Weights and Measures: Prediction in the Era of Big Data

Lev Reyzin
UIC, Math Department
In a world of Big Data...

- Google/Yahoo! approaching 1B daily visitors.
- 3B yearly doctor’s visits in the US (records digitized).
- CERN currently storing > 100 petabytes of data.
- The NSA collects…. everything.
In a world of Big Data...

- Google/Yahoo! approaching 1B daily visitors. What ads to show?
- 3B yearly doctor’s visits in the US (records digitized). How to treat them? (w/o performing too many tests.)
- CERN currently storing > 100 petabytes of data. Which of the events contain an interesting signal?
- The NSA collects…. everything. Who is a terrorist?
Some Opportunities and Challenges

Can handle rich / interesting classes of functions.

Should be really fast (linear time, faster?)

Want to leverage all the data.

It may not fit on one machine.

Can finally predict accurately.

Must predict quickly! (Or otherwise limited.)
Weights and Measures

- **Weighing** – keep track of “performance” without needing to remember the history.

- **Measuring** – given the weights, how to properly “measure” them to determine the correct outcome?
Optimal and Efficient Contextual Bandits

“the world is a bandit problem”

with
Alina Beygelzimer, Miro Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lihong Li, Rob Schapire, Tong Zhang
What is the Bandit Setting?

- The name bandit refers to slot machines in a casino.

- You choose actions (e.g. which machine to play), one after another. These actions come with rewards.

- Goal is to minimize your regret – informally, how well you did compared to how well you could have done.

- Bound expected regret or bound regret w.h.p.
Regretting the dead

Nikos pointed out this New York Times article about poor clinical design killing people. For those of us who study learning from exploration information this is a reminder that low regret algorithms are particularly important, as regret in clinical trials is measured by patient deaths.

New Drugs Stir Debate on Rules of Clinical Trials

Two Cousins, Two Paths Thomas McLaughlin, left, was given a promising experimental drug to treat his lethal skin cancer in a medical trial; Brandon Ryan had to go without it.
Contextual Bandits [Auer et al. ’02]

context:

1

2

3

…

k

N experts/function
think of N >> K
Contextual Bandits [Auer et al. ’02]

context: $x_1$

1
2
3

$\vdots$

k

N experts/function think of $N \gg K$
Contextual Bandits [Auer et al. ’02]

context: $x_1$

1  yes

2

3

\vdots

k

N experts/function think of $N >> K$
Contextual Bandits [Auer et al. ’02]

context: $x_1$ $x_2$

1. yes

2.

3. yes

\[ \vdots \]

k.

N experts/function think of $N \gg K$
Contextual Bandits [Auer et al. ’02]

context: \( x_1 \) \( x_2 \) \( x_3 \) \( x_4 \) \( \ldots \) \( x_T \)

1. yes
2. no
3. yes
4. no

\( \vdots \)

k. no

\( N \) experts/function

think of \( N \gg K \)
Contextual Bandits [Auer et al. ’02]

context: $x_1 \ x_2 \ x_3 \ x_4 \ \ldots \ x_T$

regret = 0.05T

think of $N >> K$
Contextual Bandits [Auer et al. ’02]

context:  \(x_1\)  \(x_2\)  \(x_3\)  \(x_4\)  \(\ldots\)  \(x_T\)

1. the contexts & rewards can come
   from a distribution (stochastic) or be adversarial

2. The experts can be present (contextual) or not.

\(\vdots\)

\(k\)
harder than supervised (usual) learning:
In the bandit setting, we do not know the 
rewards of actions not taken.

many applications:
Medicine, ad auctions, finance, ...

exploration/exploitation dilemma:
exploit policies you’ve learned to be good?
explore policies you’re not sure about?
\( \Omega(TK \ln N)^{1/2} \) is a known lower bound on regret [Auer et al. '02] even in the stochastic setting.

Algorithms achieving \( \tilde{\Omega}(KT \text{ polylog } N)^{1/2} \) regret are said to be optimal.

greedily first exploring (acting randomly) then exploiting (following best policy) cannot be optimal. Optimal algorithms must be adaptive.
Two Types of Approaches

Upper Confidence Bounds

[Auer '02]

1)
2)

At every time step:
1) choose action with highest UCB
2) update confidence bound of the arm pulled.

