Learning Theory of Optimal Decision Making

Part III: Online learning in adversarial environments

Csaba Szepesvári[1]

[1]Department of Computing Science
University of Alberta

Machine Learning Summer School, Ile de Re, France, 2008

with thanks to: RLAI group, SZTAKI group

Jean-Yves Audibert, Remi Munos

OUTLINE

1. HIGH LEVEL OVERVIEW OF THE TALKS
2. MOTIVATION
   - What is it?
   - Why should we care?
   - Halving: Find the perfect expert! (0/1 loss)
   - No perfect expert? (0/1 loss)
   - Predicting Continuous Outcomes
3. DISCRETE PREDICTION PROBLEMS
   - Randomized forecasters
   - Weighted Average Forecaster
   - Follow the perturbed leader
4. TRACKING THE BEST EXPERT
   - Fixed share forecaster
   - Variable-share forecaster
   - Other large classes of experts
5. NON-STOCHASTIC BANDIT PROBLEMS
   - Exp3.P: An algorithm for non-stochastic bandit problems
6. CONCLUSIONS

Outline

1. High level overview of the talks
2. Motivation
   - What is it?
   - Why should we care?
   - Halving: Find the perfect expert! (0/1 loss)
   - No perfect expert? (0/1 loss)
   - Predicting Continuous Outcomes
3. Discrete prediction problems
   - Randomized forecasters
   - Weighted Average Forecaster
   - Follow the perturbed leader
4. Tracking the best expert
   - Fixed share forecaster
   - Variable-share forecaster
   - Other large classes of experts
5. Non-stochastic bandit problems
   - Exp3.P: An algorithm for non-stochastic bandit problems
6. Conclusions

OUTLINE

1. HIGH LEVEL OVERVIEW OF THE TALKS
2. MOTIVATION
   - What is it?
   - Why should we care?
   - Halving: Find the perfect expert! (0/1 loss)
   - No perfect expert? (0/1 loss)
   - Predicting Continuous Outcomes
3. DISCRETE PREDICTION PROBLEMS
   - Randomized forecasters
   - Weighted Average Forecaster
   - Follow the perturbed leader
4. TRACKING THE BEST EXPERT
   - Fixed share forecaster
   - Variable-share forecaster
   - Other large classes of experts
5. NON-STOCHASTIC BANDIT PROBLEMS
   - Exp3.P: An algorithm for non-stochastic bandit problems
6. CONCLUSIONS

Outline

Outline

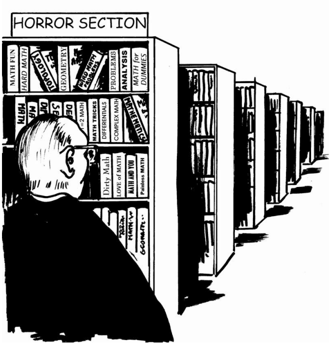
Outline

1. High level overview of the talks
2. Motivation
   - What is it?
   - Why should we care?
   - Halving: Find the perfect expert! (0/1 loss)
   - No perfect expert? (0/1 loss)
   - Predicting Continuous Outcomes
3. Discrete prediction problems
   - Randomized forecasters
   - Weighted Average Forecaster
   - Follow the perturbed leader
4. Tracking the best expert
   - Fixed share forecaster
   - Variable-share forecaster
   - Other large classes of experts
5. Non-stochastic bandit problems
   - Exp3.P: An algorithm for non-stochastic bandit problems
6. Conclusions

RL
AI

HIGH LEVEL OVERVIEW OF THE TALKS

- Day 1: Online learning in stochastic environments
- Day 2: Batch learning in Markovian Decision Processes
- Day 3: **Online learning in adversarial environments**



Outline

What is it?

Concepts: Agent, Environment, sensations, actions, rewards
Time: $t = 1, 2, \ldots$

Protocol of learning

1. Agent senses x_t coming from Environment
2. Agent sends prediction $\hat{p}_t$ to Environment
3. Environment generates outcome y_t
4. Agent receives loss $\ell_t = \ell(\hat{p}_t, y_t)$ from Environment
5. $t = t + 1$, go to Step 1

Goal: $\sum_{t=1}^{T} \ell_t \to \min$

What is it?

Concepts: Agent, Environment, sensations, actions, rewards
Time: $t = 1, 2, \ldots$

Protocol of learning

1. Agent senses x_t coming from Environment
2. Agent sends prediction $\hat{p}_t$ to Environment
3. Environment generates outcome y_t
4. Agent receives loss $\ell_t = \ell(\hat{p}_t, y_t)$ from Environment
5. $t := t + 1$, go to Step 1

Goal: $\sum_{t=1}^{T} \ell_t \to \min$

What is it?

Concepts: Agent, Environment, sensations, actions, rewards
Time: $t = 1, 2, \ldots$

Protocol of learning

1. Agent senses x_t coming from Environment
2. Agent sends prediction $\hat{p}_t$ to Environment
3. Environment generates outcome y_t
4. Agent receives loss $\ell_t = \ell(\hat{p}_t, y_t)$ from Environment
5. $t := t + 1$, go to Step 1

Goal: $\sum_{t=1}^{T} \ell_t \to \min$

Concepts: Agent, Environment, sensations, actions, rewards
Time: $t = 1, 2, \ldots$

Protocol of learning

1. Agent senses x_t coming from Environment
2. Agent sends prediction $\hat{p}_t$ to Environment
3. Environment generates outcome y_t
4. Agent receives loss $\ell_t = \ell(\hat{p}_t, y_t)$ from Environment
5. $t := t + 1$, go to Step 1

Goal: $\sum_{t=1}^{T} \ell_t \to \min$

WHAT IS IT?

Concepts: Agent, Environment, sensations, actions, rewards
Time: $t = 1, 2, \ldots$

PROTOCOL OF LEARNING

1. Agent senses x_t coming from Environment
2. Agent sends prediction $\hat{p}_t$ to Environment
3. Environment generates outcome y_t
4. Agent receives loss $\ell_t = \ell(\hat{p}_t, y_t)$ from Environment
5. $t := t + 1$, go to Step 1

Goal: $\sum_{t=1}^{T} \ell_t \to \min$

What is it?

Concepts: Agent, Environment, sensations, actions, rewards
Time: $t = 1, 2, \ldots$

Protocol of learning

1. Agent senses x_t coming from Environment
2. Agent sends prediction $\hat{p}_t$ to Environment
3. Environment generates outcome y_t
4. Agent receives loss $\ell_t = \ell(\hat{p}_t, y_t)$ from Environment
5. $t := t + 1$, go to Step 1

Goal: $\sum_{t=1}^{T} \ell_t \to \min$

WHAT IS IT?

Concepts: Agent, Environment, sensations, actions, rewards
Time: $t = 1, 2, \ldots$

PROTOCOL OF LEARNING

1. Agent senses x_t coming from Environment
2. Agent sends prediction $\hat{p}_t$ to Environment
3. Environment generates outcome y_t
4. Agent receives loss $\ell_t = \ell(\hat{p}_t, y_t)$ from Environment
5. $t := t + 1$, go to Step 1

Goal: $\sum_{t=1}^{T} \ell_t \to \min$

What is it?

Concepts: Agent, Environment, sensations, actions, rewards
Time: $t = 1, 2, \ldots$

Protocol of learning

1. Agent senses x_t coming from Environment
2. Agent sends prediction $\hat{p}_t$ to Environment
3. Environment generates outcome y_t
4. Agent receives loss $\ell_t = \ell(\hat{p}_t, y_t)$ from Environment
5. $t := t + 1$, go to Step 1

Goal: $\sum_{t=1}^{T} \ell_t \to \min$

OUTLINE

Why should we care?

- No assumptions about the Environment!
- We compare the return with that of algorithms from a set: experts

  "Competitive analysis"
- Results hold for any sequence of observations and returns
- Broader applicability
- Lesson:
  - stochastic, stationary assumptions are not essential for learning
  - algorithms are obtained by robustifying familiar algorithms (plus, some new ideas)

PROTOCOL

Initialization: Algorithm gets N and loss function $\ell(\cdot,\cdot)$

$t := 1$

Main loop:

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes prediction $\hat{p}_t$
3. Environment computes outcome y_t, which is revealed to Learner
4. Learner learns
5. $t := t + 1$; go to Step 1

Protocol

Initialization: Algorithm gets N and loss function $\ell(\cdot, \cdot)$

$t := 1$

Main loop:

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes prediction $\hat{p}_t$
3. Environment computes outcome y_t, which is revealed to Learner
4. Learner learns
5. $t := t + 1$; go to Step 1

PROTOCOL

Initialization: Algorithm gets N and loss function $\ell(\cdot, \cdot)$

$t := 1$

Main loop:

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes prediction $\hat{p}_t$
3. Environment computes outcome y_t, which is revealed to Learner
4. Learner learns
5. $t := t + 1$; go to Step 1

PROTOCOL

Initialization: Algorithm gets N and loss function $\ell(\cdot, \cdot)$

$t := 1$

Main loop:

1. Experts' predictions $f_{1,t}, \dots, f_{N,t}$ are revealed to Learner
2. Learner computes prediction $\hat{p}_t$
3. Environment computes outcome y_t, which is revealed to Learner
4. Learner learns
5. $t := t + 1$; go to Step 1

PROTOCOL

Initialization: Algorithm gets N and loss function $\ell(\cdot,\cdot)$

$t := 1$

Main loop:

1. Experts' predictions $f_{1,t},\ldots,f_{N,t}$ are revealed to Learner
2. Learner computes prediction $\hat{p}_t$
3. Environment computes outcome y_t, which is revealed to Learner
4. Learner learns
5. $t := t+1$; go to Step 1

PROTOCOL

Initialization: Algorithm gets N and loss function $\ell(\cdot, \cdot)$

$t := 1$

Main loop:

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes prediction $\hat{p}_t$
3. Environment computes outcome y_t, which is revealed to Learner
4. Learner learns
5. $t := t + 1$; go to Step 1

PROTOCOL

Initialization: Algorithm gets N and loss function $\ell(\cdot, \cdot)$

$t := 1$

Main loop:

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes prediction $\hat{p}_t$
3. Environment computes outcome y_t, which is revealed to Learner
4. Learner learns
5. $t := t + 1$; go to Step 1

PROTOCOL

Initialization: Algorithm gets N and loss function $\ell(\cdot, \cdot)$

$t := 1$

Main loop:

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes prediction $\hat{p}_t$
3. Environment computes outcome y_t, which is revealed to Learner
4. Learner learns
5. $t := t + 1$; go to Step 1

Notation

- (Total) loss of expert i:

$$L_{i,n} = \sum_{t=1}^{n} \ell(f_{it}, y_t)$$

- (Total) loss of best expert:

$$L_n^* = \min_i L_{in}$$

- (Total) loss of algorithm:

$$\hat{L}_n = \sum_{t=1}^{n} \ell(\hat{p}_t, y_t)$$

- (Total) regret:

$$R_n = \hat{L}_n - L_n^*$$

- Goal: Design algorithm that keeps the regret small

Notation

- (Total) loss of expert i:

$$L_{i,n} = \sum_{t=1}^{n} \ell(f_{it}, y_t)$$

- (Total) loss of best expert:

$$L_n^* = \min_i L_{in}$$

- (Total) loss of algorithm:

$$\hat{L}_n = \sum_{t=1}^{n} \ell(\hat{p}_t, y_t)$$

- (Total) regret:

$$R_n = \hat{L}_n - L_n^*$$

- Goal: Design algorithm that keeps the regret small

Notation

- (Total) loss of expert i:

$$L_{i,n} = \sum_{t=1}^{n} \ell(f_{it}, y_t)$$

- (Total) loss of best expert:

$$L_n^* = \min_i L_{in}$$

- (Total) loss of algorithm:

$$\hat{L}_n = \sum_{t=1}^{n} \ell(\hat{p}_t, y_t)$$

- (Total) regret:

$$R_n = \hat{L}_n - L_n^*$$

- Goal: Design algorithm that keeps the regret small

NOTATION

- (Total) loss of expert i:

$$L_{i,n} = \sum_{t=1}^{n} \ell(f_{it}, y_t)$$

- (Total) loss of best expert:

$$L_n^* = \min_i L_{in}$$

- (Total) loss of algorithm:

$$\hat{L}_n = \sum_{t=1}^{n} \ell(\hat{p}_t, y_t)$$

- (Total) regret:

$$R_n = \hat{L}_n - L_n^*$$

- Goal: Design algorithm that keeps the regret small

NOTATION

- (Total) loss of expert i:

$$L_{i,n} = \sum_{t=1}^{n} \ell(f_{it}, y_t)$$

- (Total) loss of best expert:

$$L_n^* = \min_i L_{in}$$

- (Total) loss of algorithm:

$$\hat{L}_n = \sum_{t=1}^{n} \ell(\hat{p}_t, y_t)$$

- (Total) regret:

$$R_n = \hat{L}_n - L_n^*$$

- Goal: Design algorithm that keeps the regret small

Outline

1. High level overview of the talks
2. Motivation
   - What is it?
   - Why should we care?
   - Halving: Find the perfect expert! (0/1 loss)
   - No perfect expert? (0/1 loss)
   - Predicting Continuous Outcomes
3. Discrete prediction problems
   - Randomized forecasters
   - Weighted Average Forecaster
   - Follow the perturbed leader
4. Tracking the best expert
   - Fixed share forecaster
   - Variable-share forecaster
   - Other large classes of experts
5. Non-stochastic bandit problems
   - Exp3.P: An algorithm for non-stochastic bandit problems
6. Conclusions

When there is a infallible expert..

- Binary world:
$$\mathcal{Y} = \mathcal{D} = \{0, 1\}$$
- Loss:
$$\ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$
- N experts
- Expert predictions: $f_{i1}, f_{i2}, \ldots \in \{0, 1\}$

Assumption

There is an expert that never makes a mistake.

Problem

How to keep the regret small?

When there is a infallible expert..

- Binary world:
$$\mathcal{Y} = \mathcal{D} = \{0, 1\}$$
- Loss:
$$\ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$
- N experts
- Expert predictions: $f_{i1}, f_{i2}, \ldots \in \{0, 1\}$

ASSUMPTION

There is an expert that never makes a mistake.

PROBLEM

How to keep the regret small?

When there is a infallible expert..

- Binary world:

$$\mathcal{Y} = \mathcal{D} = \{0, 1\}$$

- Loss:

$$\ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$

- N experts
- Expert predictions: $f_{i1}, f_{i2}, \ldots \in \{0, 1\}$

Assumption

There is an expert that never makes a mistake.

Problem

How to keep the regret small?

When there is a infallible expert..

- Binary world:
$$\mathcal{Y} = \mathcal{D} = \{0, 1\}$$
- Loss:
$$\ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$
- N experts
- Expert predictions: $f_{i1}, f_{i2}, \ldots \in \{0, 1\}$

Assumption

There is an expert that never makes a mistake.

Problem

How to keep the regret small?

When there is a infallible expert..

- Binary world:
$$\mathcal{Y} = \mathcal{D} = \{0, 1\}$$
- Loss:
$$\ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$
- N experts
- Expert predictions: $f_{i1}, f_{i2}, \ldots \in \{0, 1\}$

Assumption

There is an expert that never makes a mistake.

Problem

How to keep the regret small?

When there is a infallible expert..

- Binary world:
$$\mathcal{Y} = \mathcal{D} = \{0, 1\}$$
- Loss:
$$\ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$
- N experts
- Expert predictions: $f_{i1}, f_{i2}, \ldots \in \{0, 1\}$

Assumption

There is an expert that never makes a mistake.

Problem

How to keep the regret small?

