Bandit-style Geometric decision algorithm against an Adaptive adversary

Amirreza Velaei

Sharif University of Technology

December 31, 2024

Outline

Online Geometric Optimization in the Bandit Setting

- **Paper Title**: Online Geometric Optimization in the Bandit Setting Against an Adaptive Adversary
- **Authors :** H. Brendan McMahan & Avrim Blum
- **Published in:** 2004
- **Conference:** Learning Theory, Springer Berlin Heidelberg

Introduction to Online Optimization

Definition: Online optimization involves making a sequence of decisions based on incoming data, where each decision is made without knowledge of future data.

Two fundamental assumptions in online optimization:

1. **Bunded Losses:** The losses determined by an adversary should not be allowed to be unbounded.
2. **Bounded Decision Set:** The decision set must be somehow bounded and/or structured, though not necessarily finite.

Prediction from Expert Advice

- **Prediction from Expert Advice** is a framework in online learning and decision-making.
- Involves making sequential predictions by aggregating advice from a set of experts.
- The goal is to perform nearly as well as the best expert in hindsight.

Key Components

1. **Experts:** Sources that provide predictions or advice.
2. **Learner (Main Character):** Aggregates expert advice to make decisions.
3. **Feedback:** Information about the actual outcome to update future predictions.
4. **Objective:** Minimize the difference between the learner's performance and the best expert's performance.

Family Members as Experts

- Main character decides whether to take an umbrella each day.
- Three family members act as **experts** providing daily weather predictions, the father, mother, and brother.

Figure: Family Members as Experts

Example Scenario

Table: Daily Weather Predictions and Outcomes

Day	Father	Mother	Brother	Actual Weather
1	No Rain	No Rain	Rain	Rain

Example Scenario

Table: Daily Weather Predictions and Outcomes

Day	Father	Mother	Brother	Actual Weather
1	No Rain	No Rain	Rain	Rain
2	Rain	No Rain	Rain	No Rain

Example Scenario

Table: Daily Weather Predictions and Outcomes

Day	Father	Mother	Brother	Actual Weather
1	No Rain	No Rain	Rain	Rain
2	Rain	No Rain	Rain	No Rain
3	Rain	No Rain	No Rain	Rain

Example Scenario

Table: Daily Weather Predictions and Outcomes

Day	Father	Mother	Brother	Actual Weather
1	No Rain	No Rain	Rain	Rain
2	Rain	No Rain	Rain	No Rain
3	Rain	No Rain	No Rain	Rain
4	No Rain	Rain	No Rain	No Rain

Example Scenario

Table: Daily Weather Predictions and Outcomes

Day	Father	Mother	Brother	Actual Weather
1	No Rain	No Rain	Rain	Rain
2	Rain	No Rain	Rain	No Rain
3	Rain	No Rain	No Rain	Rain
4	No Rain	Rain	No Rain	No Rain
5	Rain	No Rain	No Rain	Rain

Example Scenario

Table: Daily Weather Predictions and Outcomes

Day	**Father**	**Mother**	**Brother**	**Actual Weather**
1	No Rain	No Rain	Rain	Rain
2	Rain	No Rain	Rain	No Rain
3	Rain	No Rain	No Rain	Rain
4	No Rain	Rain	No Rain	No Rain
5	Rain	No Rain	No Rain	Rain
Cost	2	4	3	-

Weighted Majority Algorithm

- **Mechanism:**
  - Assigns a weight to each expert based on their past performance.
  - Aggregates predictions by considering these weights.
  - Updates weights multiplicatively based on the correctness of experts' predictions.
- **Applications:**
  - Ensemble learning in machine learning.
  - Financial decision-making.

