Exponential Computational Improvement
by Reduction

John Langford @ Microsoft Research

Computational Challenges Workshop, May 2
The Empirical Age

Speech Recognition
The Empirical Age

Speech Recognition
ImageNet
The Empirical Age

Speech Recognition
ImageNet
Deep Learning
The Empirical Age

Speech Recognition
ImageNet
Deep Learning
Neural Machine Translation
The Empirical Age

Speech Recognition
ImageNet
Deep Learning
Neural Machine Translation

What is a theorist to do?
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.
## Reduction Results

<table>
<thead>
<tr>
<th>Algo</th>
<th>$\epsilon$-greedy</th>
<th>Bagg</th>
<th>LinUCB</th>
<th>Online C.</th>
<th>Super.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss</td>
<td>0.095</td>
<td>0.059</td>
<td>0.128</td>
<td>0.053</td>
<td>0.051</td>
</tr>
<tr>
<td>Time</td>
<td>22</td>
<td>339</td>
<td>$212 \times 10^3$</td>
<td>17</td>
<td>6.9</td>
</tr>
</tbody>
</table>

Progressive validation loss on RCV1.
The Offline Contextual Bandit Setting

Given exploration data \((x, a, r, p)^*\)
Learn a good policy for choosing actions given context.
Offline Reduction Results

Classification Error

ecoli  glass  letter  optdigits  page-blocks  pendigits  satimage  vehicle  yeast
Agnostic Active Learning

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

1. The world produces some context $x \in X$
2. The learner predicts a label $\hat{y} \in Y$
3. The learner chooses to request a label or not. If label requested:
   1. observe $y$
   2. update learning algorithm

Goal: Compete with supervised learning using all labels while requesting as few as possible.
Logarithmic Time Prediction

Repeatedly

1. See $x$
2. Predict $\hat{y} \in \{1, \ldots, K\}$
3. See $y$
Logarithmic Time Prediction

Repeatedly

1. See $x$
2. Predict $\hat{y} \in \{1, \ldots, K\}$
3. See $y$

Goal: Find $h(x)$ minimizing error rate:

$$\Pr_{(x,y) \sim D} (h(x) \neq y)$$

with $h(x)$ in time $O(\log K)$.
Log-time prediction results

Statistical Performance

Delta Test Error (%) From OAA

-4
0
4
8
12

ALOI Imagenet ODP

LOMTree
Recall Tree
### Summary

<table>
<thead>
<tr>
<th>Problem</th>
<th>Learning Reductions</th>
<th>OCO</th>
<th>PAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB Explore</td>
<td>Yes</td>
<td>Sorta?</td>
<td>No</td>
</tr>
<tr>
<td>CB Learn</td>
<td>Yes</td>
<td>Sorta?</td>
<td>No</td>
</tr>
<tr>
<td>Agnostic Active</td>
<td>Yes</td>
<td>Sorta?</td>
<td>No</td>
</tr>
<tr>
<td>Log-time</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Outline

1. Why Reductions
2. What is a learning reduction?
3. Exponential Improvements
Learning Reduction Basics

Goal: minimize $\ell$ on $D$
Learning Reduction Basics

Goal: minimize $\ell$ on $D$

$\ell'$ optimizer

Transform $D$ into $D_R$

Transform $h$ with small $\ell'$ ($h, D_R$) into $R h$ with small $\ell$ ($R h, D$).

$h$ such that if $h$ does well on ($D_R, \ell'$), $R h$ is guaranteed to do well on ($D, \ell$).
Goal: minimize $\ell$ on $D$

Transform $D$ into $D_R$

A

$\ell'$ optimizer
Learning Reduction Basics

Goal: minimize $\ell$ on $D$

Transform $D$ into $D_R$

$\ell'$ optimizer

Transform $h$ with small $\ell'(h, D_R)$ into $R_h$ with small $\ell(R_h, D)$...
Goal: minimize $\ell$ on $D$

Transform $D$ into $D_R$

$\ell'$ optimizer

Transform $h$ with small $\ell'(h, D_R)$ into $R_h$ with small $\ell(R_h, D)$...

such that if $h$ does well on $(D_R, \ell')$, $R_h$ is guaranteed to do well on $(D, \ell)$. 
Error Reductions: the simplest possible

Prove: Small $\ell'$ error $\Rightarrow$ small $\ell$ error.
Error Reductions: the simplest possible

Prove: Small $\ell'$ error $\Rightarrow$ small $\ell$ error.