Halving Algorithm

- Keep regret small ⇒ Learn from mistakes
- Idea:
  - Eliminate immediately experts that make a mistake
  - Take majority vote of remaining experts

⇒ "Halving Algorithm"

[Barzdin and Freivalds, 1972, Angluin, 1988]

Theorem (Finite regret for the Halving algorithm)

No matter what $y_1, y_2, \ldots$ is,

$$R_n = \hat{L}_n - L_n^* \leq \lfloor \log_2 N \rfloor .$$

Halving Algorithm

- Keep regret small $\Rightarrow$ Learn from mistakes
- Idea:
  - Eliminate immediately experts that make a mistake
  - Take majority vote of remaining experts

$\Rightarrow$ "Halving Algorithm"

[Barzdin and Freivalds, 1972, Angluin, 1988]

Theorem (Finite regret for the Halving algorithm)

No matter what $y_1, y_2, \ldots$ is,

$$R_n = \hat{L}_n - L_n^* \leq \lfloor \log_2 N \rfloor .$$

Halving Algorithm

- Keep regret small ⇒ Learn from mistakes
- Idea:
  - Eliminate immediately experts that make a mistake
  - Take majority vote of remaining experts

⇒ "Halving Algorithm"

[Barzdin and Freivalds, 1972, Angluin, 1988]

Theorem (Finite regret for the Halving algorithm)

No matter what $y_1, y_2, \ldots$ is,

$$R_n = \hat{L}_n - L_n^* \le \lfloor \log_2 N \rfloor .$$

Halving Algorithm

- Keep regret small $\Rightarrow$ Learn from mistakes
- Idea:
  - Eliminate immediately experts that make a mistake
  - Take majority vote of remaining experts

$\Rightarrow$ "Halving Algorithm"

[Barzdin and Freivalds, 1972, Angluin, 1988]

Theorem (Finite regret for the Halving algorithm)

No matter what $y_1, y_2, \ldots$ is,

$$R_n = \hat{L}_n - L_n^* \leq \lfloor \log_2 N \rfloor .$$

Halving Algorithm

- Keep regret small $\Rightarrow$ Learn from mistakes
- Idea:
  - Eliminate immediately experts that make a mistake
  - Take majority vote of remaining experts

$\Rightarrow$ "Halving Algorithm"

[Barzdin and Freivalds, 1972, Angluin, 1988]

Theorem (Finite regret for the Halving algorithm)

No matter what $y_1, y_2, \ldots$ is,

$$R_n = \hat{L}_n - L_n^* \leq \lfloor \log_2 N \rfloor .$$

Halving Algorithm

- Keep regret small $\Rightarrow$ Learn from mistakes
- Idea:
  - Eliminate immediately experts that make a mistake
  - Take majority vote of remaining experts

$\Rightarrow$ "Halving Algorithm"

[Barzdin and Freivalds, 1972, Angluin, 1988]

Theorem (Finite regret for the Halving algorithm)

No matter what $y_1, y_2, \ldots$ is,

$$R_n = \hat{L}_n - L_n^* \leq \lfloor \log_2 N \rfloor .$$

ANALYSIS

- Weight $w_{it} \in \{0, 1\}$:
  Is expert i alive at time t? (after y_t is received)
- Let $w_{i0} = 1$, $i = 1, 2, \ldots, N$.
- $W_t = \sum_{i=1}^{N} w_{it}$: Number of alive experts at time t
- $\hat{L}_t$: number of mistakes up to time t (including time t)

CLAIM

If Halving makes a mistake ($\ell(\hat{p}_t, y_t) = 1$) then $W_t \leq W_{t-1}/2$.
Further W_t cannot grow.

COROLLARY

$W_t \leq W_0/2^{\hat{L}_t} = N/2^{\hat{L}_t}$.

Finish: Now, $1 \leq W_t$, hence $1 \leq N/2^{\hat{L}_t}$, i.e., $\hat{L}_t \leq \lfloor \log_2 N \rfloor$.

ANALYSIS

- Weight $w_{it} \in \{0, 1\}$:
  Is expert i alive at time t? (after y_t is received)
- Let $w_{i0} = 1$, $i = 1, 2, \ldots, N$.
- $W_t = \sum_{i=1}^{N} w_{it}$: Number of alive experts at time t
- $\hat{L}_t$: number of mistakes up to time t (including time t)

CLAIM

If Halving makes a mistake ($\ell(\hat{p}_t, y_t) = 1$) then $W_t \le W_{t-1}/2$.
Further W_t cannot grow.

COROLLARY

$W_t \le W_0/2^{\hat{L}_t} = N/2^{\hat{L}_t}$.

Finish: Now, $1 \le W_t$, hence $1 \le N/2^{\hat{L}_t}$, i.e., $\hat{L}_t \le \lfloor \log_2 N \rfloor$.

Analysis

- Weight $w_{it} \in \{0, 1\}$:
  Is expert i alive at time t? (after y_t is received)
- Let $w_{i0} = 1$, $i = 1, 2, \ldots, N$.
- $W_t = \sum_{i=1}^N w_{it}$: Number of alive experts at time t
- $\hat{L}_t$: number of mistakes up to time t (including time t)

Claim

If Halving makes a mistake ($\ell(\hat{p}_t, y_t) = 1$) then $W_t \le W_{t-1}/2$.
Further W_t cannot grow.

Corollary

$W_t \le W_0/2^{\hat{L}_t} = N/2^{\hat{L}_t}$.

Finish: Now, $1 \le W_t$, hence $1 \le N/2^{\hat{L}_t}$, i.e., $\hat{L}_t \le \lfloor \log_2 N \rfloor$.

ANALYSIS

- Weight $w_{it} \in \{0, 1\}$:
  Is expert i alive at time t? (after y_t is received)
- Let $w_{i0} = 1$, $i = 1, 2, \ldots, N$.
- $W_t = \sum_{i=1}^{N} w_{it}$: Number of alive experts at time t
- $\hat{L}_t$: number of mistakes up to time t (including time t)

CLAIM

If Halving makes a mistake ($\ell(\hat{p}_t, y_t) = 1$) then $W_t \leq W_{t-1}/2$.
Further W_t cannot grow.

COROLLARY

$W_t \leq W_0/2^{\hat{L}_t} = N/2^{\hat{L}_t}$.

Finish: Now, $1 \leq W_t$, hence $1 \leq N/2^{\hat{L}_t}$, i.e., $\hat{L}_t \leq \lfloor \log_2 N \rfloor$.

- Weight $w_{it} \in \{0, 1\}$:
  Is expert i alive at time t? (after y_t is received)
- Let $w_{i0} = 1$, $i = 1, 2, \ldots, N$.
- $W_t = \sum_{i=1}^{N} w_{it}$: Number of alive experts at time t
- $\hat{L}_t$: number of mistakes up to time t (including time t)

CLAIM

If Halving makes a mistake ($\ell(\hat{p}_t, y_t) = 1$) then $W_t \le W_{t-1}/2$.
Further W_t cannot grow.

COROLLARY

$W_t \le W_0/2^{\hat{L}_t} = N/2^{\hat{L}_t}$.

Finish: Now, $1 \le W_t$, hence $1 \le N/2^{\hat{L}_t}$, i.e., $\hat{L}_t \le \lfloor \log_2 N \rfloor$.

- Weight $w_{it} \in \{0, 1\}$:
  Is expert i alive at time t? (after y_t is received)
- Let $w_{i0} = 1$, $i = 1, 2, \ldots, N$.
- $W_t = \sum_{i=1}^{N} w_{it}$: Number of alive experts at time t
- $\hat{L}_t$: number of mistakes up to time t (including time t)

CLAIM

If Halving makes a mistake ($\ell(\hat{p}_t, y_t) = 1$) then $W_t \le W_{t-1}/2$.
Further W_t cannot grow.

COROLLARY

$W_t \le W_0/2^{\hat{L}_t} = N/2^{\hat{L}_t}$.

Finish: Now, $1 \le W_t$, hence $1 \le N/2^{\hat{L}_t}$, i.e., $\hat{L}_t \le \lfloor \log_2 N \rfloor$.

ANALYSIS

- Weight $w_{it} \in \{0, 1\}$:
  Is expert i alive at time t? (after y_t is received)
- Let $w_{i0} = 1$, $i = 1, 2, \ldots, N$.
- $W_t = \sum_{i=1}^N w_{it}$: Number of alive experts at time t
- $\hat{L}_t$: number of mistakes up to time t (including time t)

CLAIM

If Halving makes a mistake ($\ell(\hat{p}_t, y_t) = 1$) then $W_t \leq W_{t-1}/2$.
Further W_t cannot grow.

COROLLARY

$W_t \leq W_0/2^{\hat{L}_t} = N/2^{\hat{L}_t}$.

Finish: Now, $1 \leq W_t$, hence $1 \leq N/2^{\hat{L}_t}$, i.e., $\hat{L}_t \leq \lfloor \log_2 N \rfloor$.

OUTLINE

No Perfect Expert: "Weighted Majority"

- Problem with elimination: fails if there is no perfect expert!
- Improved algorithm: "Weighted Majority"
  [Littlestone and Warmuth, 1994]
  - Keep weights positive!
  - Have weights of mistaken experts decay:

$$w_{it} = \beta w_{i,t-1}, \text{ if } f_{it} \neq y_t \; (0 < \beta < 1)$$

  - Keep majority vote!

Theorem (Loss bound for WM)

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

No Perfect Expert: "Weighted Majority"

- Problem with elimination: fails if there is no perfect expert!
- Improved algorithm: "Weighted Majority"
  [Littlestone and Warmuth, 1994]
  - Keep weights positive!
  - Have weights of mistaken experts decay:

$$w_{it} = \beta w_{i,t-1}, \text{ if } f_{it} \neq y_t \; (0 < \beta < 1)$$

  - Keep majority vote!

Theorem (Loss bound for WM)

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor.$$

No Perfect Expert: "Weighted Majority"

- Problem with elimination: fails if there is no perfect expert!
- Improved algorithm: "Weighted Majority"
  [Littlestone and Warmuth, 1994]
  - Keep weights positive!
  - Have weights of mistaken experts decay:

$$w_{it} = \beta w_{i,t-1}, \text{if } f_{it} \neq y_t \; (0 < \beta < 1)$$

  - Keep majority vote!

Theorem (Loss bound for WM)

$$\hat{L}_n \leq \left[\frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right].$$

No Perfect Expert: "Weighted Majority"

- Problem with elimination: fails if there is no perfect expert!
- Improved algorithm: "Weighted Majority"
  [Littlestone and Warmuth, 1994]
  - Keep weights positive!
  - Have weights of mistaken experts decay:

$$w_{it} = \beta w_{i,t-1}, \text{if } f_{it} \neq y_t \; (0 < \beta < 1)$$

  - Keep majority vote!

Theorem (Loss bound for WM)

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

No Perfect Expert: "Weighted Majority"

- Problem with elimination: fails if there is no perfect expert!
- Improved algorithm: "Weighted Majority" [Littlestone and Warmuth, 1994]
  - Keep weights positive!
  - Have weights of mistaken experts decay:

$$w_{it} = \beta w_{i,t-1}, \text{if } f_{it} \neq y_t \; (0 < \beta < 1)$$

  - Keep majority vote!

Theorem (Loss bound for WM)

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

No Perfect Expert: "Weighted Majority"

- Problem with elimination: fails if there is no perfect expert!
- Improved algorithm: "Weighted Majority"
  [Littlestone and Warmuth, 1994]
  - Keep weights positive!
  - Have weights of mistaken experts decay:

$$w_{it} = \beta w_{i,t-1}, \text{if } f_{it} \neq y_t \; (0 < \beta < 1)$$

  - Keep majority vote!

Theorem (Loss bound for WM)

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

Outline

Predicting Continuous Outcomes

- What if $\mathcal{Y} = \mathcal{D} = [0,1]$ or $\mathbb{R}^d$ (or a convex subset of some vector space)?
- Bounded loss: $\ell : \mathcal{D} \times \mathcal{Y} \to [0,1]$
- Example: $\mathcal{D} = \mathcal{Y} = [0,1]$, $\ell(p,y) = \frac{1}{2}|p-y|$.
- Can we generalize the previous algorithm?
- Take the weighted combination of the experts' predictions!

$$\hat{p}_t = \frac{\sum_{i=1}^{N} w_{i,t-1} f_{it}}{\sum_{i=1}^{N} w_{it}}$$

- How to set the weights?

$$w_{i,t} = w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}.$$

Predicting Continuous Outcomes

- What if $\mathcal{Y} = \mathcal{D} = [0, 1]$ or $\mathbb{R}^d$ (or a convex subset of some vector space)?
- Bounded loss: $\ell : \mathcal{D} \times \mathcal{Y} \to [0, 1]$
- Example: $\mathcal{D} = \mathcal{Y} = [0, 1]$, $\ell(p, y) = \frac{1}{2}|p - y|$.
- Can we generalize the previous algorithm?
- Take the weighted combination of the experts' predictions!

$$\hat{p}_t = \frac{\sum_{i=1}^{N} w_{i,t-1} f_{it}}{\sum_{i=1}^{N} w_{it}}$$

- How to set the weights?

$$w_{i,t} = w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}.$$

Predicting Continuous Outcomes

- What if $\mathcal{Y} = \mathcal{D} = [0, 1]$ or $\mathbb{R}^d$ (or a convex subset of some vector space)?
- Bounded loss: $\ell : \mathcal{D} \times \mathcal{Y} \to [0, 1]$
- Example: $\mathcal{D} = \mathcal{Y} = [0, 1]$, $\ell(p, y) = \frac{1}{2}|p - y|$.
- Can we generalize the previous algorithm?
- Take the weighted combination of the experts' predictions!

$$\hat{p}_t = \frac{\sum_{i=1}^N w_{i,t-1} f_{it}}{\sum_{i=1}^N w_{it}}$$

- How to set the weights?

$$w_{i,t} = w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}.$$

Predicting Continuous Outcomes

- What if $\mathcal{Y} = \mathcal{D} = [0, 1]$ or $\mathbb{R}^d$ (or a convex subset of some vector space)?
- Bounded loss: $\ell : \mathcal{D} \times \mathcal{Y} \to [0, 1]$
- Example: $\mathcal{D} = \mathcal{Y} = [0, 1]$, $\ell(p, y) = \frac{1}{2}|p - y|$.
- Can we generalize the previous algorithm?
- Take the weighted combination of the experts' predictions!

$$\hat{p}_t = \frac{\sum_{i=1}^{N} w_{i,t-1} f_{it}}{\sum_{i=1}^{N} w_{it}}$$

- How to set the weights?

$$w_{i,t} = w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}.$$

Predicting Continuous Outcomes

- What if $\mathcal{Y} = \mathcal{D} = [0,1]$ or $\mathbb{R}^d$ (or a convex subset of some vector space)?
- Bounded loss: $\ell : \mathcal{D} \times \mathcal{Y} \to [0,1]$
- Example: $\mathcal{D} = \mathcal{Y} = [0,1]$, $\ell(p,y) = \frac{1}{2}|p-y|$.
- Can we generalize the previous algorithm?
- Take the weighted combination of the experts' predictions!

$$\hat{p}_t = \frac{\sum_{i=1}^{N} w_{i,t-1} f_{it}}{\sum_{i=1}^{N} w_{it}}$$

- How to set the weights?

$$w_{i,t} = w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}.$$

PREDICTING CONTINUOUS OUTCOMES

- What if $\mathcal{Y} = \mathcal{D} = [0,1]$ or $\mathbb{R}^d$ (or a convex subset of some vector space)?
- Bounded loss: $\ell : \mathcal{D} \times \mathcal{Y} \to [0,1]$
- Example: $\mathcal{D} = \mathcal{Y} = [0,1]$, $\ell(p,y) = \frac{1}{2}|p-y|$.
- Can we generalize the previous algorithm?
- Take the weighted combination of the experts' predictions!

$$\hat{p}_t = \frac{\sum_{i=1}^{N} w_{i,t-1} f_{it}}{\sum_{i=1}^{N} w_{it}}$$

- How to set the weights?

$$w_{i,t} = w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}.$$

Predicting Continuous Outcomes

- What if $\mathcal{Y} = \mathcal{D} = [0, 1]$ or $\mathbb{R}^d$ (or a convex subset of some vector space)?
- Bounded loss: $\ell : \mathcal{D} \times \mathcal{Y} \to [0, 1]$
- Example: $\mathcal{D} = \mathcal{Y} = [0, 1]$, $\ell(p, y) = \frac{1}{2}|p - y|$.
- Can we generalize the previous algorithm?
- Take the weighted combination of the experts' predictions!

$$\hat{p}_t = \frac{\sum_{i=1}^{N} w_{i,t-1} f_{it}}{\sum_{i=1}^{N} w_{it}}$$

- How to set the weights?

$$w_{i,t} = w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}.$$

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it}/V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it} / V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it} / V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it}/V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it} / V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates!

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it} / V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it} / V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it}/V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates!

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it}/V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates!

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it} / V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it} / V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates!

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it}/V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates!

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it} / V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates!