EXP3 Exponential Weights

[Auer et al '02]

1)
2)

At every time step:
1) sample from distribution defined by weights (mixed w/ uniform)
2) “exponential” weight updates
## UCB vs EXP3: A Comparison

<table>
<thead>
<tr>
<th></th>
<th>UCB [Auer ’02]</th>
<th>EXP3 &amp; Friends [Auer et al. ’02]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td>Optimal in stochastic setting. Succeeds w.h.p.</td>
<td>Optimal for adversarial and stochastic settings. Adaptable to the contextual setting</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>Fails in adversarial setting. Not optimal in the contextual setting.</td>
<td>Cons Succeeds only in expectation.</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Optimal?</td>
<td>High Prob?</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>Exp4 [ACFS '02]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>epoch-geedy [LZ '07]</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Exp3.P[ACFS '02] UCB [Auer '00]</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Optimal?</td>
<td>High Prob?</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>Exp4 [ACFS '02]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>epoch-geedy [LZ '07]</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Exp3.P [ACFS '02]</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UCB [Auer '00]</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Exp4.P [BLLRS '11]</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Main Theorem [BLLRS ’11]: For any $\delta > 0$, w.p. at least $1 - \delta$, EXP4.P has regret at most $O(KT \ln (N/\delta))^{1/2}$ in adversarial contextual bandit setting.

Combines advantages of Exponential Weights and UCB.

• Optimal for both the stochastic and adversarial settings
• Works for contextual case (also non-contextual case)
• A high probability result
Ideas Behind Exp4.P
(all appeared in previous algorithms)

- **exponential weights**
  keep a weight on each expert that drops exponentially in the expert’s (estimated) performance

- **upper confidence bounds**
  adds an upper confidence bound on each expert’s estimated reward

- **ensuring exploration**
  make sure each action is taken with some minimum probability

- **importance sampling**
  give rare events more importance to keep estimates unbiased
key insights (on top of UCB/EXP)

1) exponential weights and confidence bounds “stack”
2) generalized Bernstein’s inequality for martingales
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp4 [ACFS '02]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>epoch-geedy [LZ '07]</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Exp3.P [ACFS '02]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>UCB [A '00]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp4.P [BLLRS '10]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Questions surround newly discovered ancient virus

The find raises the concern that climate change and global exploration could release dormant diseases. Buried in Siberia

Camera lowered into deep ocean trench finds unexpected creatures

Scientists have taken their first look into the previously unexplored New Hebrides deep-sea trench in the Pacific Ocean. At the The Verge

At White House, Israel's Netanyahu pushes back against Obama diplomacy

By Jeffrey Heller and Matt Spetalnick WASHINGTON (Reuters) - Prime Minister Benjamin Netanyahu bluntly told Barack Obama on Reuters
Evaluation

- We chose a policy class for which we could efficiently keep track of the weights.
  - Created 5 clusters, using user features.
  - Policies mapped clusters to article choices.
  - Ran on personalized news article recommendations for Yahoo! front page.

- We used a learning bucket on which we ran the algorithms and a deployment bucket on which we ran the greedy (best) learned policy.
Experiments

Reported estimated (normalized) click-through rates on front page news. Over 41M user visits. 253 total articles. 21 candidate articles per visit.

<table>
<thead>
<tr>
<th></th>
<th>EXP4P</th>
<th>EXP4</th>
<th>$\epsilon$-greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning eCTR</td>
<td>1.0525</td>
<td>1.0988</td>
<td>1.3829</td>
</tr>
<tr>
<td>Deployment eCTR</td>
<td><strong>1.6512</strong></td>
<td>1.5309</td>
<td>1.4290</td>
</tr>
</tbody>
</table>

Why does this work in practice? [McMahan ’11]
A General Efficient Algorithm?  
[DHKKL\textsc{RZ} '11]

\textbf{EXP4.P's regret grows only logarithmically with N.}

this suggests

We could compete with a huge set of policies!  
(e.g. $N=K^{100}$ becomes $10 \log^{1/2} K$ in the regret)

however

\textbf{Exp4.P explicitly “keeps track” of all policies. Reading in all recommendations, for large $N$, would take too long.}
Reduce to Supervised Learning! (Idea from [Langford-Zhang ’07])

- “Competing” with an exponentially large set of policies is commonplace in supervised learning.
- Recommendations of the policies/functions don’t need to be explicitly read when the policy class has structure!

\[
x_1 \quad x_2 \quad x_3 \quad x_4 \quad \ldots \quad x_5 \quad x_6
\]

Policy class \( \Pi \)

A good policy \( p \) in \( \Pi \)
Reduce to Supervised Learning!
(Idea from [Langford-Zhang '07])

- "Competing" with an exponentially large set of policies is commonplace in supervised learning.
- Recommendations of the policies/functions don’t need to be explicitly read when the policy class has structure!

\[ \begin{align*}
\text{Policy class } \Pi & \rightarrow \text{Oracle} \\
\text{Supervised Learning} & \rightarrow \text{Warning: NP-Hard in General} \\
X_1, X_2 & \rightarrow A \text{ good policy } p \in \Pi \\
X_3, X_4, \ldots, X_5, X_6 & \rightarrow
\end{align*} \]
context: \( x_1 \), \( x_2 \), \( x_3 \), ..., \( x_T \)

1  yes

2  no

3  ...

