Online Geometric Optimization in the Bandit Setting Against an Adaptive Adversary

Originally by H. Brendan McMahan and Avrim Blum of Carnegie Mellon University

Presented by Daniel Obenshain for CS 101.2 at Caltech

Overview

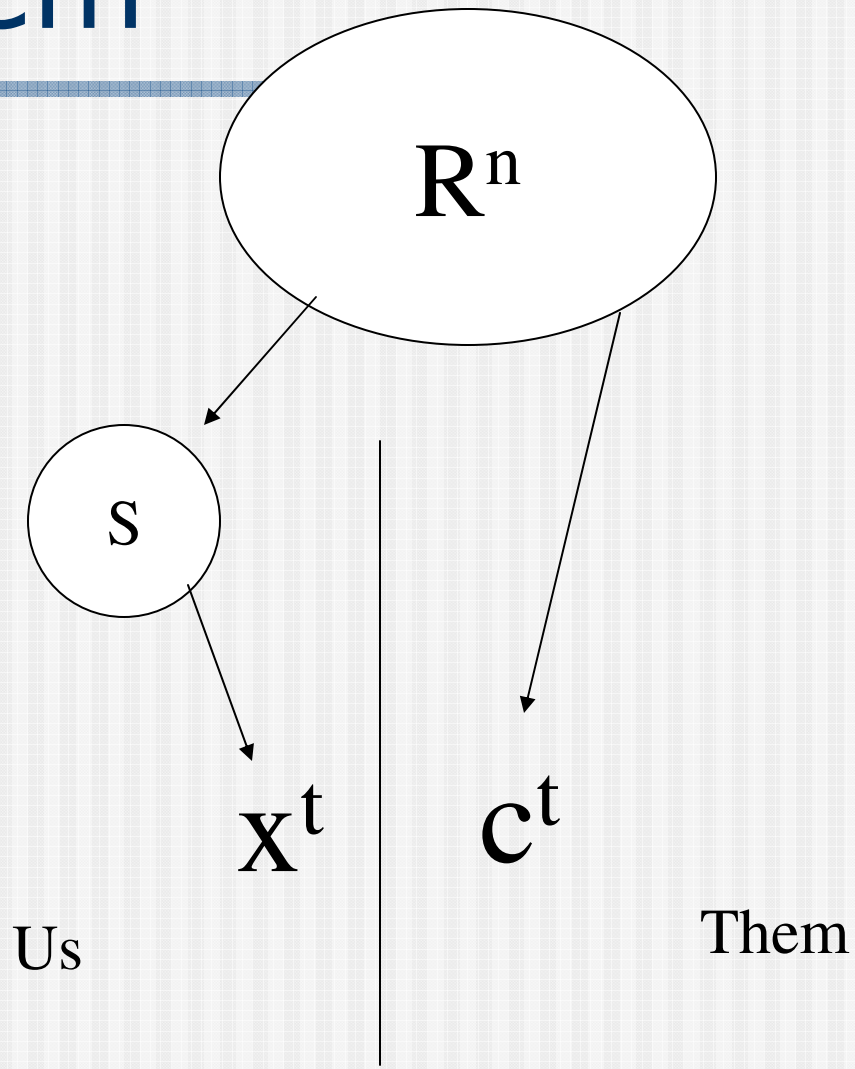
- This is a solution to the Bandit version of the previous problem (Kalai and Vempala's online decision problems).

Overview

- "Bandit Version" of problem
- Algorithm to solve Bandit Version
- Algorithm analysis
- Improvement?

Overall Problem

- S is bounded
- Cost vectors are bounded
- We choose x vector
- Opponent chooses c vector



Notation: superscripts denote iterations of the algorithm

Bandit Version

- What does this mean?
- At each step, we observe the total cost ($x^t \cdot c^t$), not the cost vector (c^t).
- The standard bandit setting can be found by specifying S appropriately.

