Verified Learning Without Regret

A Mechanized Proof of the Multiplicative Weights Update Algorithm

Sam Merten
(PhD)

Gordon Stewart
Assistant Professor, EECS
Ohio University

Alex Bagnall
(MSc)
In 2014 alone...

- April 2014: HeartBleed OpenSSL bug
  - buffer overread due to missing bounds check
  - 17% of servers running TLS affected
- September 2014: Shellshock
  - Bash – unintended command execution
  - undiscovered for 25 years (!)
- October 2014: POODLE
  - TLS: for interoperability, fall back to SSL 3.0
  - ... exposing a padding oracle attack

2000s:

Toyota Unintended Acceleration

- lives lost...probably due to software
- $1.2b settlement
What Do We Do About It?

Expressivity

User Interaction

Interactive Theorem Proving

```
\[ \Gamma \vdash e_1 + e_2 : \tau \]
```

NuSMV

Software Model Checking

SPARK Toolset

Static Analysis

Type Systems

Astrée

MY WEAPON OF CHOICE

HOL
Interactive Theorem Prover

my weapon of choice
Interactive Theorem Proving

SPECIFICATION
\[ \forall x \forall P \exists z \ldots \]
Dependent Type Theory

DEF f (x : t) ... DEF g (x : t) ...

PROGRAM UNDER PROOF

machine-checked proof certificate

Coq IDE

Coq Proof Checker

The Coq Proof Assistant
http://coq.inria.fr

Yes 😊

No 😞
Trusted Computing Base

Which of these pieces do we need to trust?

DEF f (x : t) ...
DEF g (x : t) ...

∀x∀P∃z.... Dependent Type Theory

SPECIFICATION

machine-checked proof certificate

Coq IDE

Coq Proof Checker

The Coq Proof Assistant
http://coq.inria.fr

Yes😊
No😃
Theorem Proving In Practice

**Proof Script**

Used to construct independently checkable proof object

**Proof Window**

- current proof state
- including hypotheses & goals

---

Hypothesis (HAg0 : 0 < A).

(** The bound is proved assuming there exist real numbers [A] and [B] such that for any state [t], [Phi t] is bounded on the left by [Cost t / A] and bounded on the right by [B * Cost t]. *)

Hypothesis AB_bound_Phi :
forall t : sT, Cost t / A <= Phi t <= B * Cost t.

(** Under the conditions stated above, the Price of Stability of any potential game is at most [A * B]. (For games in which the PNE is unique, this bound gives a bound on the Price of Anarchy as well.) *)

Lemma PoS_bounded (t0 : sT) (PNE_t0 : PNE t0) :
PoS t0 <= A * B.

Proof.

set (tn := Phi_minimizer t0).

generalize (minimal_Phi_minimizer t0); move/forallP=> HtN.
case: (andP (AB_bound_Phi tn))=> H3 H4; rewrite /PoS.
set (tStar := arg_min optimal Cost t0).

move: (HtN tStar) => H5.
case: (andP (AB_bound_Phi tStar)) => H6 H7.
rewrite ler_pdvMar; last by apply: Cost_pos.
apply: ler_trans.

1 subgoal, subgoal 1 (ID 54)

T : game
X0 : Moveable T
X : Potential
Cost_pos : forall t : {ffun 'L'(numplayers T) => T}, 0 < Cost t
A, B : rty
HAg0 : 0 < A
AB_bound_Phi : forall t : {ffun 'L'(numplayers T) => T},
Cost t / A <= Phi t <= B * Cost t
t0 : {ffun 'L'(numplayers T) => T}
PNE_t0 : PNE t0
tn := Phi_minimizer t0
PNE_t0 : PNE t0
PoS t0 <= A * B
MULTIPLICATIVE WEIGHTS UPDATE (MWU)
Learning on-line, in uncertain environments
(For the remainder, I’ll assume costs in range [0, 1].)

<table>
<thead>
<tr>
<th>Actions</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>0.5</td>
</tr>
<tr>
<td>★</td>
<td>0.1</td>
</tr>
<tr>
<td>○</td>
<td>0.2</td>
</tr>
<tr>
<td>▽</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Round 1: **AGENT** pays 0.2

<table>
<thead>
<tr>
<th>Actions</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>0.0</td>
</tr>
<tr>
<td>★</td>
<td>0.7</td>
</tr>
<tr>
<td>○</td>
<td>0.1</td>
</tr>
<tr>
<td>▽</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Round 2: **AGENT** pays 0.7

**AGENT** pays 0.9 total
Regret

A learning algorithm is **bounded regret** if it has constant expected cost wrt. the best fixed action, as the number of iterations $T \to \infty$.

$$Regret(A) := \mathbf{E}[C_{tot}(A)] - \min_a C_{tot}(a)$$

**AGENT** pays 0.9 total

$$C_{tot}(\red) = 0.3$$

Regret $= 0.9 - 0.3 = 0.6$
Why (Verify) Regret?

