Online Geometric Optimization in the Bandit Setting Against an Adaptive Adversary

H. Brendan McMahan and Avrim Blum

Carnegie Mellon University

December 21, 2024

Introduction

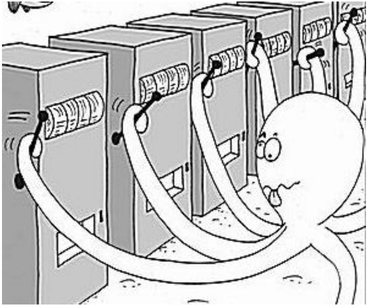
- Overview of online optimization in adversarial environments.
- Importance of studying bandit settings with limited feedback.
- Addressing challenges posed by adaptive adversaries.

Context: Online Optimization with an Adversary

- Online optimization involves making sequential decisions.
- Adversaries can influence the environment by selecting cost vectors.
- Applications in machine learning, economics, and network routing.

Objective: Solving the Bandit Version

- Transition from full-information to bandit feedback.
- Challenges due to limited information after each decision.
- Goal: Develop algorithms that perform well despite these limitations.



Description of the Online Optimization Problem

- Set of feasible points $S \subset \mathbb{R}^n$.
- Sequential decision-making over T rounds.
- At each round t, select $x_t \in S$ and incur cost $c_t \cdot x_t$.

Mathematical Example

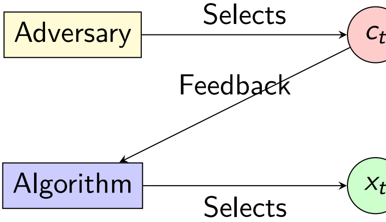
Example: Let S be the unit simplex in $\mathbb{R}^n$, i.e.,

$$S = \left\{ x \in \mathbb{R}^n \mid x_i \geq 0 \text{ for all } i, \sum_{i=1}^{n} x_i = 1 \right\}$$

This setup is common in online portfolio selection.

Adversary and Algorithm Interactions

- Adversary selects cost vectors $c_t \in \mathbb{R}^n$.
- Algorithm selects decisions $x_t \in S$ without knowing c_t in advance.
- Objective is to minimize cumulative cost over T rounds.



Set $S \subset \mathbb{R}^n$

- S represents all possible feasible decisions.
- Geometric properties of S influence algorithm design.
- Examples: Convex sets, polytopes, or combinatorial structures.

Mathematical Example

Convex Set: Let $S = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq 1\}$ be the unit ball in $\mathbb{R}^n$. This convex set allows the use of gradient-based optimization methods due to its smooth boundary.

Cost Incurred: $c_t \cdot x_t$

- Inner product representing the cost for decision x_t.
- c_t is the cost vector chosen by the adversary.
- Objective: Minimize the sum $\sum_{t=1}^{T} c_t \cdot x_t$.

$$\text{Total Cost} = \sum_{t=1}^{T} c_t \cdot x_t$$

The Role of an Oracle for Offline Optimization

- Oracle solves $\min_{x \in S}(c \cdot x)$ efficiently.
- Assumes full knowledge of cost vectors.
- Serves as a benchmark for online algorithms.

Mathematical Example

Offline Optimization: Given all cost vectors $\{c_1, c_2, \ldots, c_T\}$ in advance, the oracle solves:

$$\min_{x \in S} \sum_{t=1}^{T} c_t \cdot x$$

This provides the best possible cumulative cost.

Objective: $\min_{x \in S} (c \cdot x)$

- Offline problem assumes all cost vectors are known in advance.
- Provides the optimal benchmark for comparison.
- Online algorithms aim to perform nearly as well without this knowledge.

$$\text{Optimal Offline Cost} = \min_{x \in S} \sum_{t=1}^{T} c_t \cdot x$$

Observing Only the Cost: $c_t \cdot x_t$

- Limited feedback: Only the incurred cost is observed.
- No access to the full cost vector c_t.
- Necessitates exploration to gather information.

Concrete Example

Example: In online advertising, selecting an ad (decision x_t) and only observing the click-through rate (incurred cost) without knowing the underlying user preferences (cost vector c_t).

Example: Online Shortest Path Problem

- Decision set S consists of all possible paths in a network.
- Cost vectors represent edge weights chosen by the adversary.
- Feedback is the total cost of the chosen path only.

Concrete Example

Consider a network with 4 nodes (A, B, C, D) and 5 edges. At each round, an adversary assigns weights to the edges. The algorithm selects a path from node A to node D and only observes the sum of the weights on the chosen path.

