Certified Multiplicative Weights Update

 Verified Learning Without Regret

Gordon Stewart
HCSS – May 10, 2017

Sam Merten (PhD)

Alex Bagnall (MSc)
Hard Problems in Assurance for AI

Specification

When is, e.g., a convolutional neural network for image classification “correct”?

– Performance on test set?
– Performance in real world?
– Proof of generalizability to some well-specified distribution over inputs?

Resilience to Adversarial Input

Practitioners often (incorrectly) assume that test set accurately models inputs in the field.

– But quite easy to generate adversarial NN inputs that cause misclassification with high confidence [Goodfellow et al., ‘14]
This is not the number $\pi$ ...

- 28*28 (784) input features
- 1 hidden layer with 256 neurons, rectified linear unit (ReLU) activation
- softmax output 97.97% accuracy on original test data (MNIST)

Confidence

Predicted

Test

<table>
<thead>
<tr>
<th>Conf.</th>
<th>Predicted</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>.97</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>.99</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>.99</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>.99</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>.81</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>.99</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Adversarial (Fast Gradient Sign)

sharpLeft: (Confidence = 0.987654)
“This is the essence of intuitive heuristics: when faced with a difficult question, we often answer an easier one instead, usually without noticing the substitution.”

— Daniel Kahneman, Thinking, Fast and Slow
Online Learning in Adversarial Environments

Wilmington, DE

Annapolis, MD

Broad application to CS, algorithmic game theory, mechanism design, economics

Well-studied algorithms, strong theoretical guarantees

Amenable to conventional methods in verification (interactive theorem provers, proof by refinement)

We know when we get it right (straightforward correctness specifications, e.g., regret)

Online Learning in Adversarial Environments
Online Learning in Adversarial Environments

**Agent**

Round 1: **AGENT** pays 0.7

Round 2: **AGENT** pays 0.2

**Environment**

*Agent pays 0.9 total*
Online Learning in Adversarial Environments

Agents may randomize over set of possible actions [mixed strategies]

Agent

<table>
<thead>
<tr>
<th>Mixed Strategy</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>US 95</td>
<td>0.4</td>
</tr>
<tr>
<td>MARYLAND 301</td>
<td>0.6</td>
</tr>
<tr>
<td>0.1</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Environment

<table>
<thead>
<tr>
<th>Mixed Strategy</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>US 95</td>
<td>0.5</td>
</tr>
<tr>
<td>MARYLAND 301</td>
<td>0.5</td>
</tr>
<tr>
<td>0.2</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Good Learners Have No Regret

Regret*(A) := \( \sum_{t=1}^{T} E[C_t(A)] - \min_{a} \sum_{t=1}^{T} C_t(a) \) / T

\( E[C_{tot}(A)] \)

How well adaptive alg. A performs (in expectation)

\( \min_{a} \sum_{t=1}^{T} C_t(a) \)

against fixed action \( a \) with lowest total cost

\( \min_a C_{tot}(a) \)

A is No Regret if Regret(A) approaches 0 as \( T \to \infty \).

*External Regret
Good Learners Have No Regret

\[
\text{Regret}^*(A) := \left( \sum_{t=1}^{T} E[C_t(A)] \right) - \min_a \left( \sum_{t=1}^{T} C_t(a) \right) / T
\]

\[
E[C_{tot}(A)] \quad \text{How well adaptive alg. } A \text{ performs (in expectation)}
\]

\[
\text{Regret} = 0.9 - 0.3
\]

\* \text{External Regret}

\text{Optimal} \quad \text{Agent}

**Round 1:** Agent pays 0.7

**Actions**

**Costs**

\text{US 95}

\text{MARYLAND 301}

\text{Round 2:** Agent pays 0.2

**Actions**

**Costs**

\text{US 95}

\text{MARYLAND 301}

\text{Agent is No Regret if Regret}(A) \text{ approaches 0 as } T \to \infty.
Inputs:
- A set of fixed decision rules / classifiers / “experts”
- Sequence of points with unknown labels \{red, white\}

No-Regret Algorithm Outputs:
- *online* classification performance on input sequence nearly as good as best fixed decision rule.
No-Regret Game Dynamics

No-regret algorithms: natural *distributed* execution model for games, converging to *approximate equilibria*.

At time $T$, each AGENT has regret at most $\epsilon$.

**Intuition:**

Unilateral deviation from $\epsilon$-regret algorithm $A$ to any fixed action $a$

$$E[C_i(A, \ldots)] \leq E[C_i(a, \ldots)] + \epsilon$$

allows agent to gain at most $\epsilon$.

*Approximate Coarse Correlated Equilibria*
Multiplicative Weights (MW)

- Associate to each action \( a \in ACT \) weight \( w(a) (=1) \)
- Choose actions by drawing from the distribution
  \[
p(a) = \frac{w(a)}{\sum_b w(b)}
\]
- Adversary sends cost vector
  \[
c : A \to [-1,1]
\]
- Update weights according to the following rule
  \[
w^{i+1}(a) = w^i(a) \times (1 - \epsilon \times c^i(a))
\]

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

Exploration vs. Exploitation
**Theorem:** MW is no regret.