\( k \)  ...

\( N \) experts/functions
context: \( x_1 \) \( x_2 \) \( x_3 \) \( \ldots \) \( x_T \)

1 \( \rightarrow \) made-up data

2 \( \rightarrow \) N experts/functions

3 \( \rightarrow \) Supervised Learning Oracle

\[ \cdot \cdot \cdot \]

k \( \rightarrow \) $0.50$

$0.70$
Supervised Learning Oracle

context:  \( x_1 \), \( x_2 \), \( x_3 \), ..., \( x_T \)

made-up data

N experts/functions
Thm: [Dudik-Hsu-Kale-Karampatziakis-Langford-R-Zhang ’11]:
For any $\delta > 0$, w.p. at least $1 - \delta$, given access to a learning oracle, $R$-UCB has regret $O((KT \ln (NT/\delta))^{1/2}$ in the stochastic contextual bandit setting and runs in time $\text{poly}(K, T, \ln N)$. 
Thm: [Dudik-Hsu-Kale-Karampatziakis-Langford-R-Zhang ’11]: For any $\delta > 0$, w.p. at least $1 - \delta$, given access to a learning oracle, R-UCB has regret $O((KT \ln (NT/\delta))^{1/2})$ in the stochastic contextual bandit setting and runs in time $\text{poly}(K, T, \ln N)$.

if arms chosen among only “good” policies w/ variance < $2K$, we win
can prove this exists via a minimax theorem

↓

dthis can be softened to occasionally allow choosing of bad policies
via “randomized” upper confidence bounds

↓

creates a problem of how to choose arms as to satisfy the constraints expressed as convex optimization problem

↓

solvable by ellipsoid algorithm
can implement a separation oracle with the supervised learning oracle
Thm: [Dudik-Hsu-Kal-Karampatziakis-Langford-R-Zhang '11]:
For any $\delta > 0$, w.p. at least $1-\delta$, given access to a learning oracle, R-UCB has regret $O((KT\ln(N/T/\delta)^{1/2})$ in the stochastic contextual bandit setting and runs in time $\text{poly}(K,T,\ln N)$.

If arms chosen among only “good” policies with variance < $2K$, we win.

This can be softened to occasionally allow choosing of bad policies via “randomized” upper confidence bounds.

Creating a problem of how to choose arms as to satisfy the constraints expressed as convex optimization problems solvable by ellipsoid algorithm.

Big theoretical breakthrough!

But not practical.
(and needs stochastic assumption)
A research goal of mine: make it work in adversarial model.
Applying to Public Health

Consortium for Modeling and Analysis of Treatments and Interventions

c-mati.org
Distributed Learning
“the unreasonable effectiveness of data”

with
Jeff Cooper
Setting

data cannot fit on one machine
(each machine gets data from same distribution)
Goal

efficiently train a predictor
Distributing AdaBoost

What is boosting?

- AdaBoost [Freund-Schapire ’97] combines “weak” predictors into a strong predictor.
- Weak predictors are easy to design → lets us make strong predictors by “magic”!

Why distribute boosting?

- Probably the best “off the shelf” algorithm.
- Lends itself to being “distributed” but no adequate solution had been found.
- Is a weigh + measure approach!
weak classifiers = vertical or horizontal half-planes
Round 1

\[ h_1 \]

\[ \varepsilon_1 = 0.30 \]
\[ \alpha_1 = 0.42 \]

\[ D_2 \]
AdaBoost in Pictures (Slides from Schapire)

Round 2

$\epsilon_2 = 0.21$

$\alpha_2 = 0.65$
AdaBoost in Pictures (Slides from Schapire)

Round 3

$\epsilon_3 = 0.14$
$\alpha_3 = 0.92$
Final Classifier

\[ H_{\text{final}} = \text{sign} \left( 0.42 + 0.65 + 0.92 \right) \]
Bagging [Breiman ’96], iVoting [Breiman ’99], etc.