Exponentially Weighted Average forecaster

EWA Algorithm(η)

Initialization: $w_{it} := 1/N$, $i = 1, 2, \ldots, N$

At time t do:

1. Receive Expert predictions $(f_{1t}, \ldots, f_{Nt})$
2. $\hat{p}_t := \sum_{i=1}^{N} w_{i,t-1} f_{it}$
3. Send $\hat{p}_t$ to the Environment
4. Receive y_t from the Environment
5. Update weights:

$$v_{it} := w_{i,t-1} e^{-\eta \ell(f_{it}, y_t)}$$

$$V_t := \sum_i v_{it}$$

$$w_{it} := v_{it} / V_t$$

Remark

- Normalization is good for numerical stability
- Update resembles Bayes updates!

LOSS BOUND

THEOREM (LOSS BOUND FOR THE EWA FORECASTER)

Assume that $\mathcal{D}$ is a convex subset of some vector-space. Let $\ell : \mathcal{D} \times \mathcal{Y} \to [0, 1]$ be convex in its first argument and consider the loss $\hat{L}_n$ of EWA. Then:

$$\hat{L}_n \le L_n^* + \frac{\ln N}{\eta} + \frac{\eta}{8} n.$$

With $\eta = \sqrt{\frac{8 \ln N}{n}}$,

$$\hat{L}_n \le L_n^* + \sqrt{\frac{n \ln N}{2}}.$$

ADAPTIVE AND SELF-CONFIDENT FORECASTERS

- Problem: η depends on n, the horizon
- Small losses
  - Loss bound for WM, 0/1-predictions:

$$\hat{L}_n \le \left\lfloor \frac{\log_2(\frac{1}{\beta})L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor.$$

  - If $L_{in} = 0$ for some expert then the regret is finite!
  - Regret bound for EWA:

$$\hat{L}_n \le L_n^* + \sqrt{n/2 \, \ln N} \xrightarrow{n\to\infty} \infty \text{ even if } L_n^* = 0!$$

THEOREM ([AUER ET AL., 2002B])

Consider EWA with $\eta_t = c\sqrt{\ln N / L_{t-1}^*}$, $c > 0$. Under the same conditions as in the previous theorem for some $\kappa > 0$,

$$R_n \le 2\sqrt{2L_n^* \ln N} + \kappa \ln N.$$

Adaptive and self-confident forecasters

- Problem: η depends on n, the horizon
- Small losses
  - Loss bound for WM, 0/1-predictions:

$$\hat{L}_n \le \left\lfloor \frac{\log_2(\frac{1}{\beta})L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

  - If $L_{in} = 0$ for some expert then the regret is finite!
  - Regret bound for EWA:

$$\hat{L}_n \le L_n^* + \sqrt{n/2 \, \ln N} \stackrel{n\to\infty}{\longrightarrow} \infty \text{ even if } L_n^* = 0!$$

Theorem ([Auer et al., 2002b])

Consider EWA with $\eta_t = c\sqrt{\ln N / L_{t-1}^*}$, $c > 0$. Under the same conditions as in the previous theorem for some $\kappa > 0$,

$$R_n \le 2\sqrt{2L_n^* \ln N} + \kappa \ln N.$$

ADAPTIVE AND SELF-CONFIDENT FORECASTERS

- Problem: η depends on n, the horizon
- Small losses
  - Loss bound for WM, 0/1-predictions:

$$\hat{L}_n \le \left\lfloor \frac{\log_2(\frac{1}{\beta})L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

  - If $L_{in} = 0$ for some expert then the regret is finite!
  - Regret bound for EWA:

$$\hat{L}_n \le L_n^* + \sqrt{n/2 \ \ln N} \stackrel{n\to\infty}{\longrightarrow} \infty \text{ even if } L_n^* = 0!$$

THEOREM ([AUER ET AL., 2002B])

Consider EWA with $\eta_t = c\sqrt{\ln N / L_{t-1}^*}$, $c > 0$. Under the same conditions as in the previous theorem for some $\kappa > 0$,

$$R_n \le 2\sqrt{2L_n^* \ln N} + \kappa \ln N.$$

Adaptive and self-confident forecasters

- Problem: η depends on n, the horizon
- Small losses
  - Loss bound for WM, 0/1-predictions:

$$\hat{L}_n \le \left\lfloor \frac{\log_2(\frac{1}{\beta})L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

  - If $L_{in} = 0$ for some expert then the regret is finite!
  - Regret bound for EWA:

$$\hat{L}_n \le L_n^* + \sqrt{n/2 \ \ln N} \ \stackrel{n\to\infty}{\longrightarrow} \infty \text{ even if } L_n^* = 0!$$

Theorem ([Auer et al., 2002b])

Consider EWA with $\eta_t = c\sqrt{\ln N / L_{t-1}^*}$, $c > 0$. Under the same conditions as in the previous theorem for some $\kappa > 0$,

$$R_n \le 2\sqrt{2L_n^* \ln N} + \kappa \ln N.$$

ADAPTIVE AND SELF-CONFIDENT FORECASTERS

- Problem: η depends on n, the horizon
- Small losses
  - Loss bound for WM, 0/1-predictions:

$$\hat{L}_n \le \left\lfloor \frac{\log_2(\frac{1}{\beta})L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

  - If $L_{in} = 0$ for some expert then the regret is finite!
  - Regret bound for EWA:

$$\hat{L}_n \le L_n^* + \sqrt{n/2\ \ln N} \xrightarrow{n\to\infty} \infty \text{ even if } L_n^* = 0!$$

THEOREM ([AUER ET AL., 2002B])

Consider EWA with $\eta_t = c\sqrt{\ln N / L_{t-1}^*}$, $c > 0$. Under the same conditions as in the previous theorem for some $\kappa > 0$,

$$R_n \le 2\sqrt{2L_n^* \ln N} + \kappa \ln N.$$

ADAPTIVE AND SELF-CONFIDENT FORECASTERS

- Problem: η depends on n, the horizon
- Small losses
  - Loss bound for WM, 0/1-predictions:

$$\hat{L}_n \le \left\lfloor \frac{\log_2(\frac{1}{\beta})L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

  - If $L_{in} = 0$ for some expert then the regret is finite!
  - Regret bound for EWA:

$$\hat{L}_n \le L_n^* + \sqrt{n/2 \ \ln N} \stackrel{n\to\infty}{\longrightarrow} \infty \text{ even if } L_n^* = 0!$$

THEOREM ([AUER ET AL., 2002B])

Consider EWA with $\eta_t = c\sqrt{\ln N / L_{t-1}^*}$, $c > 0$. Under the same conditions as in the previous theorem for some $\kappa > 0$,

$$R_n \le 2\sqrt{2L_n^* \ln N} + \kappa \ln N.$$

Binary prediction problems

- Binary prediction problem:

$$\mathcal{D} = \mathcal{Y} = \{0, 1\}, \quad \ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$

- Bound of WM:

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor.$$

- Question: Can we have an additive bound, like that of EWA:

$$\hat{L}_n \leq L_n^* + B(n, N)$$

with $B(n, N) = o(n)$?
  - Can we have such a bound for WM?
  - For some other algorithm?

- Binary prediction problem:

$$\mathcal{D} = \mathcal{Y} = \{0, 1\}, \quad \ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$

- Bound of WM:

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

- Question: Can we have an additive bound, like that of EWA:

$$\hat{L}_n \leq L_n^* + B(n, N)$$

with $B(n, N) = o(n)$?
    - Can we have such a bound for WM?
    - For some other algorithm?

Binary prediction problems

- Binary prediction problem:

$$\mathcal{D} = \mathcal{Y} = \{0, 1\}, \quad \ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$

- Bound of WM:

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

- Question: Can we have an additive bound, like that of EWA:

$$\hat{L}_n \leq L_n^* + B(n, N)$$

with $B(n, N) = o(n)$?

  - Can we have such a bound for WM?
  - For some other algorithm?

Binary prediction problems

- Binary prediction problem:

$$\mathcal{D} = \mathcal{Y} = \{0, 1\}, \quad \ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$

- Bound of WM:

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta}) L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

- Question: Can we have an additive bound, like that of EWA:

$$\hat{L}_n \leq L_n^* + B(n, N)$$

with $B(n, N) = o(n)$?
  - Can we have such a bound for WM?
  - For some other algorithm?

- Binary prediction problem:

$$\mathcal{D} \quad = \quad \mathcal{Y} = \{0, 1\}, \quad \ell(p, y) = \mathbb{I}_{\{p \neq y\}}$$

- Bound of WM:

$$\hat{L}_n \leq \left\lfloor \frac{\log_2(\frac{1}{\beta})L_n^* + \log_2 N}{\log_2(\frac{2}{1+\beta})} \right\rfloor .$$

- Question: Can we have an additive bound, like that of EWA:

$$\hat{L}_n \leq L_n^* + B(n, N)$$

  with $B(n, N) = o(n)$?
  - Can we have such a bound for WM?
  - For some other algorithm?

Proposition

Consider binary prediction problems and pick any deterministic forecaster. Let $\hat{L}_n(y_{1:n})$ be the forecaster's loss on $y_{1:n}$. Then $\exists y_{1:n}$ s.t. $\hat{L}_n(y_{1:n}) = n$.

Proof.

Induction on n. □

Corollary

No deterministic forecaster can have sublinear regret.

Proof.

Let $N = 2$, $f_{1t} \equiv 0$, $f_{2t} \equiv 1$. Then $\forall\, y_{1:n}$, $L_n^*(y_{1:n}) \le n/2$.
Pick some $y_{1:n}$ that forces $\hat{L}_n(y_{1:n}) = n$. □

Idea

Randomize the forecaster!

WHY RANDOMIZE?

PROPOSITION

Consider binary prediction problems and pick any deterministic forecaster. Let $\hat{L}_n(y_{1:n})$ be the forecaster's loss on $y_{1:n}$. Then $\exists y_{1:n}$ s.t. $\hat{L}_n(y_{1:n}) = n$.

PROOF.

Induction on n. □

COROLLARY

No deterministic forecaster can have sublinear regret.

PROOF.

Let $N = 2$, $f_{1t} \equiv 0$, $f_{2t} \equiv 1$. Then $\forall\, y_{1:n}$, $L_n^*(y_{1:n}) \leq n/2$. Pick some $y_{1:n}$ that forces $\hat{L}_n(y_{1:n}) = n$. □

IDEA

Randomize the forecaster!

Why randomize?

Proposition

Consider binary prediction problems and pick any deterministic forecaster. Let $\hat{L}_n(y_{1:n})$ be the forecaster's loss on $y_{1:n}$.
Then $\exists y_{1:n}$ s.t. $\hat{L}_n(y_{1:n}) = n$.

Proof.

Induction on n. □

Corollary

No deterministic forecaster can have sublinear regret.

Proof.

Let $N = 2$, $f_{1t} \equiv 0$, $f_{2t} \equiv 1$. Then $\forall\, y_{1:n}$, $L_n^*(y_{1:n}) \le n/2$.
Pick some $y_{1:n}$ that forces $\hat{L}_n(y_{1:n}) = n$. □

Idea

Randomize the forecaster!

Why randomize?

Proposition

Consider binary prediction problems and pick any deterministic forecaster. Let $\hat{L}_n(y_{1:n})$ be the forecaster's loss on $y_{1:n}$. Then $\exists y_{1:n}$ s.t. $\hat{L}_n(y_{1:n}) = n$.

Proof.

Induction on n. □

Corollary

No deterministic forecaster can have sublinear regret.

Proof.

Let $N = 2$, $f_{1t} \equiv 0$, $f_{2t} \equiv 1$. Then $\forall\ y_{1:n}$, $L_n^*(y_{1:n}) \leq n/2$.
Pick some $y_{1:n}$ that forces $\hat{L}_n(y_{1:n}) = n$. □

Idea

Randomize the forecaster!

Why randomize?

Proposition

Consider binary prediction problems and pick any deterministic forecaster. Let $\hat{L}_n(y_{1:n})$ be the forecaster's loss on $y_{1:n}$. Then $\exists y_{1:n}$ s.t. $\hat{L}_n(y_{1:n}) = n$.

Proof.

Induction on n. □

Corollary

No deterministic forecaster can have sublinear regret.

Proof.

Let $N = 2$, $f_{1t} \equiv 0$, $f_{2t} \equiv 1$. Then $\forall\ y_{1:n}$, $L_n^*(y_{1:n}) \leq n/2$.
Pick some $y_{1:n}$ that forces $\hat{L}_n(y_{1:n}) = n$. □

Idea

Randomize the forecaster!

Outline

RANDOMIZED FORECASTERS

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

PROTOCOL

Initialization: Algorithm gets N and ε, $t := 1$
At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t)$ is revealed to Learner

RANDOMIZED FORECASTERS

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!

Randomized forecasters

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t)$ is revealed to Learner

RANDOMIZED FORECASTERS

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

PROTOCOL

RANDOMIZED FORECASTERS

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \dots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

PROTOCOL

RANDOMIZED FORECASTERS

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Randomized forecasters

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: “Simulate EWA”:

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

Randomized forecasters

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: “Simulate EWA”:

$$I_t \sim (w_{1,t-1}, \dots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \dots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \dots, \ell(N, Y_t))$ is revealed to Learner

RANDOMIZED FORECASTERS

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

Randomized forecasters

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

Randomized forecasters

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

Randomized forecasters

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

RANDOMIZED FORECASTERS

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

RANDOMIZED FORECASTERS

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

Randomized forecasters

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

Randomized forecasters

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

RANDOMIZED FORECASTERS

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

Randomized forecasters

- Can we use EWA to get sublinear regret?
  .. but predictions must be binary!
- Crucial differences:
  - predictions cannot be combined
  - $\ell(\cdot, y)$ is not convex
- Idea: "Simulate EWA":

$$I_t \sim (w_{1,t-1}, \ldots, w_{N,t-1}), \hat{p}_t = f_{I_t,t}.$$

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t

1. Experts' predictions $f_{1,t}, \ldots, f_{N,t}$ are revealed to Learner
2. Learner computes I_t
3. Environment computes outcome Y_t
4. Losses $\ell(1, Y_t), \ell(2, Y_t), \ldots, \ell(N, Y_t))$ is revealed to Learner

OUTLINE

Weighted Average Forecaster [Littlestone and Warmuth, 1994]

Previous result on EWA:

Theorem (Loss bound for the EWA forecaster)

Assume that $\mathcal{D}$ is a convex subset of some vector-space. Let $\ell : \mathcal{D} \times \mathcal{Y} \to [0,1]$ be convex in its first argument. Then, for EWA ($\hat{p}_t = \frac{\sum_i w_{i,t-1} f_{it}}{\sum_j w_{j,t-1}}$, $w_{i,t-1} = e^{-\eta L_{i,t-1}}$) it holds:

$$\hat{L}_n - L_n^* \leq \frac{\ln N}{\eta} + \frac{\eta}{8} n.$$

With $\eta = \sqrt{\frac{8 \ln N}{n}}$, $\hat{L}_n - L_n^* \leq \sqrt{n/2 \; \ln N}$.

- Let $f_{it} = e_i$ (ith unit vector), $\hat{p}_{it} = \frac{w_{i,t-1}}{\sum_{j=1}^N w_{j,t-1}}$
- $\bar{\ell}(p,y) \stackrel{\text{def}}{=} \sum_{i=1}^N p_i \ell(i,y)$, $\Rightarrow \bar{\ell}$ is convex in p
- $\mathcal{D} = \Delta_1 \stackrel{\text{def}}{=} \{ p \in \mathbb{R}^N \,|\, p_i \geq 0, \sum_j p_i = 1 \} \subset \mathbb{R}^N$ is convex.

Weighted Average Forecaster [Littlestone and Warmuth, 1994]

Previous result on EWA:

Theorem (Loss bound for the EWA forecaster)

Assume that $\mathcal{D}$ is a convex subset of some vector-space. Let $\ell : \mathcal{D} \times \mathcal{Y} \to [0,1]$ be convex in its first argument. Then, for EWA ($\hat{p}_t = \frac{\sum_i w_{i,t-1} f_{it}}{\sum_j w_{j,t-1}}$, $w_{i,t-1} = e^{-\eta L_{i,t-1}}$) it holds:

$$\hat{L}_n - L_n^* \le \frac{\ln N}{\eta} + \frac{\eta}{8} n.$$

With $\eta = \sqrt{\frac{8 \ln N}{n}}$, $\hat{L}_n - L_n^* \le \sqrt{n/2 \ \ln N}$.