Bounded-regret algorithms: natural distributed execution semantics yielding approximate equilibria

\[ s_{rc_1} \rightarrow d_{st_1} \]
\[ s_{rc_2} \rightarrow d_{st_2} \]
\[ s_{rc_N} \rightarrow d_{st_N} \]

AGENT 1: Regret At Most \( \varepsilon \)
AGENT 2: Regret At Most \( \varepsilon \)
AGENT N: Regret At Most \( \varepsilon \)

\( \varepsilon \)-approximate CCE approximates optimal configuration

DISTRIBUTED ROUTING GAME
The MWU Algorithm

- Associate to each action $a$ weight $w(a)$
- Choose actions by drawing from the distribution

$$p(a) = \frac{w(a)}{\sum_b w(b)}$$

- Update weights according to the following rule

$$w^{i+1}(a) = w^i(a) \times (1 - \epsilon \times c^i(a))$$

PARAMETER $\epsilon \in (0, \frac{1}{2}]$

MORE $\rightarrow$ LESS EXPLORATION
• “Combining Expert Advice”
• Winnow
  – an algorithm for learning linear classifiers
  – [Littlestone ‘88]

• Weighted Majority Hedging
  – Exponential update rule:
    \[ w^{i+1}(a) = w^i(a) \times \left( 1 - \epsilon^c_i(a) \right) \]

• AdaBoost / Boosting
  – [Freund and Schapire ‘97]
Round 1: 

\[
p(\text{Agent}) = \frac{w^1(\text{Agent})}{4} = \frac{1}{4}
\]

Round 2: 

\[
p(\text{Environment}) = \frac{w^2(\text{Environment})}{3.5} = 0.27
\]

\[
w^{i+1}(a) = w^i(a) \times (1 - \epsilon \times c^i(a))
\]

\[\epsilon = \frac{1}{2}\]
\[ \epsilon = \frac{1}{2} \]

### Actions

<table>
<thead>
<tr>
<th>( w^2 )</th>
<th>0.75</th>
<th>0.95</th>
<th>0.9</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Costs2</td>
<td>0.0</td>
<td>0.7</td>
<td>0.1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

**Round 2:** AGENT pays 0.7

### Actions

<table>
<thead>
<tr>
<th>( w^3 )</th>
<th>0.75</th>
<th>0.62</th>
<th>0.86</th>
<th>0.81</th>
</tr>
</thead>
<tbody>
<tr>
<td>Costs3</td>
<td>0.1</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**Round 3:** AGENT pays 0.1

\[ p(\star) = \frac{w^2(\star)}{3.5} = 0.27 \]

\[ p(\square) = \frac{w^3(\square)}{3.04} = 0.25 \]

**Penalized**
### Environment

\[ \epsilon = \frac{1}{2} \]

<table>
<thead>
<tr>
<th>Actions</th>
<th>( w^3 )</th>
<th>Costs3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.75</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.62</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.86</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.81</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**Round 3:** AGENT pays 0.2

### Agent

<table>
<thead>
<tr>
<th>Actions</th>
<th>( w^{10} )</th>
<th>Costs10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.52</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.41</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.11</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**Round 10:** AGENT pays 0.2

\[ p(\text{\textcolor{blue}{\textbullet}}) = \frac{w^3(3.04)}{3.04} = 0.25 \]

\[ p(\text{\textcolor{red}{\textbullet}}) = \frac{w^{10}(1.24)}{1.24} = 0.33 \]
**Theorem:** MWU is bounded regret.

\[
\frac{\mathbb{E}[C_{tot}(MWU)] - \min_a C_{tot}(a)}{T} \leq \epsilon + \frac{\ln |A|}{\epsilon T}
\]

**Proof:** Potential function \( \Gamma^i = \sum_a w^i(a) \)

**Corollary:**

\[
\frac{c \ln |A|}{\epsilon} \text{ steps to achieve } \epsilon + \frac{1}{c} \text{ per-step regret.}
\]
PART I
• Theorem Proving
• MWU By Example
• Bounded-Regret Learning & Why
• MWU Is Bounded Regret