How Weighted Majority Works

Algorithm Steps:

1. **Initialization:** Assign equal weights to all experts.
2. **Prediction**

$$\text{Prediction} = \text{sign}\left(\sum_{i=1}^{N} w_t(i) \cdot \text{Prediction}_t(i)\right)$$

3. **Update Weights:** After observing the outcome, update weights:

$$w_{t+1}(i) = \begin{cases} w_t(i) & \text{if expert } i \text{ was correct} \\ w_t(i) \cdot \varepsilon & \text{if expert } i \text{ was incorrect} \end{cases}$$

4. **Iteration:** Repeat the prediction and update steps for each round.

Parameters:

- N: Number of experts.
- $\varepsilon \in (0, 1)$: Penalty factor for incorrect experts.

Regret Bound for Weighted Majority

Lemma : Denote by M_t the number of mistakes the algorithm makes until time t, and by $M_t(i)$ the number of mistakes made by expert i until time t. Then, for any expert $i \in [N]$ we have

$$M_T \leq 2(1+\varepsilon)M_T(i) + \frac{2\log N}{\varepsilon}$$

Corollary : The regret of the WM algorithm is bounded by

$$M_T \leq 2M_T(i^*) + O(\sqrt{M_T(i^*) logN})$$

where i^* is the best expert.

Proof: Just let $\varepsilon^* = \sqrt{\frac{logN}{M_T(i^*)}}$.

Proof:

- Let $\phi_t = \sum_{i=1}^{N} w_i(t)$. Note that $\phi_1 = N$.
- If the prediction is wrong, then $\phi_{t+1} \leq \frac{1}{2}\phi_t(1-\varepsilon) + \frac{1}{2}\phi_t$.
- Thus $\phi_t \leq \phi_1(1-\varepsilon)^{M_t} = N(1-\varepsilon)^{M_t}$.
- By definition, $w_T(i) = (1-\varepsilon)^{M_T(i)}$. Also $w_t(i) \leq \phi_t$.
- $(1-\varepsilon)^{M_T(i)} \leq N(1-\varepsilon)^{M_T} \rightarrow M_T(i) log(1-\varepsilon) \leq logN + M_T log(1-\varepsilon)$.
- Using the fact that $-x - x^2 \leq log(1-x) \leq -x$ for $x \in (0,1)$, we get:

$$-M_T(i)(\varepsilon + \varepsilon^2) \leq logN - M_T\frac{\varepsilon}{2} \rightarrow M_T \leq 2(1+\varepsilon)M_T(i) + \frac{2logN}{\varepsilon}$$

Randomized Weighted Majority Algorithm

- **Algorithm Steps:**
  1. **Initialization:** Set $w_1(i) = 1$ for all experts.
  2. **Probability Assignment:**

$$P_t(i) = \frac{w_t(i)}{\sum_{j=1}^{N} w_t(j)}$$

  3. **Expert Selection:** Choose expert i with probability $P_i(t)$.
  4. **Prediction and Update:** Make prediction based on selected expert and update weights as in Weighted Majority.
- **Better Regret Bound:**

$$\mathbb{E}[M_T] \leq (1+\varepsilon)M_T(i^*) + \frac{logN}{\varepsilon}$$

RWM Regret Bound Proof

Proof: Let $\phi_t = \sum_{i=1}^{N} w_i(t)$ and $\tilde{m}_t$ be an indicator variable for the event that the prediction is wrong at time t and $\tilde{m}_t(i) = 1$ if expert i is wrong at time t.

- Note that

$$\phi_{t+1} = \sum_{i=1}^{N} w_i(t)(1 - \varepsilon \tilde{m}_t(i)) = \phi_t(1 - \varepsilon \sum_{i=1}^{N} P_t(i)\tilde{m}_t(i))$$
$$= \phi_t(1 - \varepsilon \mathbb{E}[\tilde{m}_t]) \leq \phi_t e^{-\varepsilon \mathbb{E}[\tilde{m}_t]}$$