Difference Between Oblivious and Adaptive Adversaries

- **Oblivious Adversary**: Chooses all cost vectors in advance.
- **Adaptive Adversary**: Chooses c_t based on past algorithm decisions.
- Adaptive adversaries are more powerful and challenging.

Adversary Type	**De**
Oblivious	Chooses a
Adaptive	Chooses c_t

Table: Comparison of Adversary Types

Impact of Adversary's Strategy on the Algorithm

- Adaptive strategies can exploit algorithm's weaknesses.
- Necessitates robust algorithms that can handle changing environments.
- Importance of maintaining low regret despite adversary's adaptability.

Formal Definition of Regret

- **Regret** measures the performance gap between the algorithm and the best offline decision.
- Mathematically:

$$\text{Regret} = \mathbb{E}\left[\sum_{t=1}^{T} c_t \cdot x_t\right] - \mathbb{E}\left[\min_{x \in S} \sum_{t=1}^{T} c_t \cdot x\right]$$

  - $\mathbb{E}$: Expectation over the algorithm's randomness.
  - c_t: Cost vector at round t.
  - x_t: Decision made by the algorithm at round t.
- Goal is to achieve sublinear regret: $\text{Regret} = o(T)$.

Goal: Minimize Regret

- Achieve sublinear regret:
  Regret $= o(T)$.
- Ensures average regret per round goes to zero as T increases.
- Fundamental objective in online learning and optimization.

Description of the Kalai-Vempala Algorithm

- General algorithm for online convex optimization.
- Utilizes the exponential weights framework.
- Assumes full feedback: Observes entire cost vector c_t after each decision.

Mathematical Formulation

The Kalai-Vempala algorithm updates the weight $w_t(x)$ for each decision $x \in S$ as:

$$w_{t+1}(x) = w_t(x) \exp\left(-\eta c_t \cdot x\right)$$

where η is the learning rate.

Assumption: Adversary Selects Cost Vectors After Decision

- The algorithm selects x_t without knowing c_t.
- After selection, c_t is revealed.
- Enables the use of gradient-based updates in the algorithm.

$$x_t = \frac{w_t(x)}{\sum_{x' \in S} w_t(x')}$$

Transition to the Bandit Setting

- In the bandit setting, only the incurred cost $c_t \cdot x_t$ is observed.
- No access to the full cost vector c_t.
- Requires estimation techniques to infer c_t from limited feedback.

Estimation Technique

Importance Sampling: To estimate the cost vector, the algorithm can use techniques like importance sampling where:

$$\hat{c}_t = \frac{c_t \cdot x_t}{P(x_t)} e_{x_t}$$

where $P(x_t)$ is the probability of selecting x_t and e_{x_t} is the standard basis vector corresponding to x_t.

Key Challenge: Working with Partial Feedback

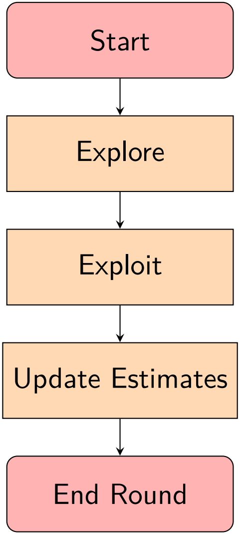
- Limited information makes it difficult to update cost estimates accurately.
- Balancing exploration (gathering information) and exploitation (using current knowledge).
- Ensuring robust performance against adaptive adversaries.
- Incorporating additional mathematical tools for estimation.

Description of the Bandit-style Geometric Decision Algorithm (BGA)

- Designed for bandit feedback in online geometric optimization.
- Combines exploration and exploitation phases.
- Leverages geometric properties of the decision set S.

Alternating Between Exploration and Exploitation

- **Exploitation**: Use current estimates to make informed decisions.
- **Exploration**: Sample randomly from a basis to gather new information.
- Ensures that the algorithm learns the cost structure over time.

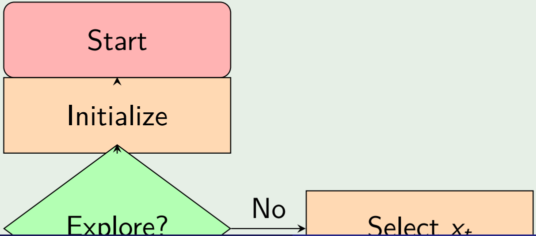


Pseudocode for BGA Algorithm

BGA Algorithm

1. Initialize estimates and sampling basis $B \subset S$.
2. For each round $t = 1$ to T:
   1. With probability γ, **explore** by selecting a random basis vector $b \in B$.
   2. Otherwise, **exploit** by selecting x_t using the Kalai-Vempala strategy on estimated costs.
   3. Observe the incurred cost $c_t \cdot x_t$.
   4. Update cost estimates based on the observed cost.