- Upside: easy to distribute for large data [Basilico et al. ’11]
- Downside: not true “boosting” algorithms and do not reach the error rates of AdaBoost

Boosting in Distributed PAC Model [Balcan et al. ’12]

- A harder model where communication complexity was studied.

Filterboost [Bradley-Schapire ’01], etc.

“The Distributed Boosting Alg.” [Lazarevic-Obradovic ’01]

- Upside: Best-yet practical distributed boosting algorithm
- Downside: unable to reach AdaBoost error rates, uses a lot of communication.
The Distributed Boosting Algorithm
[Lazarevic-Obradovic ’01]

- Data is split among K machines. The machines “boost” in parallel.

- On each round of boosting:
  1. Each machine sends its weak learner to all other machines.
  2. Each machine computes and sends the local error rate of the “majority learner” to all other machines.
  3. All machines update their local weights based on this info.

- At the end, every machine has full predictor.
DistBoost vs AdaBoost (if all data fit on one machine)
The Distributed Boosting Algorithm
[Lazarevic-Obradovic ‘01]

◆ Drawbacks:
  1. Uses lots of communication.
  2. Unable to recover full accuracy of boosting. Why? Each site overspecializes to its own data.

◆ Two different fixes [Cooper-R ‘14]:
  1. PreWeak: Smartly restrict weak learner selection.
  2. AdaSampling: Each machine selects its most informative data to send to a central processor.
Adaptive Sampling
[Cooper-R ’14]

- Main Idea: each machine uses AdaBoost to figure out which examples are “most informative”
  - Not always “hardest examples” – these could be noise.
  - Touches on margins theory

- These examples are sent to main processor, which boosts just on them.

- Rationale based on a game-theoretic view of boosting.
Empirical Results

The graph plots the performance of different ensemble methods against ensemble size. The x-axis represents the ensemble size, while the y-axis represents the particle value. Different methods are identified by various markers and line colors:

- **AdaSampling** (blue circles)
- **AdaBoost** (purple diamonds)
- **DIvote** (cyan triangles)
- **DistBoost** (green squares)
- **PreWeak** (red inverted triangles)

As the ensemble size increases, the particle values for all methods decrease, indicating improved performance. The maximum performance is achieved with a small ensemble size for all methods, with **DIvote** showing the best performance overall.
A Big-Data Example (Over 1M clicks/non-clicks)

<table>
<thead>
<tr>
<th>Error rates on Yahoo! click data.</th>
<th>Stumps</th>
<th>Depth 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost</td>
<td>36.48</td>
<td>35.02</td>
</tr>
<tr>
<td>DistBoost</td>
<td>37.23</td>
<td>35.54</td>
</tr>
<tr>
<td>DIvote</td>
<td>40.35</td>
<td>37.36</td>
</tr>
<tr>
<td>PreWeak</td>
<td>36.29</td>
<td>35.58</td>
</tr>
<tr>
<td>AdaSampling</td>
<td>37.14</td>
<td><strong>35.36</strong></td>
</tr>
<tr>
<td>AdaBoost (1/10)</td>
<td>36.76</td>
<td>35.68</td>
</tr>
</tbody>
</table>
Feature-Efficient Prediction

with
Yi Huang, Brian Powers
Feature-Efficient Prediction Examples

- **Medical testing**
  Want to predict what patients are sick with, but tests might be expensive or dangerous.

- **Displaying internet results**
  Want to give users the best results you can, but if you don’t give results within 300 milliseconds, users will leave.
Model

- Goal is to do supervised learning, using a limited number of features in test-time.
  - Given a budget on total cost: on each example, the learner must look at no more features than allowed by the budget.
  - Each feature has an associated cost.
  - Budget only limited in test data, not training.

- Predictors that do this are feature-efficient.
Lots of work on this problem

- **Sequential analysis**: when to stop sequential clinical trials. [Wald ’47] and [Chernoff ’72]

- **PAC learning** with incomplete features. [Ben-David-Dichterman ’93] and [Greiner et al. ’02]

- Robust prediction with **missing features**. [Globerson-Roweis ’06]

- Learning **linear functions** by few features [Cesa-Bianchi et al. ’10]

- Incorporating feature costs in CART impurity [Xu et al. ’12]

- **MDPs** for feature selection [He et al. ’13]
A “Weigh + Measure” Idea

[R ’11]

- An ensemble is usually a weighted vote of many simple rules.