Intuition

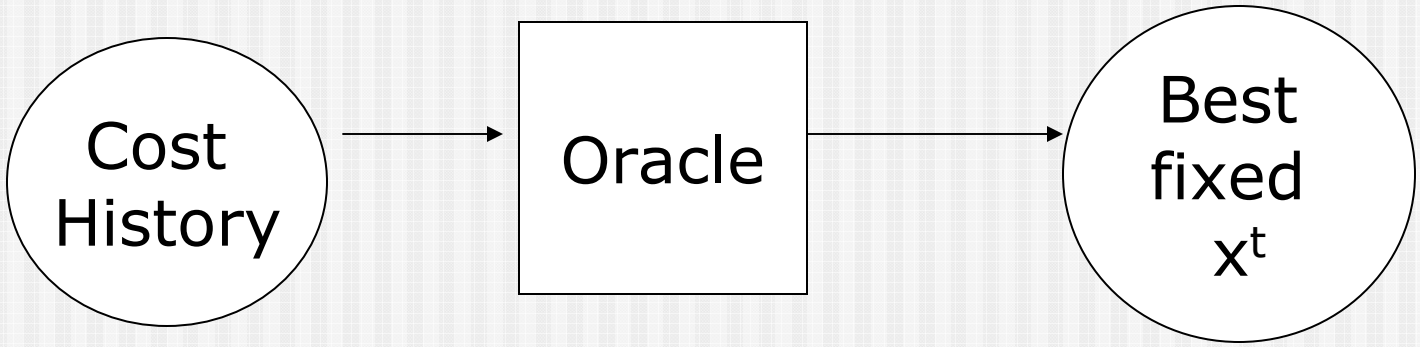
- Even a very smart opponent can't do anything worse to us than what he's already been doing (in the long run).
- So, if we choose x based on which static x would work the best for us in the past, it should work well for us for the next iteration as well.

Algorithm

- Need to minimize regret in the long term.
- Regret is the difference between our choice and the best choice.

Algorithm

- Begin by forming an "oracle"



For our purposes, we will use GEX (Geometric Experts Algorithm), based on the FPL algorithm from Kalai and Vempala.

Oracle, cont.

- In our case, the oracle takes in the cost history:
  - $\hat{c}^1$, $\hat{c}^2$, …, $\hat{c}^{t-1}$
- And outputs a distribution:
  - (choice 1:1%), (choice 2:3%), (choice 3:2%) … over all (infinite) choices in S
- Remember the intuition!