- Let $f_{it} = e_i$ (ith unit vector), $\hat{p}_{it} = \frac{w_{i,t-1}}{\sum_{j=1}^N w_{j,t-1}}$
- $\bar{\ell}(p,y) \stackrel{\text{def}}{=} \sum_{i=1}^N p_i \ell(i,y)$, $\Rightarrow$ $\bar{\ell}$ is convex in p
- $\mathcal{D} = \Delta_1 \stackrel{\text{def}}{=} \{ p \in \mathbb{R}^N \mid p_i \ge 0, \sum_j p_j = 1 \} \subset \mathbb{R}^N$ is convex.

Weighted Average Forecaster [Littlestone and Warmuth, 1994]

Previous result on EWA:

Theorem (Loss bound for the EWA forecaster)

Assume that $\mathcal{D}$ is a convex subset of some vector-space. Let $\ell : \mathcal{D} \times \mathcal{Y} \to [0,1]$ be convex in its first argument. Then, for EWA ($\hat{p}_t = \frac{\sum_i w_{i,t-1} f_{it}}{\sum_j w_{j,t-1}}$, $w_{i,t-1} = e^{-\eta L_{i,t-1}}$) it holds:

$$\hat{L}_n - L_n^* \le \frac{\ln N}{\eta} + \frac{\eta}{8} n.$$

With $\eta = \sqrt{\frac{8 \ln N}{n}}$, $\hat{L}_n - L_n^* \le \sqrt{n/2 \ \ln N}$.

- Let $f_{it} = e_i$ (ith unit vector), $\hat{p}_{it} = \frac{w_{i,t-1}}{\sum_{j=1}^N w_{j,t-1}}$
- $\bar{\ell}(p, y) \stackrel{\text{def}}{=} \sum_{i=1}^N p_i \ell(i, y)$, $\Rightarrow \bar{\ell}$ is convex in p
- $\mathcal{D} = \Delta_1 \stackrel{\text{def}}{=} \{ p \in \mathbb{R}^N \mid p_i \ge 0, \sum_j p_i = 1 \} \subset \mathbb{R}^N$ is convex.

Weighted Average Forecaster [Littlestone and Warmuth, 1994]

Previous result on EWA:

Theorem (Loss bound for the EWA forecaster)

Assume that $\mathcal{D}$ *is a convex subset of some vector-space. Let* $\ell : \mathcal{D} \times \mathcal{Y} \to [0,1]$ *be convex in its first argument. Then, for EWA (*$\hat{p}_t = \frac{\sum_i w_{i,t-1} f_{it}}{\sum_j w_{j,t-1}}$, $w_{i,t-1} = e^{-\eta L_{i,t-1}}$*) it holds:*

$$\hat{L}_n - L_n^* \le \frac{\ln N}{\eta} + \frac{\eta}{8} n.$$

With $\eta = \sqrt{\frac{8 \ln N}{n}}$, $\hat{L}_n - L_n^* \le \sqrt{n/2 \, \ln N}$.

- Let $f_{it} = e_i$ (ith unit vector), $\hat{p}_{it} = \frac{w_{i,t-1}}{\sum_{j=1}^N w_{j,t-1}}$
- $\bar{\ell}(p,y) \stackrel{\text{def}}{=} \sum_{i=1}^N p_i \ell(i,y)$, $\Rightarrow \bar{\ell}$ is convex in p
- $\mathcal{D} = \Delta_1 \stackrel{\text{def}}{=} \{ p \in \mathbb{R}^N \mid p_i \ge 0, \sum_j p_i = 1 \} \subset \mathbb{R}^N$ is convex.

Weighted Average Forecaster [Littlestone and Warmuth, 1994]

Previous result on EWA:

Theorem (Loss bound for the EWA forecaster)

Assume that $\mathcal{D}$ *is a convex subset of some vector-space. Let* $\ell : \mathcal{D} \times \mathcal{Y} \to [0,1]$ *be convex in its first argument. Then, for EWA* ($\hat{p}_t = \frac{\sum_i w_{i,t-1} f_{it}}{\sum_j w_{j,t-1}}$, $w_{i,t-1} = e^{-\eta L_{i,t-1}}$) *it holds:*

$$\hat{L}_n - L_n^* \leq \frac{\ln N}{\eta} + \frac{\eta}{8} n.$$

With $\eta = \sqrt{\frac{8 \ln N}{n}}$, $\hat{L}_n - L_n^* \leq \sqrt{n/2 \, \ln N}$.

- Let $f_{it} = e_i$ (ith unit vector), $\hat{p}_{it} = \frac{w_{i,t-1}}{\sum_{j=1}^N w_{j,t-1}}$
- $\bar{\ell}(p,y) \stackrel{\text{def}}{=} \sum_{i=1}^N p_i \ell(i,y)$, $\Rightarrow \bar{\ell}$ is convex in p
- $\mathcal{D} = \Delta_1 \stackrel{\text{def}}{=} \{ p \in \mathbb{R}^N \,|\, p_i \geq 0, \sum_j p_i = 1 \} \subset \mathbb{R}^N$ is convex.

BOUND ON THE PSEUDO-EXPECTED REGRET

EWA: $\hat{p}_t = \frac{\sum_i w_{i,t-1} f_{it}}{\sum_j w_{j,t-1}}$, $w_{i,t-1} = e^{-\eta L_{i,t-1}}$

THEOREM (LOSS BOUND FOR THE EWA FORECASTER: RANDOMIZED PREDICTIONS)

Let $\ell : \underline{N} \times \mathcal{Y} \to [0,1]$. Then, for EWA it holds:

$$\overline{L}_n - L_n^* \leq \frac{\ln N}{\eta} + \frac{\eta}{8} n.$$

With $\eta = \sqrt{\frac{8 \ln N}{n}}$, $\overline{L}_n - L_n^* \leq \sqrt{n/2\ \ln N}$. Here

$$\overline{L}_n = \sum_{t=1}^{n} \overline{\ell}(\hat{p}_t, Y_t) = \sum_{t=1}^{n} \sum_{i=1}^{N} \hat{p}_{it} \ell(i, Y_t).$$

Note:

$$\overline{\ell}(\hat{p}_t, Y_t) = \mathbb{E}\left[\ell(I_t, Y_t) \mid Y_{1:t}, I_{1:t-1}\right] (= \mathbb{E}_t\left[\ell(I_t, Y_t)\right]).$$

Bound on the pseudo-expected regret

EWA: $\hat{p}_t = \frac{\sum_i w_{i,t-1} f_{it}}{\sum_j w_{j,t-1}}$, $w_{i,t-1} = e^{-\eta L_{i,t-1}}$

Theorem (Loss bound for the EWA forecaster: Randomized predictions)

Let $\ell : \underline{N} \times \mathcal{Y} \to [0,1]$. *Then, for EWA it holds:*

$$\overline{L}_n - L_n^* \le \frac{\ln N}{\eta} + \frac{\eta}{8} n.$$

With $\eta = \sqrt{\frac{8 \ln N}{n}}$, $\overline{L}_n - L_n^* \le \sqrt{n/2\ \ln N}$. *Here*

$$\overline{L}_n = \sum_{t=1}^{n} \overline{\ell}(\hat{p}_t, Y_t) = \sum_{t=1}^{n} \sum_{i=1}^{N} \hat{p}_{it} \ell(i, Y_t).$$

Note:

$$\overline{\ell}(\hat{p}_t, Y_t) = \mathbb{E}\left[\ell(I_t, Y_t) \mid Y_{1:t}, I_{1:t-1}\right] \left(= \mathbb{E}_t\left[\ell(I_t, Y_t)\right]\right).$$

Bound on the Actual Regret

- What about $\hat{L}_n - L_n^*$??
- $\hat{L}_n = \sum_{t=1}^{n} \ell(I_t, Y_t) \overset{??}{\approx} \sum_{t=1}^{n} \overline{\ell}(\hat{p}_t, Y_t) = \overline{L}_n$
- $\overline{\ell}(\hat{p}_t, Y_t)$ is the (conditional) "expected value" of $\ell(I_t, Y_t)$
- Hoeffding $\Rightarrow$ Sums of i.i.d. random variables are $\sqrt{n}$-close to their expectations!

  Extension to martingales $\Rightarrow$ Hoeffding-Azuma

- What about $\hat{L}_n - L_n^*$??
- $\hat{L}_n = \sum_{t=1}^{n} \ell(I_t, Y_t) \overset{??}{\approx} \sum_{t=1}^{n} \overline{\ell}(\hat{p}_t, Y_t) = \overline{L}_n$
- $\overline{\ell}(\hat{p}_t, Y_t)$ is the (conditional) "expected value" of $\ell(I_t, Y_t)$
- Hoeffding $\Rightarrow$ Sums of i.i.d. random variables are $\sqrt{n}$-close to their expectations!

  Extension to martingales $\Rightarrow$ Hoeffding-Azuma

Bound on the Actual Regret

- What about $\hat{L}_n - L_n^*$??
- $\hat{L}_n = \sum_{t=1}^n \ell(I_t, Y_t) \overset{??}{\approx} \sum_{t=1}^n \bar{\ell}(\hat{p}_t, Y_t) = \bar{L}_n$
- $\bar{\ell}(\hat{p}_t, Y_t)$ is the (conditional) "expected value" of $\ell(I_t, Y_t)$
- Hoeffding $\Rightarrow$ Sums of i.i.d. random variables are $\sqrt{n}$-close to their expectations!

  Extension to martingales $\Rightarrow$ Hoeffding-Azuma

Bound on the Actual Regret

- What about $\hat{L}_n - L_n^*$??
- $\hat{L}_n = \sum_{t=1}^n \ell(I_t, Y_t) \overset{??}{\approx} \sum_{t=1}^n \bar{\ell}(\hat{p}_t, Y_t) = \bar{L}_n$
- $\bar{\ell}(\hat{p}_t, Y_t)$ is the (conditional) "expected value" of $\ell(I_t, Y_t)$
- Hoeffding $\Rightarrow$ Sums of i.i.d. random variables are $\sqrt{n}$-close to their expectations!

Extension to martingales $\Rightarrow$ Hoeffding-Azuma

Bound on the Actual Regret

- What about $\hat{L}_n - L_n^*$??
- $\hat{L}_n = \sum_{t=1}^n \ell(I_t, Y_t) \overset{??}{\approx} \sum_{t=1}^n \bar{\ell}(\hat{p}_t, Y_t) = \bar{L}_n$
- $\bar{\ell}(\hat{p}_t, Y_t)$ is the (conditional) "expected value" of $\ell(I_t, Y_t)$
- Hoeffding $\Rightarrow$ Sums of i.i.d. random variables are $\sqrt{n}$-close to their expectations!

  Extension to martingales $\Rightarrow$ Hoeffding-Azuma

Bound on the random regret

Theorem (Loss bound for the EWA forecaster: Random regret)

Let $\ell : \underline{N} \times \mathcal{Y} \to [0, 1]$*. Then, for EWA it holds:*

$$\hat{L}_n - L_n^* \leq \frac{\ln N}{\eta} + \frac{\eta}{8} n + \sqrt{\frac{n}{2} \ln(1/\delta)}$$

With $\eta = \sqrt{\frac{8 \ln N}{n}}$,

$$\hat{L}_n - L_n^* \leq \sqrt{\frac{n}{2} \ln N} + \sqrt{\frac{n}{2} \ln(1/\delta)}.$$

Small losses

- Previous "small-loss" bound:

$$2\sqrt{2L_n^* \ln N} + \kappa \ln N$$

- Random fluctuations: add $\sqrt{n/2 \ln(1/\delta)}$ – too big!
- Bernstein's inequality uses the "predictable variance" to bound the fluctuations
- Bound on the "predictable variance":

$$\begin{aligned}\mathbb{E}_t\left[\left(\ell(I_t, Y_t) - \overline{\ell}(\hat{p}_t, Y_t)\right)^2\right] &= \mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] - \overline{\ell}^2(\hat{p}_t, Y_t) \\ &\le \mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] \le \mathbb{E}_t\left[\ell(I_t, Y_t)\right] = \overline{\ell}(\hat{p}_t, Y_t)\end{aligned}$$

- $\Rightarrow$ the effect of random fluctuations is comparable with the bound on the expected regret:

$$\sum_{t=1}^{n}\left(\ell(I_t, Y_t) - \overline{\ell}(\hat{p}_t, Y_t)\right) \le \sqrt{2\overline{L}_n \ln(1/\delta)} + \frac{2\sqrt{2}}{3}\ln(1/\delta).$$

Small losses

- Previous "small-loss" bound:

$$2\sqrt{2L_n^* \ln N} + \kappa \ln N$$

- Random fluctuations: add $\sqrt{n/2 \ln(1/\delta)}$ – too big!
- Bernstein's inequality uses the "predictable variance" to bound the fluctuations
- Bound on the "predictable variance":

$$\mathbb{E}_t\left[\left(\ell(I_t, Y_t) - \bar{\ell}(\hat{p}_t, Y_t)\right)^2\right] = \mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] - \bar{\ell}^2(\hat{p}_t, Y_t)$$

$$\leq \quad \mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] \leq \mathbb{E}_t\left[\ell(I_t, Y_t)\right] = \bar{\ell}(\hat{p}_t, Y_t)$$

- $\Rightarrow$ the effect of random fluctuations is comparable with the bound on the expected regret:

$$\sum_{t=1}^{n} \left(\ell(I_t, Y_t) - \bar{\ell}(\hat{p}_t, Y_t)\right) \leq \sqrt{2\bar{L}_n \ln(1/\delta)} + \frac{2\sqrt{2}}{3} \ln(1/\delta).$$

Small losses

- Previous "small-loss" bound:

$$2\sqrt{2L_n^* \ln N} + \kappa \ln N$$

- Random fluctuations: add $\sqrt{n/2 \ln(1/\delta)}$ – too big!
- Bernstein's inequality uses the "predictable variance" to bound the fluctuations
- Bound on the "predictable variance":

$$\mathbb{E}_t\left[\left(\ell(I_t, Y_t) - \bar{\ell}(\hat{p}_t, Y_t)\right)^2\right] = \mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] - \bar{\ell}^2(\hat{p}_t, Y_t)$$
$$\leq \quad \mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] \leq \mathbb{E}_t\left[\ell(I_t, Y_t)\right] = \bar{\ell}(\hat{p}_t, Y_t)$$

- $\Rightarrow$ the effect of random fluctuations is comparable with the bound on the expected regret:

$$\sum_{t=1}^{n}\left(\ell(I_t, Y_t) - \bar{\ell}(\hat{p}_t, Y_t)\right) \leq \sqrt{2\bar{L}_n \ln(1/\delta)} + \frac{2\sqrt{2}}{3}\ln(1/\delta).$$

SMALL LOSSES

- Previous "small-loss" bound:

$$2\sqrt{2L_n^* \ln N} + \kappa \ln N$$

- Random fluctuations: add $\sqrt{n/2 \ln(1/\delta)}$ – too big!
- Bernstein's inequality uses the "predictable variance" to bound the fluctuations
- Bound on the "predictable variance":

$$\begin{aligned}\mathbb{E}_t\left[(\ell(I_t, Y_t) - \overline{\ell}(\hat{p}_t, Y_t))^2\right] &= \mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] - \overline{\ell}^2(\hat{p}_t, Y_t) \\ \leq \quad &\mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] \leq \mathbb{E}_t\left[\ell(I_t, Y_t)\right] = \overline{\ell}(\hat{p}_t, Y_t)\end{aligned}$$

- $\Rightarrow$ the effect of random fluctuations is comparable with the bound on the expected regret:

$$\sum_{t=1}^{n} (\ell(I_t, Y_t) - \overline{\ell}(\hat{p}_t, Y_t)) \leq \sqrt{2\overline{L}_n \ln(1/\delta)} + \frac{2\sqrt{2}}{3} \ln(1/\delta).$$

Small losses

- Previous "small-loss" bound:

$$2\sqrt{2L_n^* \ln N} + \kappa \ln N$$

- Random fluctuations: add $\sqrt{n/2 \ln(1/\delta)}$ – too big!
- Bernstein's inequality uses the "predictable variance" to bound the fluctuations
- Bound on the "predictable variance":

$$\begin{aligned} \mathbb{E}_t\left[(\ell(I_t, Y_t) - \bar{\ell}(\hat{p}_t, Y_t))^2\right] &= \mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] - \bar{\ell}^2(\hat{p}_t, Y_t) \\ \le \quad & \mathbb{E}_t\left[\ell(I_t, Y_t)^2\right] \le \mathbb{E}_t\left[\ell(I_t, Y_t)\right] = \bar{\ell}(\hat{p}_t, Y_t) \end{aligned}$$