An issue: If $R$ introduces noise, small $\ell'$ not possible.
Prove: Small $\ell'$ error $\Rightarrow$ small $\ell$ error.

An issue: If $R$ introduces noise, small $\ell'$ not possible.

$\Rightarrow$ Must prove small $\ell'$ possible for nonvacuous statement.
Error Reductions: the simplest possible

Prove: Small $\ell'$ error $\Rightarrow$ small $\ell$ error.

An issue: If $R$ introduces noise, small $\ell'$ not possible.

$\Rightarrow$ Must prove small $\ell'$ possible for nonvacuous statement.
$\Rightarrow$ Error reductions weak for noisy problems.
Regret Reductions: Dealing with noise

Let $\text{reg}_{\ell, D} = \ell(h, D) - \min_{h'} \ell(h', D)$

Prove: Small $\text{reg}_{\ell, D'}$ $\Rightarrow$ small $\text{reg}_{\ell, D}$.

Note: $\min_{h'}$ is over all functions.

$\Rightarrow$ User is responsible for choosing right hypothesis space.

$\Rightarrow$ Unable to address information gathering.
Regret Reductions: Dealing with noise

Let $\text{reg}_{\ell, D} = \ell(h, D) - \min_{h'} \ell(h', D)$

Prove: Small $\text{reg}_{\ell', D'} \Rightarrow$ small $\text{reg}_{\ell, D}$. 

Note: $\min_{h'}$ is over all functions.

⇒ User is responsible for choosing right hypothesis space.

⇒ Unable to address information gathering.
Let $\text{reg}_{\ell,D} = \ell(h, D) - \min_{h'} \ell(h', D)$

Prove: Small $\text{reg}_{\ell',D'} \Rightarrow$ small $\text{reg}_{\ell,D}$.

Note: $\min_{h'}$ is over all functions.
Regret Reductions: Dealing with noise

Let \( \text{reg}_{\ell,D} = \ell(h, D) - \min_{h'} \ell(h', D) \)

Prove: Small \( \text{reg}_{\ell',D'} \) ⇒ small \( \text{reg}_{\ell,D} \).

Note: \( \min_{h'} \) is over all functions.
⇒ User is responsible for choosing right hypothesis space.
Let \( \text{reg}_{\ell,D} = \ell(h, D) - \min_{h'} \ell(h', D) \)

Prove: Small \( \text{reg}_{\ell',D'} \Rightarrow \) small \( \text{reg}_{\ell,D} \).

Note: \( \min_{h'} \) is over all functions.
\( \Rightarrow \) User is responsible for choosing right hypothesis space.
\( \Rightarrow \) Unable to address information gathering
Oracle Reductions: Information gathering

Assume an oracle which given samples $S$ returns $\arg \min_{h \in H} \ell'(h, S)$.
Oracle Reductions: Information gathering

Assume an oracle which given samples $S$ returns $\arg\min_{h \in H} \ell'(h, S)$

Prove: Oracle (approximately) works $\Rightarrow$
Computationally efficient small online regret on original problem.
Reductions $\Rightarrow$ modularity, code reuse $\Rightarrow$ good news for programming!

Vowpal Wabbit (http://hunch.net/vw) uses this systematically.
An Open Problem: $1K reward!

Conditional Probability Estimation

Distribution $D$ over $X \times Y$, where $Y = \{1, \ldots, k\}$.

Find a Probability estimator $h : X \times Y \to [0, 1]$ minimizing squared loss

$$\ell(h, D) = E_{(x,y) \sim D}[(h(y|x) - y)^2]$$
An Open Problem: $1K reward!

**Conditional Probability Estimation**

Distribution $D$ over $X \times Y$, where $Y = \{1, \ldots, k\}$.

Find a Probability estimator $h : X \times Y \to [0, 1]$ minimizing squared loss

$$\ell(h, D) = E_{(x,y) \sim D}[(h(y|x) - y)^2]$$

The problem: How can you do this in time $O(\log(k))$ with a constant regret ratio?