi_{t-1}$:
   \[ \mathbb{E}_{x \sim D_x} \left[ \frac{1}{(1 - K\mu_t) \Pr_{\pi' \sim P}(\pi'(x) = \pi(x)) + \mu_t} \right] \leq 2K \]
2. observe $x$
3. Let $p(a) = (1 - K\mu_t) \Pr_{\pi \sim P}(\pi(x) = a) + \mu_t$
4. Choose $a \sim p$ and observe reward $r$
5. Let $\Pi_t = \{ \pi \in \Pi_{t-1} : \eta_t(\pi) \geq \max_{\pi' \in \Pi_{t-1}} \eta_t(\pi') - K\mu_t \}$

Theorem: With high probability Policy Elimination has regret

\[ O\left( \sqrt{\frac{|A| \ln |\Pi|}{T}} \right) \]
What does this mean?

1. Doesn’t adapt when world changes.
2. Much more efficient exploration. Only efficient in special cases.
3. Much Harder Approach: Need to keep track of policies, which is often intractable.

Adapting algorithms exist (EXP4).
More efficient versions exist (RUCB), but not yet efficient enough.
Can you do better?
Can you do better?

Not in general.

Theorem: For all algorithms, there exists problems imposing regret:

$$\Omega \left( \sqrt{\frac{|A| \ln |\Pi|}{T}} \right)$$
Always maintain a Bayesian posterior over policies. On each round sample policy from posterior, and act according to it.
Can you do better?
Better 2: Thompson Sampling

Always maintain a Bayesian posterior over policies. On each round sample policy from posterior, and act according to it.
Better 2: Thompson Sampling

Always maintain a Bayesian posterior over policies. On each round sample policy from posterior, and act according to it.

An efficient special case: Gaussian Posterior.

**Thompson Sampling**

Let $w =$ mean 0 multivariate gaussian.

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

1. Draw $w' \sim w$

2. Observe $x$

3. Choose $a = \max_{a'} w' x_{a'}$

4. Observe reward $r$.

5. Bayesian update $w$ with $(x, a, r)$. 
What does it mean?

1. **+ Efficient special cases for Gaussian posteriors.**
2. **+ Known to work well empirically sometimes.**
3. **- Not robust to model misspecification: \( \tilde{\Omega}\left(\frac{\log T}{T}\right) \) regret.**
The current state

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Starter</td>
<td></td>
</tr>
<tr>
<td>Baseline</td>
<td></td>
</tr>
<tr>
<td>Purring</td>
<td></td>
</tr>
<tr>
<td>Shiny</td>
<td></td>
</tr>
<tr>
<td>Something to try</td>
<td></td>
</tr>
</tbody>
</table>
## The current state

<table>
<thead>
<tr>
<th>Explore-τ</th>
<th>Simplest Possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td></td>
</tr>
<tr>
<td>Purring</td>
<td></td>
</tr>
<tr>
<td>Shiny</td>
<td></td>
</tr>
<tr>
<td>Something to try</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Explore-$\tau$</td>
<td>Simplest Possible</td>
</tr>
<tr>
<td>$\epsilon$-Greedy</td>
<td>Simplest Adaptive</td>
</tr>
<tr>
<td>Purring</td>
<td></td>
</tr>
<tr>
<td>Shiny</td>
<td></td>
</tr>
<tr>
<td>Something to try</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Explore-$\tau$</td>
<td>Simplest Possible</td>
</tr>
<tr>
<td>$\epsilon$-Greedy</td>
<td>Simplest Adaptive</td>
</tr>
<tr>
<td>Epoch Greedy</td>
<td>Unequivocal Improvement</td>
</tr>
<tr>
<td>Shiny</td>
<td></td>
</tr>
<tr>
<td>Something to try</td>
<td></td>
</tr>
<tr>
<td>Strategy</td>
<td>Evaluation</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Explore-$\tau$</td>
<td>Simplest Possible</td>
</tr>
<tr>
<td>$\epsilon$-Greedy</td>
<td>Simplest Adaptive</td>
</tr>
<tr>
<td>Epoch Greedy</td>
<td>Unequivocal Improvement</td>
</tr>
<tr>
<td>Policy Elimination</td>
<td>Optimal Impractical</td>
</tr>
<tr>
<td>Something to try</td>
<td></td>
</tr>
</tbody>
</table>
## The current state