- $\Rightarrow$ the effect of random fluctuations is comparable with the bound on the expected regret:

$$\sum_{t=1}^{n} (\ell(I_t, Y_t) - \bar{\ell}(\hat{p}_t, Y_t)) \le \sqrt{2\bar{L}_n \ln(1/\delta)} + \frac{2\sqrt{2}}{3} \ln(1/\delta).$$

Outline

Follow the Leader

- Does it work?
- Take $N = 2$:

$$\begin{array}{ll} \ell(1, y_t): & \frac{1}{2}, 0, 1, 0, 1, 0, \ldots \\ \ell(2, y_t): & \frac{1}{2}, 1, 0, 1, 0, 1, \ldots \end{array}$$

- Choices:

$$\begin{array}{ll} \ell(1, y_t): & \frac{1}{2}^{L_{11}=.5}, 0^{L_{12}=.5}, 1^{L_{13}=1.5}, 0^{L_{14}=1.5}, 1^{L_{15}=2.5}, 0, \ldots \\ \ell(2, y_t): & \frac{1}{2}^{L_{21}=.5}, 1^{L_{22}=1.5}, 0^{L_{22}=1.5}, 1^{L_{23}=2.5}, 0^{L_{24}=2.5}, 1, \ldots \end{array}$$

- $\Rightarrow \hat{L}_n = n - 2 + 0.5$, whilst $L_{in} \le n/2$, $i = 1, 2$,

$$\hat{L}_n - L_n^* \ge n/2 - 1.5$$

FOLLOW THE PERTURBED LEADER [HANNAN, 1957]

- Follow the perturbed leader (randomized fictitous play):

$$
\begin{aligned}
I_t &= \operatorname{argmin}_{i=1,\dots,N} \left(L_{i,t-1} + Z_{it} \right), \\
Z_t &\sim f(\cdot), \quad \text{i.i.d.}
\end{aligned}
$$

- Goal: develop bound on $\overline{L}_n$!
- Relate to BEH:

$$
\hat{I}_t = \operatorname{argmin}_{i \in \underline{N}} \left(L_{i,t} + Z_{i,t} \right).
$$

FOLLOW THE PERTURBED LEADER [HANNAN, 1957]

- Follow the perturbed leader (randomized fictitous play):

$$\begin{aligned} I_t &= \operatorname{argmin}_{i=1,\ldots,N}\left(L_{i,t-1} + Z_{it}\right), \\ Z_t &\sim f(\cdot), \quad \text{i.i.d.} \end{aligned}$$

- Goal: develop bound on $\overline{L}_n$!
- Relate to BEH:

$$\hat{I}_t = \operatorname{argmin}_{i \in \underline{N}}\left(L_{i,t} + Z_{i,t}\right).$$

Follow the perturbed leader [Hannan, 1957]

- Follow the perturbed leader (randomized fictitous play):

$$
\begin{aligned}
I_t &= \mathrm{argmin}_{i=1,\ldots,N} \left(L_{i,t-1} + Z_{it} \right), \\
Z_t &\sim f(\cdot), \quad \text{i.i.d.}
\end{aligned}
$$

- Goal: develop bound on $\overline{L}_n$!
- Relate to BEH:

$$
\hat{I}_t = \mathrm{argmin}_{i \in \underline{N}} \left(L_{i,t} + Z_{i,t} \right).
$$

FPL BOUND

THEOREM (FPL BOUND [KALAI AND VEMPALA, 2003])

Let $\ell : \underline{N} \times \mathcal{Y} \to [0,1]$ *and consider FPL! Let*

$$Z_t \sim f(\cdot), \quad f(z) = (\tfrac{\eta}{2})^N e^{-\eta \|z\|_1}.$$

Then

$$\mathbb{E}\left[\hat{L}_n\right] \leq e^{\eta}\left(\mathbb{E}\left[L_n^*\right] + \frac{2(1+\ln N)}{\eta}\right).$$

Choose

$$\eta = \min\left\{1, \sqrt{\frac{2(1+\ln N)}{(e-1)L_n^*}}\right\}.$$

Then

$$\mathbb{E}\left[L_n\right] - \mathbb{E}\left[L_n^*\right] \leq 2\sqrt{2L_n^*(e-1)(1+\ln N)} + 2(e+1)(1+\ln N).$$

Tracking the best expert [Herbster and Warmuth, 1998]

- Discrete prediction problem
- Want to compete with 'compound action sets':

$$B_{n,m} = \{(i_1, \ldots, i_n) \mid s(i_1, \ldots, i_n) \leq m\},$$

  where $s(i_1, \ldots, i_n) = \sum_{t=2}^{n} \mathbb{I}_{\{i_{t-1} \neq i_t\}}$ is the number of switches.
- Shorthand notation $i_{1:n} = (i_1, \ldots, i_n)$
- Regret:

$$R_{n,m} \stackrel{\text{def}}{=} \sum_{t=1}^{n} \ell(I_t, y_t) - \min_{i_{1:n} \in B_{n,m}} \sum_{t=1}^{n} \ell(i_t, y_t).$$

- Instead we use $\overline{R}_{n,m}$, where

$$\overline{R}_{n,m} \stackrel{\text{def}}{=} \max_{i_{1:n} \in B_{n,m}} \overline{R}(i_{1:n}),\ \overline{R}(i_{1:n}) \stackrel{\text{def}}{=} \sum_{t=1}^{n} \overline{\ell}(p_t, y_t) - \sum_{t=1}^{n} \ell(i_t, y_t).$$

Tracking the best expert [Herbster and Warmuth, 1998]

- Discrete prediction problem
- Want to compete with 'compound action sets':

$$B_{n,m} = \{(i_1,\dots,i_n) \,|\, s(i_1,\dots,i_n) \le m\},$$

where $s(i_1,\dots,i_n) = \sum_{t=2}^{n} \mathbb{I}_{\{i_{t-1} \ne i_t\}}$ is the number of switches.

- Shorthand notation $i_{1:n} = (i_1,\dots,i_n)$
- Regret:

$$R_{n,m} \stackrel{\text{def}}{=} \sum_{t=1}^{n} \ell(I_t, y_t) - \min_{i_{1:n} \in B_{n,m}} \sum_{t=1}^{n} \ell(i_t, y_t).$$

- Instead we use $\overline{R}_{n,m}$, where

$$\overline{R}_{n,m} \stackrel{\text{def}}{=} \max_{i_{1:n} \in B_{n,m}} \overline{R}(i_{1:n}),\; \overline{R}(i_{1:n}) \stackrel{\text{def}}{=} \sum_{t=1}^{n} \overline{\ell}(p_t, y_t) - \sum_{t=1}^{n} \ell(i_t, y_t).$$

TRACKING THE BEST EXPERT [HERBSTER AND WARMUTH, 1998]

- Discrete prediction problem
- Want to compete with 'compound action sets':

$$B_{n,m} = \{(i_1,\ldots,i_n) \mid s(i_1,\ldots,i_n) \le m\},$$

where $s(i_1,\ldots,i_n) = \sum_{t=2}^{n} \mathbb{I}_{\{i_{t-1} \neq i_t\}}$ is the number of switches.

- Shorthand notation $i_{1:n} = (i_1,\ldots,i_n)$
- Regret:

$$R_{n,m} \stackrel{\text{def}}{=} \sum_{t=1}^{n} \ell(I_t, y_t) - \min_{i_{1:n} \in B_{n,m}} \sum_{t=1}^{n} \ell(i_t, y_t).$$

- Instead we use $\overline{R}_{n,m}$, where

$$\overline{R}_{n,m} \stackrel{\text{def}}{=} \max_{i_{1:n} \in B_{n,m}} \overline{R}(i_{1:n}),\ \overline{R}(i_{1:n}) \stackrel{\text{def}}{=} \sum_{t=1}^{n} \overline{\ell}(p_t, y_t) - \sum_{t=1}^{n} \ell(i_t, y_t).$$

TRACKING THE BEST EXPERT [HERBSTER AND WARMUTH, 1998]

- Discrete prediction problem
- Want to compete with 'compound action sets':

$$B_{n,m} = \{(i_1, \ldots, i_n) \mid s(i_1, \ldots, i_n) \le m\},$$

  where $s(i_1, \ldots, i_n) = \sum_{t=2}^{n} \mathbb{I}_{\{i_{t-1} \neq i_t\}}$ is the number of switches.
- Shorthand notation $i_{1:n} = (i_1, \ldots, i_n)$
- Regret:

$$R_{n,m} \stackrel{\text{def}}{=} \sum_{t=1}^{n} \ell(I_t, y_t) - \min_{i_{1:n} \in B_{n,m}} \sum_{t=1}^{n} \ell(i_t, y_t).$$

- Instead we use $\overline{R}_{n,m}$, where

$$\overline{R}_{n,m} \stackrel{\text{def}}{=} \max_{i_{1:n} \in B_{n,m}} \overline{R}(i_{1:n}),\ \overline{R}(i_{1:n}) \stackrel{\text{def}}{=} \sum_{t=1}^{n} \overline{\ell}(p_t, y_t) - \sum_{t=1}^{n} \ell(i_t, y_t).$$

TRACKING THE BEST EXPERT [HERBSTER AND WARMUTH, 1998]

- Discrete prediction problem
- Want to compete with 'compound action sets':

$$B_{n,m} = \{(i_1, \ldots, i_n) \mid s(i_1, \ldots, i_n) \leq m\},$$

where $s(i_1, \ldots, i_n) = \sum_{t=2}^{n} \mathbb{I}_{\{i_{t-1} \neq i_t\}}$ is the number of switches.

- Shorthand notation $i_{1:n} = (i_1, \ldots, i_n)$
- Regret:

$$R_{n,m} \stackrel{\text{def}}{=} \sum_{t=1}^{n} \ell(I_t, y_t) - \min_{i_{1:n} \in B_{n,m}} \sum_{t=1}^{n} \ell(i_t, y_t).$$

- Instead we use $\overline{R}_{n,m}$, where

$$\overline{R}_{n,m} \stackrel{\text{def}}{=} \max_{i_{1:n} \in B_{n,m}} \overline{R}(i_{1:n}), \; \overline{R}(i_{1:n}) \stackrel{\text{def}}{=} \sum_{t=1}^{n} \overline{\ell}(p_t, y_t) - \sum_{t=1}^{n} \ell(i_t, y_t).$$

Outline

Randomized EWA applied to tracking problems

- Action set: $B_{n,m}$.
- We always select a compound, but just play the next primitive action.
- Previous regret bound gives:

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\ln(|B_{n,m}|)}.$$

- $M = |B_{n,m}| \leq ?$
- $M = \sum_{k=0}^{m} \binom{n-1}{k} N(N-1)^k$.
- $M \leq N^{m+1} \exp\left((n-1)H\left(\frac{m}{n-1}\right)\right)$,
  $H : [0,1] \to \mathbb{R}$, $H(x) = -x\ln x - (1-x)\ln(1-x)$.
- Hence

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\left((m+1)\ln N + (n-1)H\left(\frac{m}{n-1}\right)\right)}.$$

- Problem: randomized EWA is not efficient (M weights!)

Randomized EWA applied to tracking problems

- Action set: $B_{n,m}$.
- We always select a compound, but just play the next primitive action.
- Previous regret bound gives:

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\ln(|B_{n,m}|)}.$$

- $M = |B_{n,m}| \leq ?$
- $M = \sum_{k=0}^{m} \binom{n-1}{k} N(N-1)^k$.
- $M \leq N^{m+1} \exp\left((n-1) H\left(\frac{m}{n-1}\right) \right)$,

$$H : [0,1] \to \mathbb{R},\ H(x) = -x \ln x - (1-x)\ln(1-x).$$

- Hence

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\left((m+1)\ln N + (n-1) H\left(\frac{m}{n-1}\right)\right)}.$$

- Problem: randomized EWA is not efficient (M weights!)

Randomized EWA applied to tracking problems

- Action set: $B_{n,m}$.
- We always select a compound, but just play the next primitive action.
- Previous regret bound gives:

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\ln(|B_{n,m}|)}.$$

- $M = |B_{n,m}| \leq ?$
- $M = \sum_{k=0}^{m} \binom{n-1}{k} N(N-1)^k$.
- $M \leq N^{m+1} \exp\left((n-1) H\left(\frac{m}{n-1}\right) \right)$,

$$H : [0,1] \to \mathbb{R},\ H(x) = -x \ln x - (1-x)\ln(1-x).$$

- Hence

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\left((m+1)\ln N + (n-1) H\left(\frac{m}{n-1}\right) \right)}.$$

- Problem: randomized EWA is not efficient (M weights!)

Randomized EWA applied to tracking problems

- Action set: $B_{n,m}$.
- We always select a compound, but just play the next primitive action.
- Previous regret bound gives:

$$\overline{R}_{n,m} \le \sqrt{\frac{n}{2} \ln(|B_{n,m}|)}.$$

- $M = |B_{n,m}| \le ?$
- $M = \sum_{k=0}^{m} \binom{n-1}{k} N(N-1)^k$.
- $M \le N^{m+1} \exp\left((n-1) H\left(\frac{m}{n-1}\right) \right)$,

  $H : [0,1] \to \mathbb{R}$, $H(x) = -x \ln x - (1-x)\ln(1-x)$.
- Hence

$$\overline{R}_{n,m} \le \sqrt{\frac{n}{2}\left((m+1)\ln N + (n-1) H\left(\frac{m}{n-1}\right)\right)}.$$

- Problem: randomized EWA is not efficient (M weights!)

Randomized EWA applied to tracking problems

- Action set: $B_{n,m}$.
- We always select a compound, but just play the next primitive action.
- Previous regret bound gives:

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\ln(|B_{n,m}|)}.$$

- $M = |B_{n,m}| \leq ?$
- $M = \sum_{k=0}^{m} \binom{n-1}{k} N(N-1)^k$.
- $M \leq N^{m+1} \exp\left((n-1)H\left(\frac{m}{n-1}\right)\right)$,

$$H : [0,1] \to \mathbb{R},\ H(x) = -x\ln x - (1-x)\ln(1-x).$$

- Hence

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\left((m+1)\ln N + (n-1)H\left(\frac{m}{n-1}\right)\right)}.$$

- Problem: randomized EWA is not efficient (M weights!)

Randomized EWA applied to tracking problems

- Action set: $B_{n,m}$.
- We always select a compound, but just play the next primitive action.
- Previous regret bound gives:

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2} \ln(|B_{n,m}|)}.$$

- $M = |B_{n,m}| \leq ?$
- $M = \sum_{k=0}^{m} \binom{n-1}{k} N(N-1)^k$.
- $M \leq N^{m+1} \exp\left((n-1) H\left(\frac{m}{n-1}\right) \right)$,

$$H : [0,1] \to \mathbb{R}, \; H(x) = -x \ln x - (1-x)\ln(1-x).$$

- Hence

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\left((m+1)\ln N + (n-1) H\left(\frac{m}{n-1}\right)\right)}.$$

- Problem: randomized EWA is not efficient (M weights!)

Randomized EWA applied to tracking problems

- Action set: $B_{n,m}$.
- We always select a compound, but just play the next primitive action.
- Previous regret bound gives:

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2} \ln(|B_{n,m}|)}.$$

- $M = |B_{n,m}| \leq ?$
- $M = \sum_{k=0}^{m} \binom{n-1}{k} N(N-1)^k$.
- $M \leq N^{m+1} \exp\left((n-1) H\left(\frac{m}{n-1}\right) \right)$,

  $H : [0,1] \to \mathbb{R},\ H(x) = -x \ln x - (1-x)\ln(1-x)$.
- Hence

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2} \left((m+1) \ln N + (n-1) H\left(\frac{m}{n-1}\right) \right)}.$$

- Problem: randomized EWA is not efficient (M weights!)

Randomized EWA applied to tracking problems

- Action set: $B_{n,m}$.
- We always select a compound, but just play the next primitive action.
- Previous regret bound gives:

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2} \ln(|B_{n,m}|)}.$$

- $M = |B_{n,m}| \leq ?$
- $M = \sum_{k=0}^{m} \binom{n-1}{k} N(N-1)^k$.
- $M \leq N^{m+1} \exp(\,(n-1)H\left(\frac{m}{n-1}\right)\,)$,
  $H : [0,1] \to \mathbb{R},\ H(x) = -x \ln x - (1-x)\ln(1-x)$.
- Hence

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2}\left((m+1)\ln N + (n-1)H\left(\frac{m}{n-1}\right)\right)}.$$

- Problem: randomized EWA is not efficient (M weights!)