Flowchart Representation



Explanation of the Sampling Basis $B \subset S$

- B is a carefully chosen subset of S that facilitates exploration.
- Ensures coverage of different regions in the decision space.
- Basis vectors are used to approximate the cost structure.
- Geometric properties of B influence the efficiency of exploration.

Mathematical Insight

If B forms a **basis** in the linear algebra sense, any decision $x \in S$ can be expressed as a linear combination of vectors in B. This allows efficient reconstruction and estimation of the cost vector c_t.

Decision Making Based on Basis B

- During exploration, a basis vector b is selected uniformly at random.
- Provides unbiased estimates of the cost vector c_t.
- Facilitates efficient updating of cost estimates.

$$\hat{c}_t = \frac{c_t \cdot b}{P(b)} e_b$$

where $P(b)$ is the probability of selecting basis vector b, and e_b is the corresponding basis vector.

Exploration Probability γ

- γ controls the frequency of exploration.
- Higher γ leads to more exploration, improving cost estimates.
- Lower γ emphasizes exploitation, leveraging current knowledge.

Balancing γ

The optimal choice of γ depends on problem parameters such as T and n. Typically, γ is set to decrease over time to balance exploration and exploitation effectively.

Trade-off Between Exploration and Exploitation

- Balancing γ is crucial for minimizing regret.
- Too much exploration can waste resources, while too little can lead to poor estimates.
- Optimal γ depends on problem parameters like T and n.

Update Rule for Cost Vector Estimates

- Use observed costs to refine estimates of c_t.
- Employ matrix inversion techniques to handle correlated estimates.
- Ensures accurate approximation of the true cost vectors over time.
- Utilizes geometric properties of the decision set for efficient updates.

$$\hat{c}_{t+1} = \hat{c}_t - \eta \nabla L_t(x_t)$$

where η is the learning rate and $L_t(x_t)$ is the loss at round t.

Introduction to the Mathematical Analysis of BGA

- Formalize the algorithm's update rules and decision-making process.
- Define assumptions and properties of the cost vectors and decision set.
- Set the stage for deriving regret bounds.

Assumptions

1. The cost vectors c_t are bounded, i.e., $\|c_t\| \leq C$ for some constant C.
2. The decision set S is convex and compact.
3. The adversary is adaptive, choosing c_t based on past decisions $x_1, \ldots, x_{t-1}$.

Regret Bounds and Performance Guarantees

- Analyze the cumulative regret over T rounds.
- Provide theoretical guarantees under adaptive adversary models.
- Compare performance with existing algorithms like Kalai-Vempala.

Theoretical Insight

Under the adaptive adversary model, BGA maintains a regret bound of:

$$\text{Regret} = O\left(T^{3/4}\sqrt{\ln T}\right)$$

This ensures that the algorithm performs competitively even in dynamic environments.

Theoretical Regret Result

- BGA achieves regret $O\left(T^{3/4}\sqrt{\ln T}\right)$.
- Sublinear growth ensures that average regret per round diminishes.
- Demonstrates effectiveness in the bandit setting against adaptive adversaries.

Mathematical Proof Sketch

The regret bound is derived using:

1. Concentration inequalities to bound estimation errors.
2. Matrix inversion to handle dependencies in cost estimates.
3. Balancing exploration and exploitation through parameter tuning.

Impact of Parameters γ, ϵ, and T

- γ: Balances exploration and exploitation.
- ϵ: Determines precision of cost estimates.
- T: Number of rounds influences overall regret.
- Optimal tuning of parameters is essential for best performance.

$$\gamma = T^{-1/4}, \quad \epsilon = T^{-1/2}$$

High-Probability Bounds on Cost Vector Estimates

- Establish bounds that hold with high probability.
- Use concentration inequalities to ensure reliable estimates.
- Critical for guaranteeing low regret in adversarial settings.

$$\Pr\left(\|\hat{c}_t - c_t\| \geq \delta\right) \leq \exp(-k\delta^2)$$

where k is a constant depending on T and n.

Using Martingale Inequalities for Estimating c_t

- Apply martingale-based techniques to handle dependencies over time.
- Ensure that estimates remain unbiased and concentrated around true values.
- Facilitates robust analysis against adaptive adversaries.