- With the same argument as in the WM proof, we get:

$$(1-\varepsilon)^{M_T(i)} \leq N e^{-\varepsilon M_T}$$
$$\to M_T(i) log(1-\varepsilon) \leq log N - \varepsilon \mathbb{E}[M_T]$$
$$\to -M_T(\varepsilon + \varepsilon^2) \leq log N - \varepsilon \mathbb{E}[M_T]$$
$$\to \mathbb{E}[M_T] \leq (1+\varepsilon) M_T(i^*) + \frac{log N}{\varepsilon}$$

Introduction to Multi-Armed Bandit Algorithms

- **Definition:** A framework for making a sequence of decisions under uncertainty, aiming to maximize cumulative rewards.
- **Key Concepts:**
  - **Exploration:** Trying different actions to gather more information.
  - **Exploitation:** Selecting the best-known action to maximize immediate reward.
- **Types of Bandit Problems:**
  - **Stochastic Bandits:** Rewards are drawn from fixed probability distributions.
  - **Contextual Bandits:** Incorporates contextual information to make more informed decisions.
- **Applications:**
  - Recommendation Systems
  - Clinical Trials
  - Online Advertising

Bandit Algorithms in Online Advertising

- **Use Case:** Optimizing Ad Selection to Maximize CTR
- **How It Works:**
  - Each ad variant is considered an arm of the bandit.
  - The algorithm dynamically selects which ad to display based on past performance.
  - Balances exploration (trying new ads) with exploitation (showing top-performing ads).

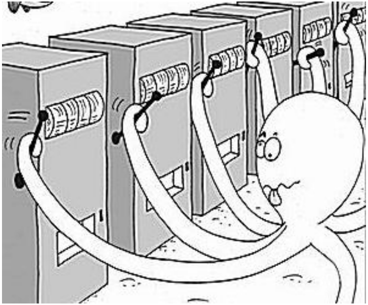


Figure: Bandit Setting

Online Optimization vs. Bandit Algorithms

Both online optimization and bandit algorithms involve sequential decisions, but differ in feedback and objectives.

Online Optimization

- **Full Feedback:** Receives complete information about all possible actions after each decision.
- **Objective:** Minimize cumulative loss compared to the best fixed decision in hindsight.

Bandit Algorithms

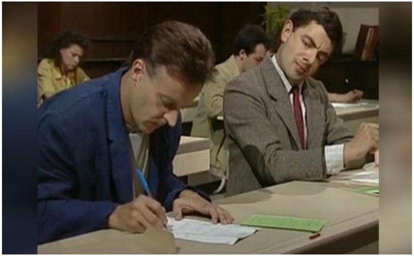
- **Partial Feedback:** Only receives feedback for the action actually taken, not for all possible actions.
- **Objective:** Balance exploration and exploitation to maximize cumulative rewards.

Oblivious Adversary

- **Definition:** Fixes the sequence of events beforehand.
- **Traits:**
  - Non-responsive.
  - Simpler to analyze.

Adaptive Adversary

- **Definition:** Adjusts based on algorithm's past actions.
- **Traits:**
  - Responsive and dynamic.
  - Harder to counter.



Introducing Follow the Perturbed Leader (FPL)

- **Overview:**
  - FPL is a randomized online algorithm that minimizes regret in adversarial settings by adding noise to each expert's cumulative loss and selecting the leader with the lowest perturbed loss.
- **Algorithm Steps:**
  1. **Initialization:** Set cumulative loss $L_i(0) = 0$ for all experts i.
  2. **For each round** $t = 1, 2, \ldots, T$:
     1. **Perturbation:** Draw a random perturbation γ_i for each expert i from a specified distribution (e.g., exponential).
     2. **Leader Selection:** Choose the expert i^* with the minimum $L_i(t-1) + \gamma_i$.
     3. **Decision:** Follow the prediction of expert i^*.
     4. **Update:** Update the cumulative loss $L_i(t) = L_i(t-1) + \ell_i(t)$ for each expert.

Why Perturbation?