FIXED-SHARE FORECASTER (FSF)

Initialize: $w_{i0} = 1/N$.

1. Draw expert index I_t from $w_{i,t-1}/\sum_{j=1}^{N} w_{j,t-1}$.
2. Send I_t to Environment
3. Receive y_t and losses $(\ell(i, y_t))_i$ from Environment
4. Update weights:
5. $v_{it} := w_{i,t-1} e^{-\eta\ell(i,y_t)}$
6. $V_t := \sum_{j=1}^{N} v_{jt}$
7. $w_{it} := \frac{\alpha}{N} V_t + (1-\alpha) v_{it}$

Regret bound for FSF

Theorem ([Herbster and Warmuth, 1998])

Consider a discrete prediction problem and pick any sequence $y_{1:n}$. *For any compound action* $i_{1:n}$,

$$\overline{R}(i_{1:n}) \leq \frac{s(i_{1:n})+1}{\eta} \ln N + \frac{1}{\eta} \ln \left(\frac{1}{\alpha^{s(i_{1:n})}(1-\alpha)^{n-s(i_{1:n})}} \right) + \frac{\eta}{8} n.$$

For $0 \leq m \leq n$, $\alpha = m/(n-1)$, *with a specific choice of* $\eta = \eta(n, m, N)$,

$$\overline{R}_{n,m} \leq \sqrt{\frac{n}{2} \left((m+1) \ln N + (n-1) H\left(\frac{m}{n-1}\right) + \ln \left(\frac{1}{1-\frac{m}{n-1}} \right) \right)}.$$

OUTLINE

Variable-share forecaster: Algorithm

Variable-share forecaster (VSF)

Initialize: $w_{i0} = 1/N$.

1. Draw primitive action I_t from $w_{i,t-1} / \sum_{j=1}^N w_{j,t-1}$.
2. Observe y_t, losses $\ell(i, y_t)$ (suffers loss $\ell(I_t, y_t)$)
3. Compute $v_{it} = w_{i,t-1} e^{-\eta \ell(i,y_t)}$
4. Let $w_{it} = \frac{1}{N-1} \sum_{j \neq i} \left(1 - (1-\alpha)^{\ell(j,y_t)}\right) v_{jt} + (1-\alpha)^{\ell(i,y_t)} v_{it}$.

   // If loss of current action is small, stay at it, otherwise encourage switching!

- Result: For binary losses, $\frac{n - s(i_{1:n}) - 1}{\eta} \ln \frac{1}{1-\alpha}$ is replaced by $s(i_{1:n}) + \frac{1}{\eta} L(i_{1:n}) \ln \frac{1}{1-\alpha}$.
- Small complexity ($s(i_{1:n})$) and small loss ($L(i_{1:n})$): big win

VARIABLE-SHARE FORECASTER: ALGORITHM

VARIABLE-SHARE FORECASTER (VSF)

Initialize: $w_{i0} = 1/N$.

1. Draw primitive action I_t from $w_{i,t-1} / \sum_{j=1}^{N} w_{j,t-1}$.
2. Observe y_t, losses $\ell(i, y_t)$ (suffers loss $\ell(I_t, y_t)$)
3. Compute $v_{it} = w_{i,t-1} e^{-\eta \ell(i,y_t)}$
4. Let $w_{it} = \frac{1}{N-1} \sum_{j \neq i} \left(1 - (1-\alpha)^{\ell(j,y_t)}\right) v_{jt} + (1-\alpha)^{\ell(i,y_t)} v_{it}$.

   // If loss of current action is small, stay at it, otherwise encourage switching!

- Result: For binary losses, $\frac{n - s(i_{1:n}) - 1}{\eta} \ln \frac{1}{1-\alpha}$ is replaced by $s(i_{1:n}) + \frac{1}{\eta} L(i_{1:n}) \ln \frac{1}{1-\alpha}$.
- Small complexity ($s(i_{1:n})$) and small loss ($L(i_{1:n})$): big win

VARIABLE-SHARE FORECASTER (VSF)

Initialize: $w_{i0} = 1/N$.

1. Draw primitive action I_t from $w_{i,t-1} / \sum_{j=1}^{N} w_{j,t-1}$.
2. Observe y_t, losses $\ell(i, y_t)$ (suffers loss $\ell(I_t, y_t)$)
3. Compute $v_{it} = w_{i,t-1} e^{-\eta \ell(i,y_t)}$
4. Let $w_{it} = \frac{1}{N-1} \sum_{j \neq i} \left(1 - (1-\alpha)^{\ell(j,y_t)}\right) v_{jt} + (1-\alpha)^{\ell(i,y_t)} v_{it}$.

   // If loss of current action is small, stay at it, otherwise encourage switching!

- Result: For binary losses, $\frac{n-s(i_{1:n})-1}{\eta} \ln \frac{1}{1-\alpha}$ is replaced by $s(i_{1:n}) + \frac{1}{\eta} L(i_{1:n}) \ln \frac{1}{1-\alpha}$.
- Small complexity ($s(i_{1:n})$) and small loss ($L(i_{1:n})$): big win

OUTLINE

OTHER EXAMPLES

- Tree experts (side info); e.g. [D.P. Helmbold, 1997]
- Shortest path FPL: [Kalai and Vempala, 2003]; additive losses
- Shortest path EWA [György et al., 2005]; compression – best scalar quantizers [György et al., 2004]
- Shortest path tracking
- Further applications:
  - Sequential allocation
  - Motion planning
  - Opponent modelling

Other examples

- Tree experts (side info); e.g. [D.P. Helmbold, 1997]
- Shortest path FPL: [Kalai and Vempala, 2003]; additive losses
- Shortest path EWA [György et al., 2005]; compression – best scalar quantizers [György et al., 2004]
- Shortest path tracking
- Further applications:
  - Sequential allocation
  - Motion planning (robot arms)
  - Opponent modeling in poker

OTHER EXAMPLES

- Tree experts (side info); e.g. [D.P. Helmbold, 1997]
- Shortest path FPL: [Kalai and Vempala, 2003]; additive losses
- Shortest path EWA [György et al., 2005]; compression – best scalar quantizers [György et al., 2004]
- Shortest path tracking
- Further applications:
  - Sequential allocation
  - Motion planning (robot arms)
  - Opponent modelling in poker

Other examples

- Tree experts (side info); e.g. [D.P. Helmbold, 1997]
- Shortest path FPL: [Kalai and Vempala, 2003];
  additive losses
- Shortest path EWA [György et al., 2005];
  compression – best scalar quantizers [György et al., 2004]
- Shortest path tracking
- Further applications:
  - Sequential allocation
  - Motion planning (robot arms)
  - Opponent modeling in poker

Other examples

- Tree experts (side info); e.g. [D.P. Helmbold, 1997]
- Shortest path FPL: [Kalai and Vempala, 2003]; additive losses
- Shortest path EWA [György et al., 2005]; compression – best scalar quantizers [György et al., 2004]
- Shortest path tracking
- Further applications:
  - Sequential allocation
  - Motion planning (robot arms)
  - Opponent modeling in poker

OTHER EXAMPLES

- Tree experts (side info); e.g. [D.P. Helmbold, 1997]
- Shortest path FPL: [Kalai and Vempala, 2003]; additive losses
- Shortest path EWA [György et al., 2005]; compression – best scalar quantizers [György et al., 2004]
- Shortest path tracking
- Further applications:
  - Sequential allocation
  - Motion planning (robot arms)
  - Opponent modeling in poker

OTHER EXAMPLES

- Tree experts (side info); e.g. [D.P. Helmbold, 1997]
- Shortest path FPL: [Kalai and Vempala, 2003]; additive losses
- Shortest path EWA [György et al., 2005]; compression – best scalar quantizers [György et al., 2004]
- Shortest path tracking
- Further applications:
  - Sequential allocation
  - Motion planning (robot arms)
  - Opponent modeling in poker

Other examples

- Tree experts (side info); e.g. [D.P. Helmbold, 1997]
- Shortest path FPL: [Kalai and Vempala, 2003]; additive losses
- Shortest path EWA [György et al., 2005]; compression – best scalar quantizers [György et al., 2004]
- Shortest path tracking
- Further applications:
  - Sequential allocation
  - Motion planning (robot arms)
  - Opponent modeling in poker

Bandit setting

Bandit setting

Feedback is restricted to the expert (action) chosen

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t:

1. Expert predictions are revealed to Learner
2. Learner chooses expert $I_t \in \{1, \ldots, N\}$
3. Environment generates outcome Y_t
4. Learner receives $\ell_t = \ell(I_t, Y_t)$ from Environment
5. $t := t + 1$; go to Step 1

No feedback is received for actions $i \neq I_t$

BANDIT SETTING

BANDIT SETTING

Feedback is restricted to the expert (action) chosen

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$
At time t:

1. Expert predictions are revealed to Learner
2. Learner chooses expert $I_t \in \{1, \ldots, N\}$
3. Environment generates outcome Y_t
4. Learner receives $\ell_t = \ell(I_t, Y_t)$ from Environment
5. $t := t + 1$; go to Step 1

No feedback is received for actions $i \neq I_t$

BANDIT SETTING

Feedback is restricted to the expert (action) chosen

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$
At time t:

1. Expert predictions are revealed to Learner
2. Learner chooses expert $I_t \in \{1, \ldots, N\}$
3. Environment generates outcome Y_t
4. Learner receives $\ell_t = \ell(I_t, Y_t)$ from Environment
5. $t := t + 1$; go to Step 1

No feedback is received for actions $i \neq I_t$

BANDIT SETTING

Feedback is restricted to the expert (action) chosen

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$

At time t:

1. Expert predictions are revealed to Learner
2. Learner chooses expert $I_t \in \{1, \dots, N\}$
3. Environment generates outcome Y_t
4. Learner receives $\ell_t = \ell(I_t, Y_t)$ from Environment
5. $t := t + 1$; go to Step 1

No feedback is received for actions $i \neq I_t$

BANDIT SETTING

Feedback is restricted to the expert (action) chosen

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$
At time t:

1. Expert predictions are revealed to Learner
2. Learner chooses expert $I_t \in \{1, \ldots, N\}$
3. Environment generates outcome Y_t
4. Learner receives $\ell_t = \ell(I_t, Y_t)$ from Environment
5. $t := t + 1$; go to Step 1

No feedback is received for actions $i \neq I_t$

BANDIT SETTING

Feedback is restricted to the expert (action) chosen

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$
At time t:

1. Expert predictions are revealed to Learner
2. Learner chooses expert $I_t \in \{1, \ldots, N\}$
3. Environment generates outcome Y_t
4. Learner receives $\ell_t = \ell(I_t, Y_t)$ from Environment
5. $t := t + 1$; go to Step 1

No feedback is received for actions $i \neq I_t$

Bandit setting

Feedback is restricted to the expert (action) chosen

Protocol

Initialization: Algorithm gets N and ℓ, $t := 1$
At time t:

1. Expert predictions are revealed to Learner
2. Learner chooses expert $I_t \in \{1, \ldots, N\}$
3. Environment generates outcome Y_t
4. Learner receives $\ell_t = \ell(I_t, Y_t)$ from Environment
5. $t := t + 1$; go to Step 1

No feedback is received for actions $i \neq I_t$

BANDIT SETTING

Feedback is restricted to the expert (action) chosen

PROTOCOL

Initialization: Algorithm gets N and ℓ, $t := 1$
At time t:

1. Expert predictions are revealed to Learner
2. Learner chooses expert $I_t \in \{1, \ldots, N\}$
3. Environment generates outcome Y_t
4. Learner receives $\ell_t = \ell(I_t, Y_t)$ from Environment
5. $t := t + 1$; go to Step 1

No feedback is received for actions $i \neq I_t$

OBSERVATION

That we do not receive feedback for all experts does not mean that no "appropriate" feedback can be derived for them!

- Consider randomized EWA and expected losses
- Only $\mathbb{E}\left[\ell(i, Y_t)\right]$ matters:
  When ℓ, ℓ' are such that for $\forall i, t$: $\mathbb{E}\left[\ell(i, Y_t)\right] = \mathbb{E}\left[\ell'(i, Y_t)\right]$ and $0 \le \ell, \ell' \le 1$, then the bounds on the expected regret of EWA are the same for both ℓ and ℓ'.
- Idea: construct feedback for the unselected experts

OBSERVATION

That we do not receive feedback for all experts does not mean that no "appropriate" feedback can be derived for them!

- Consider randomized EWA and expected losses
- Only $\mathbb{E}[\ell(i, Y_t)]$ matters:
  When ℓ, ℓ' are such that for $\forall i, t$: $\mathbb{E}[\ell(i, Y_t)] = \mathbb{E}[\ell'(i, Y_t)]$ and $0 \leq \ell, \ell' \leq 1$, then the bounds on the expected regret of EWA are the same for both ℓ and ℓ'.
- Idea: construct feedback for the unselected experts

OBSERVATION

That we do not receive feedback for all experts does not mean that no "appropriate" feedback can be derived for them!

- Consider randomized EWA and expected losses
- Only $\mathbb{E}[\ell(i, Y_t)]$ matters:
  When ℓ, ℓ' are such that for $\forall i, t$: $\mathbb{E}[\ell(i, Y_t)] = \mathbb{E}[\ell'(i, Y_t)]$ and $0 \leq \ell, \ell' \leq 1$, then the bounds on the expected regret of EWA are the same for both ℓ and ℓ'.
- Idea: construct feedback for the unselected experts

OBSERVATION

That we do not receive feedback for all experts does not mean that no “appropriate” feedback can be derived for them!

- Consider randomized EWA and expected losses
- Only $\mathbb{E}\left[\ell(i, Y_t)\right]$ matters:
  When ℓ, ℓ' are such that for $\forall i, t$: $\mathbb{E}\left[\ell(i, Y_t)\right] = \mathbb{E}\left[\ell'(i, Y_t)\right]$ and $0 \leq \ell, \ell' \leq 1$, then the bounds on the expected regret of EWA are the same for both ℓ and ℓ'.
- Idea: construct feedback for the unselected experts

OUTLINE

Feedback for all experts!

- Work with gains: $g(i, Y_t) = 1 - \ell(i, Y_t)$
- Proposed feedback:

$$\tilde{g}(i, Y_t) = \begin{cases} \frac{g(i,Y_t)}{p_{i,t}}, & \text{if } I_t = i \\ 0 & \text{otherwise.} \end{cases}$$

- Compact notation: $\tilde{g}(i, Y_t) = \mathbb{I}_{\{I_t = i\}} g(I_t, Y_t)/p_{I_t,t}$.
- Prop: If $p_{j,t} > 0$ holds $\forall j \in \underline{N}$, where p_{jt} depends on $g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1})$, then $\forall i \in \underline{N}$,

$$\mathbb{E}\left[\tilde{g}(i, Y_t) | g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1}), Y_t\right] = g(i, Y_t).$$

- 1st problem: as $p_{it} \to 0$, $\tilde{g}(i, Y_t) \to \infty$ (not $\tilde{g}(i, Y_t) \leq 1$!).
- Idea: prevent $p_{it} \to 0$ by adding exploration!
- 2nd problem: If $\sum_{t=1}^{n} g'(i, Y_t) \leq \sum_{t=1}^{n} g(i, Y_t)$, starvation may happen.

Feedback for all experts!

- Work with gains: $g(i, Y_t) = 1 - \ell(i, Y_t)$
- Proposed feedback:

$$\tilde{g}(i, Y_t) = \begin{cases} \frac{g(i, Y_t)}{p_{i,t}}, & \text{if } I_t = i \\ 0 & \text{otherwise.} \end{cases}$$

- Compact notation: $\tilde{g}(i, Y_t) = \mathbb{I}_{\{I_t = i\}} g(I_t, Y_t) / p_{I_t, t}$.
- Prop: If $p_{j,t} > 0$ holds $\forall j \in \underline{N}$, where p_{jt} depends on $g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1})$, then $\forall i \in \underline{N}$,

$$\mathbb{E}\left[\tilde{g}(i, Y_t) | g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1}), Y_t\right] = g(i, Y_t).$$

- 1st problem: as $p_{it} \to 0$, $\tilde{g}(i, Y_t) \to \infty$ (not $\tilde{g}(i, Y_t) \leq 1$!).
- Idea: prevent $p_{it} \to 0$ by adding exploration!
- 2nd problem: If $\sum_{t=1}^{n} g'(i, Y_t) \leq \sum_{t=1}^{n} g(i, Y_t)$, starvation may happen.