Theorem (Azuma-Hoeffding Inequality)

Let $\{X_t\}$ be a martingale with bounded differences $|X_t - X_{t-1}| \leq c_t$. Then, for any $\lambda > 0$,

$$\Pr\left(X_T - X_0 \geq \lambda\right) \leq \exp\left(-\frac{\lambda^2}{2\sum_{t=1}^{T} c_t^2}\right)$$

Random Exploration for Estimating True Cost Vectors

- Randomly selecting basis vectors aids in uncovering the cost structure.
- Ensures diverse coverage of the decision space.
- Reduces bias in cost estimates by providing varied perspectives.

$$\mathbb{E}[\hat{c}_t] = c_t$$

Unbiased Estimation

The estimator $\hat{c}_t$ is unbiased because:

$$\mathbb{E}[\hat{c}_t] = \sum_{b \in B} P(b) \cdot \frac{c_t \cdot b}{P(b)} e_b = c_t$$

Importance of Unbiased Estimates

- Unbiased estimates are crucial for accurate decision-making.
- Prevents systematic errors that could be exploited by the adversary.
- Enhances the reliability and performance of the BGA algorithm.

$$\mathbb{E}[\hat{c}_t] = c_t$$

Consequence

Ensures that the algorithm's decisions are based on accurate representations of the cost vectors, leading to effective optimization over time.

Detailed Analysis of Expected Regret for BGA

- Derive bounds on the expected cumulative regret.
- Show how BGA maintains low regret despite adaptive adversaries.
- Compare theoretical performance with empirical observations.

Expected Regret Bound

The expected regret of BGA satisfies:

$$\mathbb{E}[\text{Regret}] \leq O\left(T^{3/4}\sqrt{\ln T}\right)$$

- The bound holds under the adaptive adversary model.
- Demonstrates sublinear growth, ensuring diminishing average regret.

Summary of Performance Guarantees

- BGA achieves sublinear regret in the bandit setting.
- Provides robustness against adaptive adversaries.
- Maintains competitive performance compared to full-information algorithms.

Key Takeaways

- Sublinear regret ensures that the algorithm becomes more effective over time.
- Robustness against adaptive adversaries makes BGA suitable for dynamic environments.
- The bandit setting presents unique challenges that BGA effectively addresses.

Comparison with Kalai-Vempala Algorithm

- Kalai-Vempala assumes full feedback, achieving different regret bounds.
- BGA extends these ideas to the bandit setting with limited feedback.
- Demonstrates comparable performance with additional challenges.

Algorithm	Feedback	Regret Bound
Kalai-Vempala	Full Information	$O(\sqrt{T})$
BGA	Bandit	$O(T^{3/4}\sqrt{\ln T})$

Table: Comparison of Regret Bounds

Recap of the Main Contribution

- Introduction of BGA for online geometric optimization in bandit settings.
- Achieves low-regret performance against adaptive adversaries.
- Bridges the gap between full-information and bandit feedback models.

Implications

BGA provides a robust framework for decision-making in environments where feedback is limited and adversaries can adapt, making it applicable to various real-world scenarios.

Theoretical Bounds for Performance Against an Adaptive Adversary

- Established $O(T^{3/4}\sqrt{\ln T})$ regret bound.
- Guarantees hold under general adversarial conditions.
- Highlights the algorithm's robustness and efficiency.

Future Directions

Further research can explore tightening these bounds and extending BGA to other optimization settings.

Discussion of Open Questions and Potential Future Work

- Can regret bounds be further improved for adaptive adversaries?
- Exploration of different sampling strategies for basis selection.
- Extensions to other types of online optimization problems.
- Incorporating additional feedback mechanisms to enhance performance.

Improving Regret Bounds for Adaptive Adversaries

- Investigate tighter analyses and alternative algorithms.
- Explore the impact of different geometric properties of S.
- Consider hybrid models combining full and bandit feedback.
- Study the interplay between exploration rates and adversary adaptability.

Potential Approaches

1. Utilize advanced concentration inequalities.
2. Incorporate adaptive learning rates.
3. Leverage multi-armed bandit techniques.

Key References

- Kalai, E., & Vempala, S. (2005). A polynomial time algorithm for online convex programming. *Proceedings of the ACM Conference on Learning Theory*, 161-170.
- McMahan, H. B., & Blum, A. (Year). *Title of the Paper*. *Journal/Conference Name*.
- Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3), 235-256.

ithms.

Acknowledgments

- Thank collaborators and contributors.
- Acknowledge funding sources and institutional support.
- Express gratitude to reviewers and advisors.

Example

We would like to thank our colleagues at Carnegie Mellon University for their invaluable feedback and support. This work was supported by NSF Grant #XXXXXX.