Table: Daily Weather Predictions, Outcomes, and Probability of Mistake using Random Weighted Majority

Day	Father	Mother	P of Mistake	Actual Weather
1	Rain	No Rain	$\frac{1}{2}$	Rain
2	Rain	No Rain	$\frac{1}{2}$	No Rain
3	Rain	No Rain	$\frac{1}{2}$	Rain
4	Rain	No Rain	$\frac{1}{2}$	No Rain
5	Rain	No Rain	$\frac{1}{2}$	Rain

Regret Bound for GEX

Lemma: Let $S \subseteq \mathbb{R}^n$ be a set of (not necessarily positive) decisions, and $k^t = [\mathbf{c}^1, \ldots, \mathbf{c}^T]$ a set of cost vectors on those decisions, such that $|\mathbf{c}^t \cdot \mathbf{x}| \leq R$ for all $\mathbf{x} \in S$ and $\mathbf{c}^t \in k^t$. Then, there is an algorithm $\mathcal{A}(\varepsilon)$ that achieves

$$E[\mathrm{loss}(\mathcal{A}(\varepsilon), k^t)] \leq \mathrm{OPT}(k^t) + \varepsilon(4n+2)R^2 T + \frac{4n}{\varepsilon}$$

Use your Maximum Power

In general, oblivious adversaries are easier to handle than adaptive adversaries. However, we can convert an oblivious adversary to an adaptive by being pessimistic.

Figure: An evil adversary in machine learning using its maximum power

Regret Bound for Any Adversary

Lemma: Fix T, let H^* be the set of decision histories of length 0 to $T-1$, and let K^* be the set of all cost histories of length 0 to $T-1$. Then, fix a decision algorithm $\mathcal{A} : K^* \to \Delta(S)$, where $\Delta(S)$ is the set of probability distributions on the set S of possible decisions. Define

$$R(\mathcal{A}, \mathcal{V}) = \mathbb{E}_{\mathcal{A},\mathcal{V}}\left[\sum_{t=1}^{T} \mathbf{c}^t \cdot \mathbf{x}^t - \min_{\mathbf{x}\in S} \sum_{t=1}^{T} \mathbf{c}^t \cdot \mathbf{x}\right].$$

Let $\mathcal{V}$ be an arbitrary adversary. Then, there exists an oblivious adversary $\mathcal{V}'$ such that

$$R(\mathcal{A}, \mathcal{V}') \geq R(\mathcal{A}, \mathcal{V}).$$

Exploration and Exploitation

Now that we have a way to convert an oblivious adversary to an adaptive one, we can focus on using online optimization algorithms to solve bandit problems.

- **Exploration:** Use the perturbation or other methods to explore different experts and their predictions.
- **Exploitation:** Follow the leader to exploit the best-performing expert.

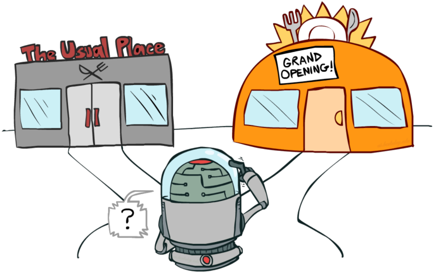


Figure: Exploration vs. Exploitation in Bandit Algorithms

Usage of Basis

- **Definition:** A basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ is a set of linearly independent vectors that span a vector space.
- **Advantages:**
  - Reduces the problem of exploring an infinite set to exploring a finite set.
  - Simplifies the decision-making process.
- **Dimention:** At most n.