<table>
<thead>
<tr>
<th>Method</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explore-(\tau)</td>
<td>Simplest Possible</td>
</tr>
<tr>
<td>(\epsilon)-Greedy</td>
<td>Simplest Adaptive</td>
</tr>
<tr>
<td>Epoch Greedy</td>
<td>Unequivocal Improvement</td>
</tr>
<tr>
<td>Policy Elimination</td>
<td>Optimal Impractical</td>
</tr>
<tr>
<td>Thompson Sampling</td>
<td>Sometimes Excellent</td>
</tr>
</tbody>
</table>
### The current state

<table>
<thead>
<tr>
<th>Method</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explore-$\tau$</td>
<td>Simplest Possible</td>
</tr>
<tr>
<td>$\epsilon$-Greedy</td>
<td>Simplest Adaptive</td>
</tr>
<tr>
<td>Epoch Greedy</td>
<td>Unequivocal Improvement</td>
</tr>
<tr>
<td>Policy Elimination</td>
<td>Optimal Impractical</td>
</tr>
<tr>
<td>Thompson Sampling</td>
<td>Sometimes Excellent</td>
</tr>
</tbody>
</table>

You can see the edge of the understood world here. We hope to see further soon.

Further discussion: [http://hunch.net](http://hunch.net)
Inverse  An old technique, not sure where it was first used.

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


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.
Bibliography: Doing Exploration

Tau-first Unclear first use?

$\epsilon$-Greedy Unclear first use?


<table>
<thead>
<tr>
<th>Bibliography: Doing Exploration II</th>
</tr>
</thead>
</table>
### The current state

<table>
<thead>
<tr>
<th>Method</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explore-$\tau$</td>
<td>Simplest Possible</td>
</tr>
<tr>
<td>$\epsilon$-Greedy</td>
<td>Simplest Adaptive</td>
</tr>
<tr>
<td>Epoch Greedy</td>
<td>Unequivocal Improvement</td>
</tr>
<tr>
<td>Policy Elimination</td>
<td>Optimal Impractical</td>
</tr>
<tr>
<td>Thompson Sampling</td>
<td>Sometimes Excellent</td>
</tr>
</tbody>
</table>

You can see the edge of the understood world here. We hope to see further soon.

Further discussion: [http://hunch.net](http://hunch.net)
Policy Elimination

Let $\Pi_0 = \Pi$ and $\mu_t = 1/\sqrt{Kt}$ and $\eta_t(\pi) =$ empirical reward
For each $t = 1, 2, \ldots$

1. Choose distribution $P$ over $\Pi_{t-1}$ s.t. for every remaining policy $\pi$, the expected variance of a value estimate is small.

2. Observe $x$

3. Let $p(a) =$ fraction of $P$ choosing $a$ given $x$.

4. Choose $a \sim p$ and observe reward $r$

5. Let $\Pi_t =$ remaining nearly empirical best policies.

Theorem: With high probability Policy Elimination has regret

$$O\left(\sqrt{\frac{|A| \ln |\Pi|}{T}}\right)$$
ε-Greedy

1. Observe $x$.
2. With probability $1 - \varepsilon$
   1. Choose learned $a$
   2. Observe $r$, and learn with $(x, a, r, 1 - \varepsilon)$.
With probability $\varepsilon$
   1. Choose Uniform random other $a$
   2. Observe $r$, and learn with $(x, a, r, \varepsilon/(|A| - 1))$.

Theorem: $\varepsilon$-Greedy has regret $O\left(\varepsilon + \sqrt{\frac{|A| \ln |\Pi|}{T\varepsilon}}\right)$
Explore \( \tau \) then Follow the Leader (Explore-\( \tau \))

Initially, \( h = \emptyset \)

For the first \( \tau \) rounds

1. Observe \( x \).
2. Choose \( a \) uniform randomly.
3. Observe \( r \), and add \((x, a, r)\) to \( h \).

For the next \( T \) rounds, use empirical best.

Suppose all examples are drawn from a fixed distribution \( D(x, \bar{r}) \).