PART II
• Formalizing MWU
• Verifying Regret

VERIFIED MWU
## MWU Formalized

### The Coq Proof Assistant

#### Core Files

<table>
<thead>
<tr>
<th>spec</th>
<th>proof</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>349</td>
<td>754</td>
<td>31 weights.v</td>
</tr>
<tr>
<td>738</td>
<td>932</td>
<td>75 weightslang.v</td>
</tr>
<tr>
<td>370</td>
<td>831</td>
<td>69 weightsextract.v</td>
</tr>
<tr>
<td>1457</td>
<td>2517</td>
<td>175 total</td>
</tr>
</tbody>
</table>

#### Auxiliary Files

<table>
<thead>
<tr>
<th>spec</th>
<th>proof</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>269</td>
<td>1062</td>
<td>17 numerics.v</td>
</tr>
<tr>
<td>75</td>
<td>3</td>
<td>1 strings.v</td>
</tr>
<tr>
<td>110</td>
<td>80</td>
<td>5 dist.v</td>
</tr>
<tr>
<td>60</td>
<td>109</td>
<td>11 extrema.v</td>
</tr>
<tr>
<td>60</td>
<td>75</td>
<td>1 bigops.v</td>
</tr>
<tr>
<td>42</td>
<td>475</td>
<td>28 neps_exp_le.v</td>
</tr>
<tr>
<td>616</td>
<td>1804</td>
<td>63 total</td>
</tr>
</tbody>
</table>

**TOTAL:** 6632 LOC
Theorem: MWU Is Bounded Regret

**Formal:**

Notation \( \text{astar} := (\text{best_action a0 cs}). \)

Notation \( \text{OPT} := (\sum_{c \leq \text{cs}} c \astar). \)

Notation \( \text{OPTR} := (\text{rat_to_R OPT}). \)

... more definitions and notations ...

**Lemma** \( \text{perstep_weights_noregret} : \)

\[
\frac{(\exp\text{CostsR} - \text{OPTR})}{T} \leq \epsilon + \frac{\ln |A|}{\epsilon T}\%
\]

**Informal:**

\[
\left( E[C_{\text{tot}}(\text{MWU})] - \min_{a} C_{\text{tot}}(a) \right) / T \leq \epsilon + \frac{\ln |A|}{\epsilon T}
\]

**EXPECTED TOTAL COST OF MWU**

**COST OF BEST FIXED ACTION**

**NUMBER OF STEPS**

**SIZE OF ACTION SPACE**
A Hierarchy of Refinements

High-Level Functional Specification

Definition update_weights (w:weights) (c:costs) : weights := finfun (fun a : A => w a * (1 - eps * c a)).

MWU DSL
Binary Arith. Operations
  b ::= + | - | *
Expressions
e ::= q
  | eps
  | e b e | ...
Commands
c ::= skip
  | update f | ...

Operational Semantics
\[ \vdash c, \sigma \Rightarrow c', \sigma' \]

MWU DSL

Operational Semantics

Fixpoint interp (c:com A.t) (s:cstate) : option cstate := match c with ... end.

Executable Interpreter

Even moderate-size proof developments (just like moderate-size software developments!) benefit from abstraction
Definition \textit{update weights} \((w:\text{weights}) (c:\text{costs}) : \text{weights} := \text{finfun (fun } a : A \Rightarrow w a \times (1 - \text{eps} \times c a))\).

Data Refinement

weights = A.t \to \text{rat}

Sweights s : M.t Q

Efficient RBTree
Simplified distributed routing game with
- 5 players
- 50 iterations
- mean and std. dev. over 10 trials
- $\epsilon = \frac{1}{4}$
• **Bandit Model**
  – revealing cost of all actions at each step imposes high communication overhead
  – assume, instead, only chosen action’s cost is revealed
  – slightly more complex algorithms, slightly worse bounds, but perhaps faster in practice?

• **Linear Programming**
  – Verified MWU as a verified approximate LP solver!

• **AdaBoost** [Freund & Schapire ‘97]
  • [Arora et al., ‘12]
    – a treasure trove of additional connections!
Conclusion

Verified Multiplicative Weights Update:
- **Machine-verified implementation** of a simple yet powerful algorithm for “combining expert advice”
- **Proof strategy**: layered program refinements, from executable MWU to high-level specification

Short/Medium Term Plans:
- From bounded regret to whole-system performance guarantees
- with applications to distributed systems (e.g., distributed routing, load balancing, etc.)
QUESTIONS?

References


Thank You!