Problem Formulation

Table: Summary of notation

$S \subseteq \mathbb{R}^n$	set of decisions, a compact subset of $\mathbb{R}^n$
$D \in \mathbb{R}$	L_1 bound on diameter of S, $\forall \mathbf{x}, \mathbf{y} \in S$, $\lvert \mathbf{x} - \mathbf{y} \rvert_1 \le D$
$n \in \mathbb{N}$	dimension of decision space
$\mathcal{V} : H^* \to \mathbb{R}^n$	adversary, function from decision histories to cost vectors
$\mathcal{A}$	an online optimization algorithm
$\mathbf{c}^t \in \mathbb{R}^n$	cost vector on time t
$\hat{\mathbf{c}}^t \in \mathbb{R}^n$	BGA's estimate of the cost vector on time t
$M \in \mathbb{R}^+$	bound on single-iteration cost, $\lvert \mathbf{c}^t \cdot \mathbf{x}^t \rvert \le M$
$B \subseteq S$	sampling basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$
$\ell^t \in [-M, M]^n$	vector, $\ell_i^t = \mathbf{c}^t \cdot \mathbf{b}_i$ for $\mathbf{b}_i \in B$
$\hat{\ell}^t \in \mathbb{R}^n$	BGA's estimate of ℓ^t
$T \in \mathbb{N}$	end of time, index of final iteration
$\mathbf{x}^t \in S$	BGA's decision on time t
$\tilde{\mathbf{x}}^t \in S$	decision recommended by GEX on time t
$\chi^t \in \{0, 1\}$	indicator, $\chi^t = 1$ if BGA explores on t, 0 otherwise
$\gamma \in [0, 1]$	the probability BGA explores on each timestep
$\tilde{z}^t \in [-R, R]$	loss of GEX, $\tilde{z}^t = \hat{\mathbf{c}}^t \cdot \tilde{\mathbf{x}}^t$

Bandit-style Geometric decision algorithm against an Adaptive adversary

Algorithm BGA

Choose parameters γ and ϵ, **where** ϵ **is a parameter of GEX**
$t = 1$
Fix a basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subseteq S$
while playing **do**
  Let $\chi^t = 1$ **with probability** γ **and** $\chi^t = 0$ **otherwise**
  if $\chi^t = 0$ **then**
    Select $\mathbf{x}^t$ **from the distribution** $\mathrm{GEX}(\hat{\mathbf{c}}^1, \ldots, \hat{\mathbf{c}}^{t-1})$
    Incur cost $z^t = \mathbf{c}^t \cdot \mathbf{x}^t$
    $\hat{\mathbf{c}}^t = 0 \in \mathbb{R}^n$
  else
    Draw j **uniformly at random from** $\{1, \ldots, n\}$
    $\mathbf{x}^t = \mathbf{b}_j$
    Incur cost and observe $z^t = \mathbf{c}^t \cdot \mathbf{x}^t$
    Define $\hat{\ell}^t$ **by** $\hat{\ell}_i^t = 0$ **for** $i \neq j$ **and** $\hat{\ell}_j^t = (n/\gamma) z^t$
    $\hat{\mathbf{c}}^t = (B^\top)^{-1} \hat{\ell}^t$
  end if
  $\hat{\mathbf{c}}^{1:t} = \hat{\mathbf{c}}^{1:t-1} + \hat{\mathbf{c}}^t$
  $t = t + 1$
end while

How BGA Works - Part 1

- **Initialization:**
  - Choose parameters γ and ϵ, where ϵ is a parameter of GEX.
  - Set $t \leftarrow 1$.
  - Fix a basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subseteq S$.

How BGA Works - Part 1

- **Initialization:**
  - Choose parameters γ and ϵ, where ϵ is a parameter of GEX.
  - Set $t \leftarrow 1$.
  - Fix a basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subseteq S$.
- **Main Loop: While** playing **do**
  - Let $\chi^t = 1$ with probability γ and $\chi^t = 0$ otherwise.

