1  A user with some hidden interests make a query on Yahoo.

2  Yahoo chooses an ad to display.

3  The user either clicks on the ad or not, (resulting in a payoff to Yahoo or not).
A Mathematical Description

Data generation process:

1. The world chooses \((x, r_1, \ldots, r_k)\) and reveals \(x\).

2. A policy chooses \(a \in \{1, \ldots, k\}\) according to some distribution (uniform for the talk).

3. The world reveals \(r_a\).
A Mathematical Description

Data generation process:

1. The world chooses \((x, r_1, \ldots, r_k)\) and reveals \(x\).
2. A policy chooses \(a \in \{1, \ldots, k\}\) according to some distribution (uniform for the talk).
3. The world reveals \(r_a\).

**Goal:** Find a policy \(\pi : X \rightarrow \{1, \ldots, k\}\) maximizing the expected reward

\[
E_{(x, \vec{r}) \sim D} \left[ r_{\pi(x)} \right]
\]

with respect to the underlying distribution \(D\) over \(X \times [0, 1]^k\).
A Mathematical Description

Data generation process:

1. The world chooses \((x, r_1, \ldots, r_k)\) and reveals \(x\).

2. A policy chooses \(a \in \{1, \ldots, k\}\) according to some distribution (uniform for the talk).

3. The world reveals \(r_a\).

Goal: Find a policy \(\pi : X \rightarrow \{1, \ldots, k\}\) maximizing the expected reward

\[
E_{(x, \bar{r}) \sim D} \left[ r_{\pi(x)} \right]
\]

with respect to the underlying distribution \(D\) over \(X \times [0, 1]^k\).

Loss is unknown even at training time! Exploration required, but still simpler than reinforcement learning.
The Offset Trick for \( k = 2 \) (two actions)

Partial label sample \( (x, a, r_a) \mapsto \) binary importance weighted sample

\[
\begin{cases}
(x, a, r_a - \frac{1}{2}) & \text{if } r_a \geq \frac{1}{2} \\
(x, \bar{a}, \frac{1}{2} - r_a) & \text{if } r_a < \frac{1}{2}
\end{cases}
\]

\( x = \) side information

\( \bar{a} = \) the other label (action)

\(|r_a - \frac{1}{2}| = \) importance weight
The Offset Trick for \( k = 2 \) (two actions)

Partial label sample \((x, a, r_a)\) \(\mapsto\) binary importance weighted sample

\[
\begin{cases}
(x, a, r_a - \frac{1}{2}) & \text{if } r_a \geq \frac{1}{2} \\
(x, \bar{a}, \frac{1}{2} - r_a) & \text{if } r_a < \frac{1}{2}
\end{cases}
\]

\( x = \) side information

\( \bar{a} = \) the other label (action)

\(|r_a - \frac{1}{2}| = \) importance weight

Remove the weights using rejection sampling with probability proportionate to the weight [Zadrozny, L, Abe, ICDM2002]

Binary classification problem
The Offset Trick for $k = 2$ (two actions)

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

\[
\begin{cases} 
(x, a, r_a - \frac{1}{2}) \quad \text{if } r_a \geq \frac{1}{2} \\
(x, \overline{a}, \frac{1}{2} - r_a) \quad \text{if } r_a < \frac{1}{2} 
\end{cases}
\]

$x = \text{side information}$

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

$|r_a - \frac{1}{2}| = \text{importance weight}$

Remove the weights using rejection sampling with probability proportionate to the weight [Zadrozny, L, Abe, ICDM2002] $\mapsto$ Binary classification problem

Learn a binary classifier and use it as a partial label policy
Induced binary distribution $D'$

- Draw partial label sample $(x, \vec{r}) \sim D$ and action $a$.
- With probability $2|r_a - \frac{1}{2}|$:
  - If $r_a \geq \frac{1}{2}$, generate $(x, a)$; otherwise generate $(x, \overline{a})$.
- The induced problem is noisy. The importance trick reduces the range of importances, reducing the noise rate.
Induced binary distribution $D'$

- Draw partial label sample $(x, \bar{r}) \sim D$ and action $a$.
- With probability $2|r_a - \frac{1}{2}|$:
  
  If $r_a \geq \frac{1}{2}$, generate $(x, a)$; otherwise generate $(x, a)$.
- The induced problem is noisy. The importance trick reduces the range of importances, reducing the noise rate.

Examples: Actions 1 and 2

1. $r_1 = \frac{1}{2}, r_2 = 1$: Examples of class 1 have weight 0; learn a constant 2 classifier.