FEEDBACK FOR ALL EXPERTS!

- Work with gains: $g(i, Y_t) = 1 - \ell(i, Y_t)$
- Proposed feedback:

$$\tilde{g}(i, Y_t) = \begin{cases} \frac{g(i, Y_t)}{p_{i,t}}, & \text{if } I_t = i \\ 0 & \text{otherwise.} \end{cases}$$

- Compact notation: $\tilde{g}(i, Y_t) = \mathbb{I}_{\{I_t = i\}} g(I_t, Y_t)/p_{I_t,t}$.
- Prop: If $p_{j,t} > 0$ holds $\forall j \in \underline{N}$, where p_{jt} depends on $g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1})$, then $\forall i \in \underline{N}$,

$$\mathbb{E}\left[\tilde{g}(i, Y_t) | g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1}), Y_t\right] = g(i, Y_t).$$

- 1st problem: as $p_{it} \to 0$, $\tilde{g}(i, Y_t) \to \infty$ (not $\tilde{g}(i, Y_t) \leq 1$!).
- Idea: prevent $p_{it} \to 0$ by adding exploration!
- 2nd problem: If $\sum_{t=1}^{n} g'(i, Y_t) \leq \sum_{t=1}^{n} g(i, Y_t)$, starvation may happen.

Feedback for all experts!

- Work with gains: $g(i, Y_t) = 1 - \ell(i, Y_t)$
- Proposed feedback:

$$\tilde{g}(i, Y_t) = \begin{cases} \frac{g(i, Y_t)}{p_{i,t}}, & \text{if } I_t = i \\ 0 & \text{otherwise.} \end{cases}$$

- Compact notation: $\tilde{g}(i, Y_t) = \mathbb{I}_{\{I_t = i\}} g(I_t, Y_t)/p_{I_t,t}$.
- Prop: If $p_{j,t} > 0$ holds $\forall j \in \underline{N}$, where p_{jt} depends on $g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1})$, then $\forall i \in \underline{N}$,

$$\mathbb{E}\left[\tilde{g}(i, Y_t) | g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1}), Y_t\right] = g(i, Y_t).$$

- 1st problem: as $p_{it} \to 0$, $\tilde{g}(i, Y_t) \to \infty$ (not $\tilde{g}(i, Y_t) \leq 1$!).
- Idea: prevent $p_{it} \to 0$ by adding exploration!
- 2nd problem: If $\sum_{t=1}^{n} g'(i, Y_t) \leq \sum_{t=1}^{n} g(i, Y_t)$, starvation may happen.

Feedback for all experts!

- Work with gains: $g(i, Y_t) = 1 - \ell(i, Y_t)$
- Proposed feedback:

$$\tilde{g}(i, Y_t) = \begin{cases} \frac{g(i, Y_t)}{p_{i,t}}, & \text{if } I_t = i \\ 0 & \text{otherwise.} \end{cases}$$

- Compact notation: $\tilde{g}(i, Y_t) = \mathbb{I}_{\{I_t = i\}} g(I_t, Y_t)/p_{I_t,t}$.
- Prop: If $p_{j,t} > 0$ holds $\forall j \in \underline{N}$, where p_{jt} depends on $g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1})$, then $\forall i \in \underline{N}$,

$$\mathbb{E}\left[\tilde{g}(i, Y_t) | g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1}), Y_t\right] = g(i, Y_t).$$

- 1st problem: as $p_{it} \to 0$, $\tilde{g}(i, Y_t) \to \infty$ (not $\tilde{g}(i, Y_t) \leq 1$!).
- Idea: prevent $p_{it} \to 0$ by adding exploration!
- 2nd problem: If $\sum_{t=1}^{n} g'(i, Y_t) \leq \sum_{t=1}^{n} g(i, Y_t)$, starvation may happen.

Feedback for all experts!

- Work with gains: $g(i, Y_t) = 1 - \ell(i, Y_t)$
- Proposed feedback:

$$\tilde{g}(i, Y_t) = \begin{cases} \frac{g(i,Y_t)}{p_{i,t}}, & \text{if } I_t = i \\ 0 & \text{otherwise.} \end{cases}$$

- Compact notation: $\tilde{g}(i, Y_t) = \mathbb{I}_{\{I_t = i\}} g(I_t, Y_t)/p_{I_t,t}$.
- Prop: If $p_{j,t} > 0$ holds $\forall j \in \underline{N}$, where p_{jt} depends on $g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1})$, then $\forall i \in \underline{N}$,

$$\mathbb{E}\left[\tilde{g}(i, Y_t) | g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1}), Y_t\right] = g(i, Y_t).$$

- 1st problem: as $p_{it} \to 0$, $\tilde{g}(i, Y_t) \to \infty$ (not $\tilde{g}(i, Y_t) \leq 1$!).
- Idea: prevent $p_{it} \to 0$ by adding exploration!
- 2nd problem: If $\sum_{t=1}^{n} g'(i, Y_t) \leq \sum_{t=1}^{n} g(i, Y_t)$, starvation may happen.

Feedback for all experts!

- Work with gains: $g(i, Y_t) = 1 - \ell(i, Y_t)$
- Proposed feedback:

$$\tilde{g}(i, Y_t) = \begin{cases} \frac{g(i, Y_t)}{p_{i,t}}, & \text{if } I_t = i \\ 0 & \text{otherwise.} \end{cases}$$

- Compact notation: $\tilde{g}(i, Y_t) = \mathbb{I}_{\{I_t = i\}} g(I_t, Y_t)/p_{I_t,t}$.
- Prop: If $p_{j,t} > 0$ holds $\forall j \in \underline{N}$, where p_{jt} depends on $g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1})$, then $\forall i \in \underline{N}$,

$$\mathbb{E}\left[\tilde{g}(i, Y_t) | g(I_1, Y_1), \ldots, g(I_{t-1}, Y_{t-1}), Y_t\right] = g(i, Y_t).$$

- 1st problem: as $p_{it} \to 0$, $\tilde{g}(i, Y_t) \to \infty$ (not $\tilde{g}(i, Y_t) \leq 1$!).
- Idea: prevent $p_{it} \to 0$ by adding exploration!
- 2nd problem: If $\sum_{t=1}^{n} g'(i, Y_t) \leq \sum_{t=1}^{n} g(i, Y_t)$, starvation may happen.

Algorithm

Exp3.P($\eta, \beta, \gamma > 0$) [Auer et al., 2002a]

Initialize: $w_{i0} = 1$, $p_{i1} = 1/N$

A time t do:

1. Compute action selection probabilities:

$$p_{it} = (1 - \gamma) \frac{w_{i,t-1}}{\sum_{j=1}^{N} w_{j,t-1}} + \gamma \frac{1}{N}.$$

2. Select $I_t \sim p_{.,t}$
3. Compute inflated feedbacks:

$$g'(i, Y_t) = \tilde{g}(i, Y_t) + \frac{\beta}{p_{it}}.$$

4. Update weights:

$$w_{it} = w_{i,t-1} e^{\eta g'(i, Y_t)}$$

EXP3.P($\eta, \beta, \gamma > 0$) [AUER ET AL., 2002A]

Initialize: $w_{i0} = 1$, $p_{i1} = 1/N$

A time t do:

1. Compute action selection probabilities:

$$p_{it} = (1-\gamma)\frac{w_{i,t-1}}{\sum_{j=1}^{N} w_{j,t-1}} + \gamma\frac{1}{N}.$$

2. Select $I_t \sim p_{\cdot,t}$
3. Compute inflated feedbacks:

$$g'(i, Y_t) = \tilde{g}(i, Y_t) + \frac{\beta}{p_{it}}.$$

4. Update weights:

$$w_{it} = w_{i,t-1}e^{\eta g'(i,Y_t)}$$

Algorithm

Exp3.P($\eta, \beta, \gamma > 0$) [Auer et al., 2002a]

Initialize: $w_{i0} = 1$, $p_{i1} = 1/N$

A time t do:

1. Compute action selection probabilities:

$$p_{it} = (1-\gamma)\frac{w_{i,t-1}}{\sum_{j=1}^{N} w_{j,t-1}} + \gamma\frac{1}{N}.$$

2. Select $I_t \sim p_{\cdot,t}$
3. Compute inflated feedbacks:

$$g'(i, Y_t) = \tilde{g}(i, Y_t) + \frac{\beta}{p_{it}}.$$

4. Update weights:

$$w_{it} = w_{i,t-1} e^{\eta g'(i,Y_t)}$$

EXP3.P($\eta, \beta, \gamma > 0$) [AUER ET AL., 2002A]

Initialize: $w_{i0} = 1$, $p_{i1} = 1/N$

A time t do:

1. Compute action selection probabilities:

$$p_{it} = (1-\gamma)\frac{w_{i,t-1}}{\sum_{j=1}^{N} w_{j,t-1}} + \gamma\frac{1}{N}.$$

2. Select $I_t \sim p_{\cdot,t}$
3. Compute inflated feedbacks:

$$g'(i, Y_t) = \tilde{g}(i, Y_t) + \frac{\beta}{p_{it}}.$$

4. Update weights:

$$w_{it} = w_{i,t-1}\, e^{\eta g'(i,Y_t)}$$

Regret bound for Exp3.P

Theorem (Regret of Exp3.P [Auer et al., 2002a])

Consider Exp3.P. Let $0 < \delta < 1$ *arbitrary,* $n \geq 8N \ln(N/\delta)$,

$$\gamma \leq \frac{1}{2}, \quad 0 < \eta \leq \frac{\gamma}{2N}, \quad \sqrt{\frac{1}{nN} \ln \frac{N}{\delta}} \leq \beta \leq 1.$$

Then with probability at least $1 - \delta$*, we have*

$$\hat{L}_n - L_n^* \leq n(\gamma + \eta(1+\beta)N) + \frac{\ln N}{\eta} + 2nN\beta.$$

Choosing β *as its lower bound,* η *as its upper bound,* $\gamma = 4N\beta/(3+\beta)$*, then*

$$\hat{L}_n - \min_i L_{in} \leq \frac{11}{2}\sqrt{nN \ln \frac{N}{\delta}} + \frac{\ln N}{2}.$$

Note: $n \geq 8N \ln(N/\delta)$ ensures that γ (2nd part) is at most $1/2$.

Regret bound for Exp3.P

Theorem (Regret of Exp3.P [Auer et al., 2002a])

Consider Exp3.P. Let $0 < \delta < 1$ *arbitrary,* $n \geq 8N \ln(N/\delta)$,

$$\gamma \leq \frac{1}{2}, \quad 0 < \eta \leq \frac{\gamma}{2N}, \quad \sqrt{\frac{1}{nN} \ln \frac{N}{\delta}} \leq \beta \leq 1.$$

Then with probability at least $1 - \delta$*, we have*

$$\hat{L}_n - L_n^* \leq n(\gamma + \eta(1+\beta)N) + \frac{\ln N}{\eta} + 2nN\beta.$$

Choosing β *as its lower bound,* η *as its upper bound,* $\gamma = 4N\beta/(3+\beta)$*, then*

$$\hat{L}_n - \min_i L_{in} \leq \frac{11}{2} \sqrt{nN \ln \frac{N}{\delta}} + \frac{\ln N}{2}.$$

Note: $n \geq 8N \ln(N/\delta)$ ensures that γ (2nd part) is at most $1/2$.

Why use inflated values?

- $\beta = 0 \Rightarrow$ Exp3
- The expected regret of Exp3 is

$$O(\sqrt{nN \ln N}).$$

- Problem:

  No high-probability bound on the actual regret!

- Inflated values $\Rightarrow$
  estimated gains are upper bounds on the true gains with high probability

Why use inflated values?

- $\beta = 0 \Rightarrow$ Exp3
- The expected regret of Exp3 is

$$O(\sqrt{nN \ln N}).$$

- Problem:

  No high-probability bound on the actual regret!

- Inflated values $\Rightarrow$
  estimated gains are upper bounds on the true gains with high probability

Why use inflated values?

- $\beta = 0 \Rightarrow$ Exp3
- The expected regret of Exp3 is

$$O(\sqrt{nN \ln N}).$$

- Problem:

No high-probability bound on the actual regret!

- Inflated values $\Rightarrow$
estimated gains are upper bounds on the true gains with high probability

Why use inflated values?

- $\beta = 0 \Rightarrow$ Exp3
- The expected regret of Exp3 is

$$O(\sqrt{nN \ln N}).$$

- Problem:

  No high-probability bound on the actual regret!

- Inflated values $\Rightarrow$
  estimated gains are upper bounds on the true gains with high probability

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

LOSSES VS. GAINS

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

LOSSES VS. GAINS

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action
- Work with losses: [Cesa-Bianchi et al., 2005]

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

LOSSES VS. GAINS

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

LOSSES VS. GAINS

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

Losses vs. Gains

- The algorithm could work with losses, too!
- Gains:
  - When an action becomes bad, its weight ceases to grow
  - When an action becomes good, its weight grows
- Losses:
  - When an action becomes bad, its weight decreases
  - When an action becomes good (loss=0), its weight is not decreased
- Working with losses:
  - Better when an action becomes bad
  - Quickly reacts then
- Working with gains:
  - Better when an action becomes good
  - Warning: Takes N/γ steps to find out about this action!
- Work with losses: [Cesa-Bianchi et al., 2005]

FULL INFORMATION VS. PARTIAL INFORMATION

- Full information, discrete predictions:

$$\frac{R_n}{n} \leq C_1 \sqrt{\frac{\ln N}{n}}$$

- Bandit setting:

$$\frac{R_n}{n} \leq C_2 \sqrt{\frac{N \ln N}{n}} = C_2 \sqrt{\frac{\ln N}{\frac{n}{N}}}$$

- In the bandit case, it takes N-times more to drive the average one-step regret down to some level as it takes in the full information case
- Makes sense!!

Full information vs. partial information

- Full information, discrete predictions:

$$\frac{R_n}{n} \le C_1 \sqrt{\frac{\ln N}{n}}$$

- Bandit setting:

$$\frac{R_n}{n} \le C_2 \sqrt{\frac{N \ln N}{n}} = C_2 \sqrt{\frac{\ln N}{\frac{n}{N}}}$$

- In the bandit case, it takes N-times more to drive the average one-step regret down to some level as it takes in the full information case
- Makes sense!!

FULL INFORMATION VS. PARTIAL INFORMATION

- Full information, discrete predictions:

$$\frac{R_n}{n} \leq C_1 \sqrt{\frac{\ln N}{n}}$$

- Bandit setting:

$$\frac{R_n}{n} \leq C_2 \sqrt{\frac{N \ln N}{n}} = C_2 \sqrt{\frac{\ln N}{\frac{n}{N}}}$$

- In the bandit case, it takes N-times more to drive the average one-step regret down to some level as it takes in the full information case
- Makes sense!!

Full information vs. partial information

- Full information, discrete predictions:

$$\frac{R_n}{n} \leq C_1 \sqrt{\frac{\ln N}{n}}$$

- Bandit setting:

$$\frac{R_n}{n} \leq C_2 \sqrt{\frac{N \ln N}{n}} = C_2 \sqrt{\frac{\ln N}{\frac{n}{N}}}$$

- In the bandit case, it takes N-times more to drive the average one-step regret down to some level as it takes in the full information case
- Makes sense!!

Lower bound

Theorem (Minimax lower bound [Auer et al., 2002a])

Fix $n, N \geq 1$. Let $n > N/(4\ln(4/3))$ and assume that the output space $\mathcal{Y}$ has at least 2^N elements. Then there exists a loss function such that

$$\sup_{y_{1:n}} \left(\mathbb{E}\left[\hat{L}_n\right] - \min_{i=1,\ldots,N} L_{in} \right) \geq \frac{\sqrt{2}-1}{\sqrt{23\ln(4/3)}} \sqrt{nN}.$$

Proof.

- One uniform random variable decides which action should be the best.
- Payoffs are Bernoulli $(1/2, 1/2)$, except for the best arm, which is Bernoulli $(1/2-\epsilon, 1/2+\epsilon)$.
- Making ϵ sufficiently small (given n, N) makes the N arms hard enough to distinguish in n trials.

Lower bound

Theorem (Minimax lower bound [Auer et al., 2002a])

Fix $n, N \geq 1$. Let $n > N/(4\ln(4/3))$ and assume that the output space $\mathcal{Y}$ has at least 2^N elements. Then there exists a loss function such that

$$\sup_{y_{1:n}} \left(\mathbb{E}\left[\hat{L}_n\right] - \min_{i=1,\ldots,N} L_{in} \right) \geq \frac{\sqrt{2}-1}{\sqrt{23\ln(4/3)}} \sqrt{nN}.$$

Proof.