Theorem: For all \( D, \Pi \), Explore-\( \tau \) has regret \( O \left( \frac{\tau}{T} + \sqrt{\frac{|A| \ln |\Pi|}{\tau}} \right) \)

with high probability.

Proof: After \( \tau \) rounds, a large deviation bound implies empirical average value of a policy deviates from expectation \( E_{(x, \bar{r}) \sim D}[r_{\pi(x)}] \)
by at most \( \sqrt{\frac{|A| \ln (|\Pi|/\delta)}{\tau}} \), so regret is bounded by

\[
\frac{\tau}{T} + \frac{T}{T} \sqrt{\frac{|A| \ln (|\Pi|/\delta)}{\tau}}.
\]
Exploration = Choosing not-obviously best actions to gather information for better performance in the future. There are two kinds:

1. **Deterministic.** Choose action $A$, then $B$, then $C$, then $A$, then $B$, ...

2. **Randomized.** Choose random actions according to some distribution over actions.

We discuss **Randomized** here.

1. There are no good deterministic exploration algorithms in this setting.

2. Supports off-policy evaluation. (See first half.)

3. Randomize $=$ robust to delayed updates, which are very common in practice.
What is exploration?

Exploration = Choosing not-obviously best actions to gather information for better performance in the future.
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 $\Pi = \{\pi : X \to A\}$:

$$\text{Regret} = \max_{\pi \in \Pi} \text{average}_t (r_{\pi(x)} - r_a)$$
Explore $\tau$ then Follow the Leader (Explore-$\tau$)

Initially, $h = \emptyset$

For the first $\tau$ rounds

1. Observe $x$.
2. Choose $a$ uniform randomly.
3. Observe $r$, and add $(x, a, r)$ to $h$.

For the next $T$ rounds, use empirical best.

Suppose all examples are drawn from a fixed distribution $D(x, \bar{r})$.

Theorem: For all $D, \Pi$, Explore-$\tau$ has regret $O\left(\frac{\tau}{T} + \sqrt{\frac{|A| \ln |\Pi|}{\tau}}\right)$

with high probability.

Proof: After $\tau$ rounds, a large deviation bound implies empirical average value of a policy deviates from expectation $E_{(x, \bar{r}) \sim D} [r_\pi(x)]$ by at most $\sqrt{\frac{|A| \ln (|\Pi|/\delta)}{\tau}}$, so regret is bounded by

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

At optimal $\tau$?
1. Observe $x$.

2. With probability $1 - \epsilon$
   1. Choose learned $a$
   2. Observe $r$, and learn with $(x, a, r, 1 - \epsilon)$.

With probability $\epsilon$

1. Choose Uniform random other $a$
2. Observe $r$, and learn with $(x, a, r, \epsilon/(|A| - 1))$. 
At every timestep $t$, the learned policy has an empirical performance known up to some precision $\epsilon_t$ which can be estimated.

1. Observe $x$.
2. With probability $1 - \epsilon_t$
   1. Choose learned $a$
   2. Observe $r$, update $\epsilon_t$ and learn with $(x, a, r, 1 - \epsilon_t)$. 
What does this mean?

1. **Harder Approach**: Need online learning algorithm to use + keeping track of deviation bound.
2. +Adapts when world changes.
3. +Neither under nor over exploration.

Is it possible to do better?

<table>
<thead>
<tr>
<th></th>
<th>Supervised</th>
<th>$\tau$-first/$\epsilon$-greedy/epoch-greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regret</td>
<td>$O \left( \left( \frac{\ln</td>
<td>\Pi</td>
</tr>
</tbody>
</table>
Policy Elimination

Let $\Pi_0 = \Pi$ and $\mu_t = 1/\sqrt{Kt}$ and $\eta_t(\pi) =$ empirical reward
For each $t = 1, 2, \ldots$

1. Choose distribution $P$ over $\Pi_{t-1}$ s.t. for every remaining policy $\pi$, the expected variance of a value estimate is small.