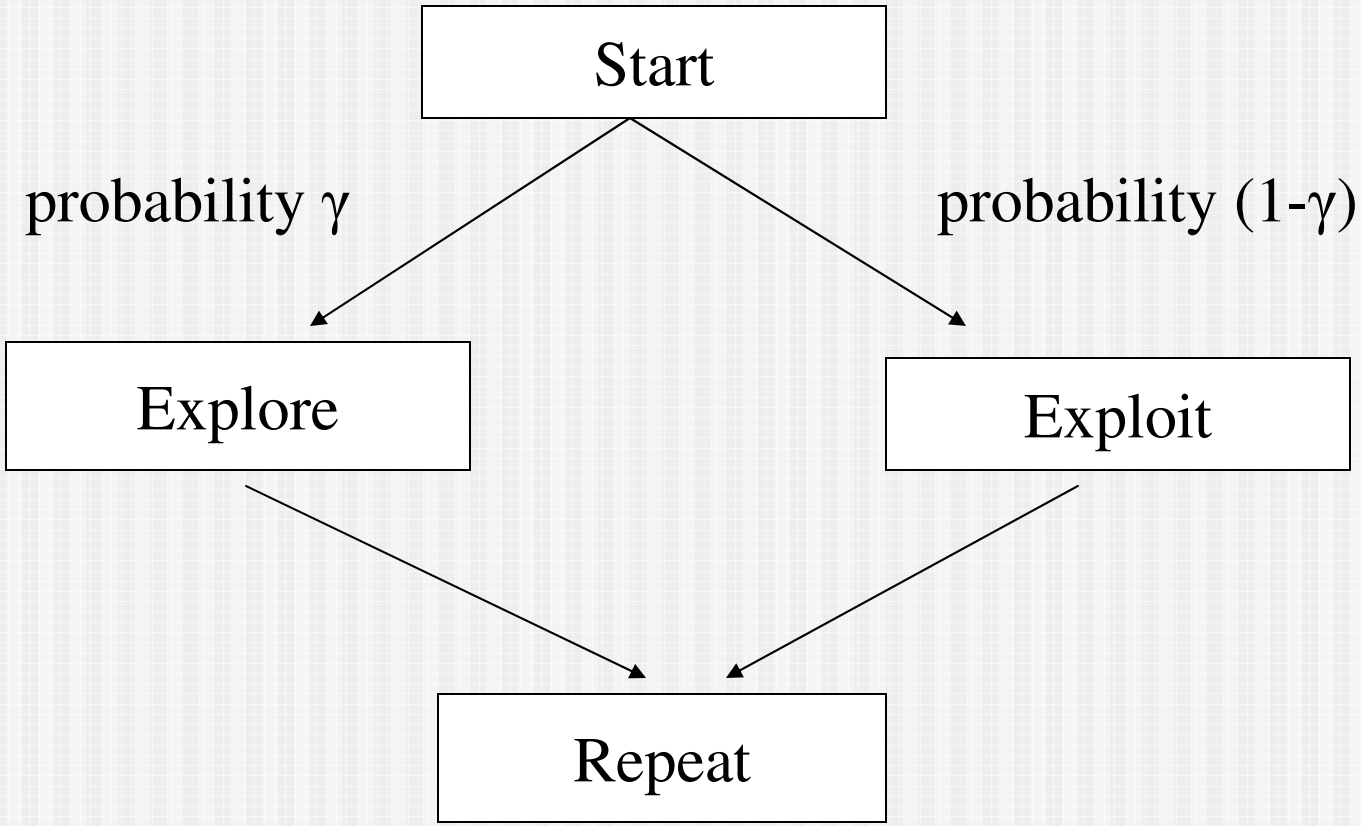
Oracle, cont.

- This distribution represents the best choice for x, considering all previous cost vectors.
- In other words, if we had to choose one x for all previous cost vectors, this would be it.
- It’s probabilistic.

Basis

- We also need a basis to span S.
- We can use an n by n matrix B.
- Each column in B is an element of S.

Algorithm


Start
probability γ
probability (1-γ)
Explore
Exploit
Repeat

Explore

- Randomly choose one of the basis vectors in B.
- Incur cost
- Store information to cost history.

Exploit

- Select x^t from the most current distribution given by the oracle GEX
- Incur cost $x^t \cdot c^t$
- $\hat{c}^t = (0,0,\ldots 0)$ (n-tuple) : Store a "zero" in the cost history to avoid biasing the history.
- Some information is lost here.

Return

- After each iteration, we need to update some data.
- The cost history gains either a zero (for Exploit) or a cost vector (for Explore).

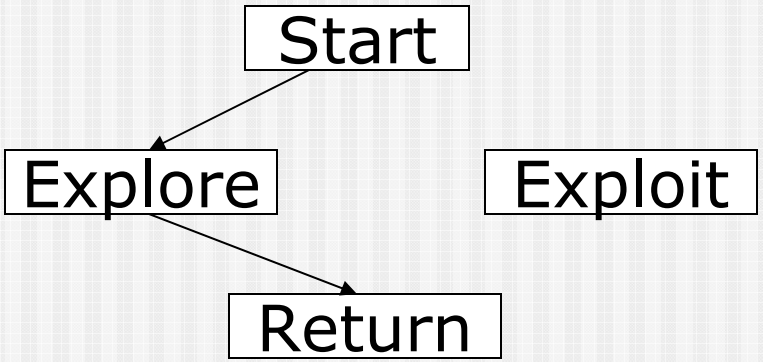
Example

- Consider n = 2 and
- Cost vector sum <= 2 (for bounding constraint) and
- S is [(-1,-1),(1,1)].

- Possible basis is (0,1) and (1,0).