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

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

**Corollary:** \( \frac{(E[C_{tot}(MW)] - \min_a C_{tot}(a))}{T} \leq 2 \sqrt{\frac{\ln |A|}{T}} \)

Letting \( 0 < \epsilon = \sqrt{\frac{\ln |A|}{T}} \leq \frac{1}{2} \)
• “Combining Expert Advice”

• Winnow
  – an algorithm for learning linear classifiers
  – [Littlestone ‘88]

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

• AdaBoost / Boosting
  – [Freund and Schapire ‘97]
PART I
• Assurance for AI
• No-Regret Learning & Why
• Multiplicative Weights (MW)

PART II
• Formalizing MW
• Verifying Regret

VERIFIED MW
### Core Files

<table>
<thead>
<tr>
<th>spec</th>
<th>proof</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>390</td>
<td>939</td>
<td>35 weights.v</td>
</tr>
<tr>
<td>842</td>
<td>1073</td>
<td>80 weightslang.v</td>
</tr>
<tr>
<td>322</td>
<td>892</td>
<td>68 weightsextract.v</td>
</tr>
<tr>
<td>1554</td>
<td>2904</td>
<td>183 total</td>
</tr>
</tbody>
</table>

### Auxiliary Files

<table>
<thead>
<tr>
<th>spec</th>
<th>proof</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>1168</td>
<td>20 numerics.v</td>
</tr>
<tr>
<td>217</td>
<td>1015</td>
<td>31 dyadic.v</td>
</tr>
<tr>
<td>144</td>
<td>9</td>
<td>1 strings.v</td>
</tr>
<tr>
<td>117</td>
<td>87</td>
<td>3 dist.v</td>
</tr>
<tr>
<td>60</td>
<td>109</td>
<td>11 extrema.v</td>
</tr>
<tr>
<td>77</td>
<td>111</td>
<td>3 bigops.v</td>
</tr>
<tr>
<td>915</td>
<td>2499</td>
<td>69 total</td>
</tr>
</tbody>
</table>

**TOTAL: 7862 LOC**
Theorem: MW Is Bounded Regret

**Formal:**

Notation $\text{astar} := (\text{best	extunderscore action } a0 \text{ cs}).$

Notation $\text{OPT} := (\sum_{c < -cs} c \text{ astar}).$

Notation $\text{OPTR} := (\text{rat	extunderscore to	extunderscore R OPT}).$

... more definitions and notations ...

**Lemma** $\text{perstep	extunderscore weights	extunderscore noregret}:
\[
\frac{\text{expCostsR} - \text{OPTR}}{T} \leq \epsilon + \ln |A| / (\epsilon T)%R.
\]

**Informal:**
\[
\frac{\mathbb{E}[C_{tot}(MW)] - \min_{a} C_{tot}(a)}{T} \leq \epsilon + \frac{\ln |A|}{\epsilon T}
\]

- cumulative expected cost of MW
- 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)).

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

Operational Semantics
⊢ c, σ ⇒ c', σ'

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
**Update Weights**

**Definition** `update_weights` (w:weights) (c:costs) : weights :=

\[
\text{finfun} (\text{fun } a : A \to w a \ast (1 - \text{eps} \ast c a))
\]

**REFINES**

**Definition** `update_weights` (f : A.t → expr A.t) (s : cstate) :

\[
\text{option } (M.t \ D) := \\
M.\text{fold} (\text{fun } a _ \_ \text{acc} => \\
\text{match } \text{acc} \text{ with} \\
| \text{None} => \text{None} \\
| \text{Some } \text{acc}' => \\
\text{match } \text{evalc} (f a) \text{ s with} \\
| \text{None} => \text{None} \\
| \text{Some } q => \\
\text{match } 0 \ ?= q \text{ with} \\
| \text{Lt} => \text{Some } (M.\text{add} a (Q\text{red } q) \text{ acc}') \\
| _ => \text{None} \\
\text{end end end end}) \\
(S\text{Weights } s) \\
\text{(Some } (M.\text{empty } Q))
\]

**Data Refinement**

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

**REFINES**

\[\text{Sweights } s : M.t \ D\]

Efficient AVLTree over dyadic rational weights
Specifying the Environment

```
Class ClientOracle {A} :=
  mkOracle { T : Type (* oracle private state *)
    ; oracle_init_state : T
    ; oracle_chanty : Type
    ; oracle_bogus_chan : oracle_chanty
  oracle_recv : T -> oracle_chanty -> (list (A*D) * T)
  oracle_send : T -> list (A*D) -> (oracle_chanty * T)
  oracle_recv_ok : forall st ch a,
    exists d,
      [\ In (a,d) (oracle_recv st ch).1
        , Dle (-D1) d & Dle d D1]
  ; oracle_recv_nodup : forall st ch,
    NoDupA (fun p q => p.1 = q.1) (oracle_recv st ch).1
}.
```
Multi-Agent Distributed Routing
- 5 players
- 50 iterations
- 10 trials
- $\epsilon = 0.1325$

Environment oracle:
Coq server + extraction to OCaml network primitives

Verified Per-Step Regret Bound

~2x reduction in cost

Affine Latencies

10x+10

10x+10

Multi-Agent

Experiment: Multi-Agent Affine Routing

AGENT 1: Regret $\leq \epsilon$

AGENT 2: Regret $\leq \epsilon$

AGENT 5: Regret $\leq \epsilon$

AGENT 1:

Regret $\leq \epsilon$

AGENT 2:

Regret $\leq \epsilon$

AGENT 5:

Regret $\leq \epsilon$

Multi-Agent

Distributed Routing

• 5 players
• 50 iterations
• 10 trials
• $\epsilon = 0.1325$
• **Linear Programming**
  – Verified MW as a verified LP solver

• **AdaBoost** [Freund & Schapire ‘97]
  – From weak to strong learners

• **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?

• [Arora et al., ‘12]
  – a treasure trove of additional connections!
Machine-verified implementation of a simple yet powerful algorithm for online learning in adversarial environments.

Proof strategy: layered program refinements, from high-level specification to executable MW.

Freely available online: https://github.com/gstew5/cage
References