How BGA Works - Part 1

- **Initialization:**
  - Choose parameters γ and ϵ, where ϵ is a parameter of GEX.
  - Set $t \leftarrow 1$.
  - Fix a basis $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subseteq S$.
- **Main Loop: While** playing **do**
  - Let $\chi^t = 1$ with probability γ and $\chi^t = 0$ otherwise.
  - **If** $\chi^t = 0$ **then**
    - Select $\mathbf{x}^t$ from the distribution $\mathrm{GEX}(\hat{\mathbf{c}}^1, \ldots, \hat{\mathbf{c}}^{t-1})$.
    - Incur cost $z^t = \mathbf{c}^t \cdot \mathbf{x}^t$.
    - Set $\hat{\mathbf{c}}^t = \mathbf{0} \in \mathbb{R}^n$.

How BGA Works - Part 2

- **Else**
  - Draw j uniformly at random from $\{1, \ldots, n\}$.
  - Set $\mathbf{x}^t = \mathbf{b}_j$.
  - Incur cost and observe $z^t = \mathbf{c}^t \cdot \mathbf{x}^t$.
  - Define $\hat{\ell}^t$ by:
    - $\hat{\ell}_i^t = 0$ for all $i \neq j$.
    - $\hat{\ell}_j^t = \left(\frac{n}{\gamma}\right) z^t$.
  - Set $\hat{\mathbf{c}}^t = (B^\top)^{-1}\hat{\ell}^t$.

How BGA Works - Part 2

- **Else**
  - Draw j uniformly at random from $\{1, \ldots, n\}$.
  - Set $\mathbf{x}^t = \mathbf{b}_j$.
  - Incur cost and observe $z^t = \mathbf{c}^t \cdot \mathbf{x}^t$.
  - Define $\hat{\ell}^t$ by:
    - $\hat{\ell}_i^t = 0$ for all $i \neq j$.
    - $\hat{\ell}_j^t = \left(\frac{n}{\gamma}\right) z^t$.
  - Set $\hat{\mathbf{c}}^t = (B^\top)^{-1} \hat{\ell}^t$.
- Update cumulative costs:
  - $\hat{\mathbf{c}}^{1:t} = \hat{\mathbf{c}}^{1:t-1} + \hat{\mathbf{c}}^t$.
  - Increment time step: $t \leftarrow t + 1$.

Regret Bound for BGA

The regret of the BGA algorithm is bounded by

$$E[\text{loss(BGA)}] \leq (1-\gamma)E[\text{loss(GEX)}] + \gamma MT$$

`Intuition:` BGA explores with probability γ and exploits with probability $1-\gamma$.
Order of Regret: It can be shown that the regret of BGA is of order $O(T^{3/4}\sqrt{lnT})$.

Future Work

1. **Exploring while Exploiting:** I think the exploitation phase has some information that can be used to explore better.
2. **Hyperparameter Tuning:** The parameters γ can be tuned to improve the performance of the algorithm.
3. **Game Theory:** The algorithm can be viewed as an Stackelberg game. Thus there is a possibility of using game theory.
4. **Experiments:** The algorithm hasn't tested on real-world data. It would be interesting to see how it performs in practice.
5. **Bound Tightening:** Investigate whether the current bounds of $O(T^{3/4}\sqrt{\ln T})$ against adaptive adversaries and $O(T^{2/3})$ against oblivious adversaries can be improved to $O(\sqrt{T})$.

References I

Tor Lattimore and Csaba Szepesvári.
Bandit Algorithms.
Cambridge University Press, 2020.

Elad Hazan.
Introduction to Online Convex Optimization.
Cambridge University Press, 2016.

H. B. McMahan and A. Blum.
Online Geometric Optimization in the Bandit Setting Against an Adaptive Adversary.
In J. Shawe-Taylor and Y. Singer, editors, *Learning Theory*, Springer Berlin Heidelberg, 2004, pp. 109–123.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire.
The Nonstochastic Multiarmed Bandit Problem.
SIAM Journal on Computing, 32(1):48–77, 2002.

References II

A. Kalai and S. Vempala.
Efficient algorithms for online decision problems.
Journal of Computer and System Sciences, 71(3):291–307, 2005.