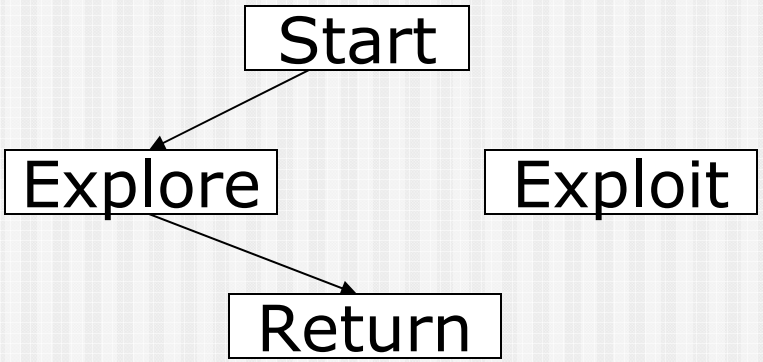
Example

- First iteration:
  - Biased coin flip chooses Explore:
  - Random selection chooses (0,1):
  - Opponent chooses (1,1) for costs:
  - Algorithm sees cost of 1.



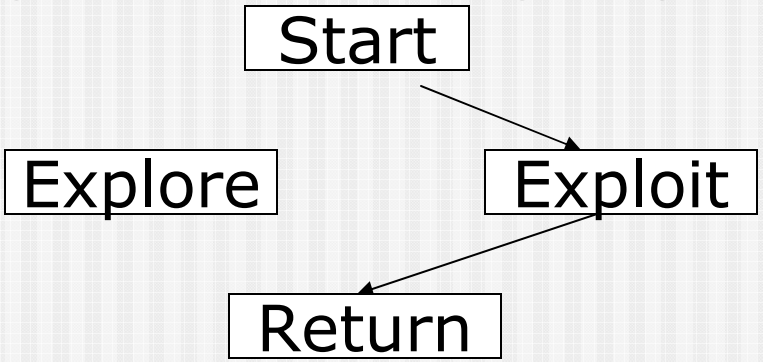
Example

- Second iteration:
  - Biased coin flip chooses Explore:
  - Random selection chooses (1,0):
  - Opponent chooses (1,1) for costs:
  - Algorithm sees cost of 1.



Example

- Third iteration:
  - Biased coin flip chooses Exploit:
  - Oracle returns random value from distribution (.3, .6)
  - Opponent chooses (1,1) for costs:
  - Algorithm sees cost of .9, stores 0



Example

- Continuing. . .
- In the long run:
  - Opponent chooses costs > 0, we choose x < 0.
  - Opponent chooses costs < 0, we choose x > 0.
- Opponent should choose cost of (0,0), which then minimizes our regret.

Algorithm Analysis

- Proof of $O(T^{3/4}\sqrt{(\ln T)})$ regret
- First, estimates.
- Our error in our estimates of the cost times a specific basis vector will decrease as we gain more information through exploration.

Inequalities

- E[loss(GEX)] <= E[optimal performance based on estimated c values] + terms
- This comes from analysis in another paper.

Inequalities

- E[loss(algorithm)] <= E[loss(GEX)]+terms
- This is because the algorithm is an improvement on the oracle itself, with the choice between exploring and exploiting.

Inequalities

- E[optimal performance based on estimated c values] <= E[optimal performance] + terms
- This actually only holds probabilistically, but it’s good enough for the proof.

Proof

- If we combine these inequalities and cleverly set γ equal to T^(-1/4), we get:
- E[regret(algorithm)]= $O(T^{3/4}\sqrt{(\ln T)})$

- Note that γ, instead of changing with time, is simply coupled to T.

The Point

- As we gain more information by exploring,
- we get better estimates of what the adaptive opponent will do
- which decreases our expected regret,
- which ends up being bounded.

Improvement

- This algorithm has $O(T^{3/4}\sqrt{(\ln T)})$
- For an oblivious adversary, $O(T^{2/3})$ is possible.
- For flat bandit problems, $O(\sqrt{T})$ is possible.
- Can this algorithm be improved? Maybe.
- Possibly the information from the exploit step can be better used.

Questions?