2. observe $x$

3. Let $p(a) =$ fraction of $P$ choosing $a$ given $x$.

4. Choose $a \sim p$ and observe reward $r$

5. Let $\Pi_t =$ remaining near empirical best policies.

Theorem: With high probability Policy Elimination has regret

$$O\left(\sqrt{\frac{|A| \ln |\Pi|}{T}}\right)$$
Policy Elimination

Let $\Pi_0 = \Pi$ and $\mu_t = 1/\sqrt{Kt}$ and $\eta_t(\pi) =$empirical reward
For each $t = 1, 2, \ldots$

1. Choose distribution $P$ over $\Pi_{t-1}$ s.t. $\forall \pi \in \Pi_{t-1}$:
   $$E_{x \sim D_x} \left[ \frac{1}{(1-K\mu_t) \Pr_{\pi' \sim P} (\pi'(x) = \pi(x)) + \mu_t} \right] \leq 2K$$

2. observe $x$

3. Let $p(a) =$ fraction of $P$ choosing $a$ given $x$.

4. Choose $a \sim p$ and observe reward $r$

5. Let $\Pi_t =$ remaining near empirical best policies.

Theorem: With high probability Policy Elimination has regret

$$O\left( \sqrt{\frac{|A| \ln |\Pi|}{T}} \right)$$
What does this mean?

1. Doesn't adapt when world changes.
2. Much more efficient exploration. Only efficient in special cases.
3. Much Harder Approach: Need to keep track of policies, which is often intractable.
Policy Elimination

Let \( \Pi_0 = \Pi \) and \( \mu_t = 1/\sqrt{Kt} \) and \( \eta_t(\pi) \) = empirical reward
For each \( t = 1, 2, \ldots \)

1. Choose distribution \( P \) over \( \Pi_{t-1} \) s.t. \( \forall \pi \in \Pi_{t-1}: \)
   \[ E_{x \sim D_x} \left[ \frac{1}{(1 - K\mu_t) \Pr_{\pi' \sim P}(\pi'(x) = \pi(x)) + \mu_t} \right] \leq 2K \]

2. observe \( x \)

3. Let \( p(a) = (1 - K\mu_t) \Pr_{\pi \sim P}(\pi(x) = a) + \mu_t \)

4. Choose \( a \sim p \) and observe reward \( r \)

5. Let \( \Pi_t = \{ \pi \in \Pi_{t-1} : \eta_t(\pi) \geq \max_{\pi' \in \Pi_{t-1}} \eta_t(\pi') - K\mu_t \} \)

Theorem: With high probability Policy Elimination has regret

\[ O\left( \sqrt{\frac{|A| \ln |\Pi|}{T}} \right) \]
What does this mean?

1. **Harder Approach**: Need online learning algorithm to use and keeping track of deviation bound.
2. +Adapts when world changes.
3. +Neither under nor over exploration.

Is it possible to do better?

<table>
<thead>
<tr>
<th></th>
<th>Supervised</th>
<th>$\tau$-first/$\varepsilon$-greedy/epoch-greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regret</td>
<td>$O \left( \left( \frac{\ln</td>
<td>\Pi</td>
</tr>
</tbody>
</table>
What is exploration?

Exploration = Choosing not-obviously best actions to gather information for better performance in the future. There are two kinds:

1. **Deterministic.** Choose action $A$, then $B$, then $C$, then $A$, then $B$, ...

2. **Randomized.** Choose random actions according to some distribution over actions.
Summary of methods


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. Offset Tree. (not discussed) Only known logarithmic time method.

Experimental Results

\[\text{IPS} = \text{Inverse probability}\]
\[\text{DR} = \text{Doubly Robust, with } \hat{r}(a, x) = w_a \cdot x\]

Filter Tree = Cost Sensitive Multiclass classifier
Offset Tree = Earlier method for CB learning with same representation

![Graph showing classification error for different datasets with IPS, DR, and Offset Tree comparisons.](image-url)
How do you train?

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

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

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

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
   ```
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.
Can we do better?

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

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

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

Theorem

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

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

The deviations multiply, so deviations \(< 1\) means we win!
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”.

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

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

$$\text{Value}(\pi) = \text{Average} \left( \frac{r_a \mathbf{1}(\pi(x) = a)}{p_a} \right)$$
Method 3: The Importance Weighting Trick

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

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

with deviations bounded by

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

Proof: [Part 1] $\mathbb{E}_{a \sim p} \left[ \frac{r_{a1(\pi(x)=a)}}{p_a} \right] = \sum_a p_a \frac{r_{a1(\pi(x)=a)}}{p_a} = r_{\pi}(x)$