2. $r_1 = 0, r_2 = 1$: All examples have class 2 with the same weight; learn a constant 2 classifier.

3. $r_1 = 0.75, r_2 = 1$: $D'(1) = \frac{1}{3}, D'(2) = \frac{2}{3}$. Some examples have each label, but the proportion is improved by the offset.
Analysis for \( k = 2 \)

Binary regret of classifier \( f \) on \( D' \):

\[
\text{reg}_{0/1}(f, D') = \Pr_{(x,y) \sim D'}(f(x) \neq y) - \min_{f'} \Pr_{(x,y) \sim D'}(f'(x) \neq y)
\]

For \( k = 2 \), the offset policy using \( f \) is \( f \).

Regret of policy \( f \) on \( D \):

\[
\text{reg}(f, D) = \mathbb{E}_{(x,\tilde{r}) \sim D} \left[ r_{f^*}(x) - r_f(x) \right]
\]

where \( f^* \) is the optimal policy.

**Binary Offset Theorem**

For all 2-action partial label problems \( D \) and binary classifiers \( f \):

\[
\text{reg}(f, D) \leq \text{reg}_{0/1}(f, D')
\]
Denoising for $k > 2$ arms

Use the same construction at each node. Each non-leaf predicts the best of a pair of winners from the previous round. Internal nodes only get an example if all leaf-ward nodes agree with the label.

Partial label policy on $x$: follow the chain of predictions from root to leaf, output the leaf.
Training on example \((x, 3)\)

\[
\begin{align*}
1 & \quad f_{1,2} \\
2 & \quad f_{1,2} \\
3 & \quad f_{3,4} \quad (x, \text{left}) \\
4 & \quad f_{3,4} \\
5 & \quad f_{5,6} \\
6 & \quad f_{5,6} \\
7 & \quad f_{5,6},7 \\
\end{align*}
\]
Training on example \((x, 3)\)

- \(f_{1,2}\)
- \(f_{3,4}\) \((x, \text{left})\)
- \(f_{\{1,2\},\{3,4\}}\) \((x, \text{right})\)
- Conditioned on \(f_{3,4}(x) = \text{left}\)

Note: Can be composed with either batch or online base learners
Training on example \((x, 3)\)

Note: Can be composed with either batch or online base learners.
\( D' = \) random binary problem according to chance that binary problem is fed an example under \( D \).
\( f = \) binary classifier that predicts based on \( x \) and the choice of binary problem according to \( D' \).
\( \pi_f = \) offset tree policy based on \( f \).

**Offset Tree Theorem**

For all \( k \)-choice partial label problems \( D \) and binary classifiers \( f \):

\[
\text{reg}(\pi_f, D) \leq (k - 1) \cdot \text{reg}_{0/1}(f, D')
\]

Lower bound: no reduction has a better regret analysis (holds for any value of \( \text{reg}_{0/1}(f, D') \)).
Other Solution Approaches

Argmax Regression

Important fact: the minimizer of squared error is the conditional mean.

1. Learn a regressor $f$ to predict $r_a$ given $(x, a)$.
2. Let $\pi_f(x) = \arg\max_a f(x, a)$

Importance-Weighted Classification Approach (Zadrozny’03)

Training:

1. For each $(x, a, r)$ example, create an importance weighted multiclass example $(x, a, rk)$.
2. Reduce importance weighted multiclass to binary using Costing and ECT for multiclass to binary reduction.

Testing: Make a multiclass prediction.
A Comparison of Approaches

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Policy Regret Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argmax Regression</td>
<td>$\sqrt{2k\text{reg}(s, D_{AR})}$</td>
</tr>
<tr>
<td>Importance-weighting Classification</td>
<td>$4k\text{reg}(b, D_{IWC})$</td>
</tr>
<tr>
<td>Offset Tree</td>
<td>$(k - 1)\text{reg}(b, D_{OT})$</td>
</tr>
</tbody>
</table>

How do you expect things to work, experimentally?
Offline Application, by simulation on UCI, comparing with Argmax and IW

![Graphs showing performance comparison between M5P, REPTree, and y=x in Offset Tree Regression and Importance weighting.](image-url)
Online Application, by simulation on RCV1, comparing with Banditron

![Error rate vs. number of examples graph](image1)

![Error rate vs. number of examples graph](image2)
Thanks!

Paper off my webpage → interactive learning
Further discussion at http://hunch.net