- The simple rules are usually feature-efficient.

- Take a vote of only a few of the rules.
AdaBoostRS [R ’11]

Training: train AdaBoost (or any ensemble).

Prediction:

1. Sample the weak learners depending on their voting weights and feature costs.
2. Take a importance-weighted vote of the sampled weak learners.

Intuition:

If ensemble has strong preference, sampling will converge fast. If ensemble is split, who cares? (Thm resembles margin bound [Schapire et al. ’98])
Experiments with AdaBoostRS

On ocr17 dataset. x-axis is number of samples taken.
Room for Improvement

Can we improve by moving the optimization into training?

Turns out: yes, by a lot! [Huang-Powers-R ’14]

- Naïve idea: train AdaBoost until budget runs out
- Improvement: choose weak learners more wisely
AdaBoost \((S)\) where: \(S \subset X \times \{-1, +1\}\)

1: given: \((x_1, y_1), \ldots, (x_m, y_m) \in S\)
2: initialize \(D_1(i) = \frac{1}{m}\)
3: for \(t = 1, \ldots, T\) do
4: train base learner using distribution \(D_t\).
5: get \(h_t \in \mathcal{H} : X \rightarrow \{-1, +1\}\).
6: choose \(\alpha_t = \frac{1}{2} \ln \frac{1+\gamma_t}{1-\gamma_t}\), where \(\gamma_t = \sum_i D_t(i)y_i h_t(x_i)\).
7: update \(D_{t+1}(i) = D_t(i) \exp(\alpha_t y_i h_t(x_i))/Z_t\),
8: end for
9: output the final classifier \(H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)\)
AdaBoostBT(S,B,C) where: $S \subseteq X \times \{-1, +1\}$, $B > 0$, $C : [n] \rightarrow \mathbb{R}^+$

1: given: $(x_1, y_1), ..., (x_m, y_m) \in S$
2: initialize $D_1(i) = \frac{1}{m}$, $B_1 = B$
3: for $t = 1, \ldots, T$ do
4: train base learner using distribution $D_t$.
5: get $h_t \in \mathcal{H} : X \rightarrow \{-1, +1\}$.
6: if the total cost of the unpaid features of $h_t$ exceeds $B_t$ then
7: set $T = t - 1$ and end for
8: else set $B_{t+1}$ as $B_t$ minus the total cost of the unpaid features of $h_t$, marking them as paid
9: choose $\alpha_t = \frac{1}{2} \ln \frac{1 + \gamma_t}{1 - \gamma_t}$, where $\gamma_t = \sum_i D_t(i) y_i h_t(x_i)$.
10: update $D_{t+1}(i) = D_t(i) \exp(\alpha_t y_i h_t(x_i)) / Z_t$,
11: end for
12: output the final classifier $H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$
How to choose weak learner $h_t$?

Training error of AdaBoost is bounded by [Freund & Schapire '97]

$$\hat{Pr}[H(x) \neq y] \leq \prod_{t=1}^{T} \sqrt{1 - \gamma_t^2}$$

With budgets, we need to consider two effects:
- High edges make individual terms smaller
- Low costs allow for more terms in the product
Two Optimizations
[Huang-Powers-R '14]

First idea: assume all future rounds will behave like current. Leads to optimization

1) \[ h_t = \arg\min_{h \in \mathcal{H}} \left( (1 - \gamma_t(h)^2) \frac{1}{c(h)} \right) \]

Second idea: smoothed version of first.

2) \[ h_t = \arg\min_{h \in \mathcal{H}} \left( (1 - \gamma_t(h)^2) \frac{1}{(B-B_t)+c(h)} \right) \]
Experiments with costs \sim U(0,2)
Discussion
Summary

- Gave first optimal and (theoretically) efficient algorithm for contextual bandits by reducing to supervised learning.

- Gave a distributed ensemble algorithm with empirical performance matching boosting.

- Showed how to make any ensemble algorithm “feature efficient”.
Progress occurred in stages – eg. contextual algorithm was first made optimal but inefficient, then theoretically efficient, then practical (by others).

All these problems originate from practical concerns.

Each of these areas heavily used theory, but included practical elements, verified on large data. E.g. distributed boosting’s taking care of noisy examples.
Open Problems

- A practical, optimal contextual bandit algorithm that works in adversarial setting.

- Distributed and feature-efficient contextual bandits. (Reduction to supervised learning doesn’t preserve these properties.)

- Better models for trading off error / prediction time.

- Many others...
The End

Thank you! Any Questions?