- One uniform random variable decides which action should be the best.
- Payoffs are Bernoulli $(1/2, 1/2)$, except for the best arm, which is Bernoulli $(1/2-\epsilon, 1/2+\epsilon)$.
- Making ϵ sufficiently small (given n, N) makes the N arms hard enough to distinguish in n trials.

LOWER BOUND

THEOREM (MINIMAX LOWER BOUND [AUER ET AL., 2002A])

Fix $n, N \geq 1$. Let $n > N/(4\ln(4/3))$ and assume that the output space $\mathcal{Y}$ has at least 2^N elements. Then there exists a loss function such that

$$\sup_{y_{1:n}} \left(\mathbb{E}\left[\hat{L}_n\right] - \min_{i=1,\ldots,N} L_{in} \right) \geq \frac{\sqrt{2}-1}{\sqrt{23\ln(4/3)}} \sqrt{nN}.$$

PROOF.

- One uniform random variable decides which action should be the best.
- Payoffs are Bernoulli $(1/2, 1/2)$, except for the best arm, which is Bernoulli $(1/2-\epsilon, 1/2+\epsilon)$.
- Making ϵ sufficiently small (given n, N) makes the N arms hard enough to distinguish in n trials.

□

Lower bound

Theorem (Minimax lower bound [Auer et al., 2002a])

Fix $n, N \geq 1$*. Let* $n > N/(4\ln(4/3))$ *and assume that the output space* $\mathcal{Y}$ *has at least* 2^N *elements. Then there exists a loss function such that*

$$\sup_{y_{1:n}} \left(\mathbb{E}\left[\hat{L}_n\right] - \min_{i=1,\dots,N} L_{in} \right) \geq \frac{\sqrt{2}-1}{\sqrt{23\ln(4/3)}} \sqrt{nN}.$$

Proof.

- One uniform random variable decides which action should be the best.
- Payoffs are Bernoulli $(1/2, 1/2)$, except for the best arm, which is Bernoulli $(1/2-\epsilon, 1/2+\epsilon)$.
- Making ϵ sufficiently small (given n, N) makes the N arms hard enough to distinguish in n trials.

Lower bound

Theorem (Minimax lower bound [Auer et al., 2002a])

Fix $n, N \geq 1$. Let $n > N/(4\ln(4/3))$ and assume that the output space $\mathcal{Y}$ has at least 2^N elements. Then there exists a loss function such that

$$\sup_{y_{1:n}} \left(\mathbb{E}\left[\hat{L}_n\right] - \min_{i=1,\ldots,N} L_{in} \right) \geq \frac{\sqrt{2}-1}{\sqrt{23\ln(4/3)}} \sqrt{nN}.$$

Proof.

- One uniform random variable decides which action should be the best.
- Payoffs are Bernoulli $(1/2, 1/2)$, except for the best arm, which is Bernoulli $(1/2 - \epsilon, 1/2 + \epsilon)$.
- Making ϵ sufficiently small (given n, N) makes the N arms hard enough to distinguish in n trials.

□

- Examples:
  - Dynamic pricing: $h(I_t, Y_t) = (Y_t - I_t)\mathbb{I}_{\{Y_t \geq I_t\}} + Y_t \mathbb{I}_{\{Y_t < I_t\}}$ – we sell; if our price I_t is higher than Y_t, we loose Y_t, otherwise loose $Y_t - I_t$
    We get price of customer only if product was sold
  - Apple (product) testing: $\mathcal{Y} = J = \{\text{“rotten''}, \text{“good for sale''}\}$, $\ell(i, Y_t) = a\,\mathbb{I}_{\{i=\text{“rotten''}\}} + b\,\mathbb{I}_{\{i\neq\text{“rotten''}, Y_t=\text{“rotten''}\}}$
    Only apples declared as “rotten” are tested
  - Bandit problems, routing in a network, cost-efficient prediction (“revealing actions” are costly)
- Result: Minimax regret bound: $(Nn)^{2/3}(\ln N)^{1/3}$
- Matching lower bound

[Mertens et al., 1994, Rustichini, 1999, Mannor and Shimkin, 2003, Piccolboni and Schindelhauer, 2001]

Partial monitoring [Cesa-Bianchi et al., 2006]

- Examples:
  - Dynamic pricing: $h(I_t, Y_t) = (Y_t - I_t)\mathbb{I}_{\{Y_t \geq I_t\}} + Y_t \mathbb{I}_{\{Y_t < I_t\}}$ – we sell; if our price I_t is higher than Y_t, we loose Y_t, otherwise loose $Y_t - I_t$
    We get price of customer only if product was sold
  - Apple (product) testing: $\mathcal{Y} = J = \{\text{"rotten"}, \text{"good for sale"}\}$, $\ell(i, Y_t) = a\,\mathbb{I}_{\{i=\text{"rotten"}\}} + b\,\mathbb{I}_{\{i \neq \text{"rotten"}, Y_t = \text{"rotten"}\}}$
    Only apples declared as "rotten" are tested
  - Bandit problems, routing in a network, cost-efficient prediction ("revealing actions" are costly)
- Result: Minimax regret bound: $(Nn)^{2/3}(\ln N)^{1/3}$
- Matching lower bound

[Mertens et al., 1994, Rustichini, 1999, Mannor and Shimkin, 2003, Piccolboni and Schindelhauer, 2001]

PARTIAL MONITORING [CESA-BIANCHI ET AL., 2006]

- Examples:
  - Dynamic pricing: $h(I_t, Y_t) = (Y_t - I_t)\mathbb{I}_{\{Y_t \geq I_t\}} + Y_t \mathbb{I}_{\{Y_t < I_t\}}$
    – we sell; if our price I_t is higher than Y_t, we loose Y_t, otherwise loose $Y_t - I_t$
    We get price of customer only if product was sold
  - Apple (product) testing: $\mathcal{Y} = J = \{\text{“rotten''}, \text{“good for sale''}\}$,
    $\ell(i, Y_t) = a\,\mathbb{I}_{\{i=\text{“rotten''}\}} + b\,\mathbb{I}_{\{i\neq\text{“rotten''}, Y_t=\text{“rotten''}\}}$
    Only apples declared as “rotten” are tested
  - Bandit problems, routing in a network, cost-efficient prediction (“revealing actions” are costly)
- Result: Minimax regret bound: $(Nn)^{2/3}(\ln N)^{1/3}$
- Matching lower bound

[Mertens et al., 1994, Rustichini, 1999, Mannor and Shimkin, 2003, Piccolboni and Schindelhauer, 2001]

PARTIAL MONITORING [CESA-BIANCHI ET AL., 2006]

- Examples:
  - Dynamic pricing: $h(I_t, Y_t) = (Y_t - I_t)\mathbb{I}_{\{Y_t \geq I_t\}} + Y_t \mathbb{I}_{\{Y_t < I_t\}}$
    – we sell; if our price I_t is higher than Y_t, we loose Y_t, otherwise loose $Y_t - I_t$
    We get price of customer only if product was sold
  - Apple (product) testing: $\mathcal{Y} = J = \{\text{“rotten”}, \text{“good for sale”}\}$, $\ell(i, Y_t) = a\,\mathbb{I}_{\{i=\text{“rotten”}\}} + b\,\mathbb{I}_{\{i \neq \text{“rotten”}, Y_t = \text{“rotten”}\}}$
    Only apples declared as “rotten” are tested
  - Bandit problems, routing in a network, cost-efficient prediction (“revealing actions” are costly)
- Result: Minimax regret bound: $(Nn)^{2/3}(\ln N)^{1/3}$
- Matching lower bound

[Mertens et al., 1994, Rustichini, 1999, Mannor and Shimkin, 2003, Piccolboni and Schindelhauer, 2001]

R L
A I

PARTIAL MONITORING [CESA-BIANCHI ET AL., 2006]

- Examples:
  - Dynamic pricing: $h(I_t, Y_t) = (Y_t - I_t)\mathbb{I}_{\{Y_t \geq I_t\}} + Y_t \mathbb{I}_{\{Y_t < I_t\}}$ – we sell; if our price I_t is higher than Y_t, we loose Y_t, otherwise loose $Y_t - I_t$
    We get price of customer only if product was sold
  - Apple (product) testing: $\mathcal{Y} = J = \{\text{"rotten"}, \text{"good for sale"}\}$, $\ell(i, Y_t) = a\,\mathbb{I}_{\{i=\text{"rotten"}\}} + b\,\mathbb{I}_{\{i \neq \text{"rotten"}, Y_t = \text{"rotten"}\}}$
    Only apples declared as "rotten" are tested
  - Bandit problems, routing in a network, cost-efficient prediction ("revealing actions" are costly)
- Result: Minimax regret bound: $(Nn)^{2/3}(\ln N)^{1/3}$
- Matching lower bound

[Mertens et al., 1994, Rustichini, 1999, Mannor and Shimkin, 2003, Piccolboni and Schindelhauer, 2001]

Partial monitoring [Cesa-Bianchi et al., 2006]

- Examples:
  - Dynamic pricing: $h(I_t, Y_t) = (Y_t - I_t)\mathbb{I}_{\{Y_t \geq I_t\}} + Y_t \mathbb{I}_{\{Y_t < I_t\}}$ – we sell; if our price I_t is higher than Y_t, we loose Y_t, otherwise loose $Y_t - I_t$
    We get price of customer only if product was sold
  - Apple (product) testing: $\mathcal{Y} = J = \{\text{“rotten”}, \text{“good for sale”}\}$, $\ell(i, Y_t) = a\,\mathbb{I}_{\{i=\text{“rotten”}\}} + b\,\mathbb{I}_{\{i \neq \text{“rotten”}, Y_t = \text{“rotten”}\}}$
    Only apples declared as “rotten” are tested
  - Bandit problems, routing in a network, cost-efficient prediction (“revealing actions” are costly)
- Result: Minimax regret bound: $(Nn)^{2/3}(\ln N)^{1/3}$
- Matching lower bound

[Mertens et al., 1994, Rustichini, 1999, Mannor and Shimkin, 2003, Piccolboni and Schindelhauer, 2001]

RL AI

Partial monitoring [Cesa-Bianchi et al., 2006]

- Examples:
  - Dynamic pricing: $h(I_t, Y_t) = (Y_t - I_t)\mathbb{I}_{\{Y_t \geq I_t\}} + Y_t \mathbb{I}_{\{Y_t < I_t\}}$
    – we sell; if our price I_t is higher than Y_t, we loose Y_t, otherwise loose $Y_t - I_t$
    We get price of customer only if product was sold
  - Apple (product) testing: $\mathcal{Y} = J = \{\text{"rotten"}, \text{"good for sale"}\}$,
    $\ell(i, Y_t) = a\,\mathbb{I}_{\{i=\text{"rotten"}\}} + b\,\mathbb{I}_{\{i \neq \text{"rotten"}, Y_t = \text{"rotten"}\}}$
    Only apples declared as "rotten" are tested
  - Bandit problems, routing in a network, cost-efficient prediction ("revealing actions" are costly)
- Result: Minimax regret bound: $(Nn)^{2/3}(\ln N)^{1/3}$
- Matching lower bound

[Mertens et al., 1994, Rustichini, 1999, Mannor and Shimkin, 2003, Piccolboni and Schindelhauer, 2001]

Conclusions

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
  - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
  - Self-confident forecaster [Auer et al., 2002b]
  - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

CONCLUSIONS

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
  - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
  - Self-confident forecaster [Auer et al., 2002b]
  - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

CONCLUSIONS

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
  - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
  - Self-confident forecaster [Auer et al., 2002b]
  - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

CONCLUSIONS

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
  - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
  - Self-confident forecaster [Auer et al., 2002b]
  - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

CONCLUSIONS

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
  - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
  - Self-confident forecaster [Auer et al., 2002b]
  - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

CONCLUSIONS

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
  - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
  - Self-confident forecaster [Auer et al., 2002b]
  - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

CONCLUSIONS

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
  - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
  - Self-confident forecaster [Auer et al., 2002b]
  - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

CONCLUSIONS

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
  - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
  - Self-confident forecaster [Auer et al., 2002b]
  - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

Conclusions

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
  - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
  - Self-confident forecaster [Auer et al., 2002b]
  - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
  - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

Conclusions

- Algorithms might work outside of their intended domain
- Increasing robustness: larger learning rates, multiplicative updates, tracking, ...
- Caveat: Algorithms might become too agressive (risky)
- Side information
    - Gradient based linear forecaster [Grove et al., 2001, Warmuth and Jagota, 1997]
    - Self-confident forecaster [Auer et al., 2002b]
    - Ridge regression forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
    - Vovk-Azoury-Warmuth (VAW) forecaster [Vovk, 2001, Azoury and Warmuth, 2001]
    - ...
- Great book: [Cesa-Bianchi and Lugosi, 2006]

References I

Angluin, D. (1988).
Queries and concept learning.
Journal of Machine Learning, 2:319–342.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. (2002a).
The nonstochastic multiarmed bandit problem.
SIAM Journal on Computing, 32:48–77.

Auer, P., Cesa-Bianchi, N., and Gentile, C. (2002b).
Adaptive and self-confident on-line learning algorithms.
Journal of Computer and System Sciences, 64(1).

Azoury, K. and Warmuth, M. (2001).
Relative loss bounds for on-line density estimation with the exponential family of distributions.
Machine Learning, 43(3):211–246.

Barzdin, Y. and Freivalds, R. (1972).
On the prediction of general recursive functions.
Soviet Mathematics (Doklady), 13:1224–1228.

Cesa-Bianchi, N. and Lugosi, G. (2006).
Prediction, Learning, and Games.
Cambridge University Press, New York, NY, USA.

Cesa-Bianchi, N., Lugosi, G., and Stoltz, G. (2006).
Regret minimization under partial monitoring.
Mathematics of Operations Research, 31:562–580.

References II

Cesa-Bianchi, N., Mansour, Y., and Stoltz, G. (2005).
Improved second-order bounds for prediction with expert advice.
In *Proceedings of the 18th Annual Conference on Learning Theory (COLT-2005)*, pages 217–232. Springer.

D.P. Helmbold, R. S. (1997).
Predicting nearly as well as the best pruning of a decision tree.
Machine Learning, 27:51–68.

Grove, A., Littlestone, N., and Schuurmans, D. (2001).
General convergence results for linear discriminant updates.
Machine Learning, 43(3):173–210.

György, A., Linder, T., and Lugosi, G. (2004).
Efficient algorithms and minimax bounds for zero-delay lossy source coding.
IEEE Transactions on Signal Processing, 52:2337–2347.

György, A., Linder, T., and Lugosi, G. (2005).
Tracking the best of many experts.
pages 204–216.

Hannan, J. (1957).
Approximation to bayes risk in repeated play.
Contributions to the theory of games, 3(97–139).

Herbster, M. and Warmuth, M. (1998).
Tracking the best expert.
Machine Learning, 32:151–178.

References III

Kalai, A. and Vempala, S. (2003).
Efficient algorithms for the online decision problem.
In *Proceedings of the 16th Annual Conference on Learning Theory*, pages 26–40. Springer.

Littlestone, N. and Warmuth, M. (1994).
The weighted majority algorithm.
Information and Computation, 108:212–261.

Mannor, S. and Shimkin, N. (2003).
On-line learning with imperfect monitoring.
In *Proceedings of the 16th Annual Conference on Learning Theory*, pages 552–567. Springer.

Mertens, J.-F., Sorin, S., and Zamir, S. (1994).
Repeated games.
CORE Discussion paper, no. 9420, 9421, 9422, Louvain-la-Neuve, Belgium.

Piccolboni, A. and Schindelhauer, C. (2001).
Discrete prediction games with arbitrary feedback and loss.
In *14th Annual Conference on Computational Learning Theory*, pages 208–223. Springer.

Rustichini, A. (1999).
Minimizing regret: The general case.
Games and Economic Behavior, 29:224–243.

Vovk, V. (2001).
Competitive on-line statistics.
International Statistical Review, 69:213–248.

REFERENCES IV

Warmuth, M. and Jagota, A. (1997).
Continuous and discrete-time nonlinear gradient descent: Relative loss bounds and convergence.
In *Electronic proceedings of the 5th International Symposium on Artificial Intelligence and Mathematics*.