Lecture 16: Solving LPs and SDPs with multiplicative weights

Recall: online learning.

For $t = 1, \dots, T$

1. You choose distribution over actions $p_t(i) : [n] \to \mathbb{R}_{\geq 0}$.
2. Adversary chooses loss function $\ell_t(i) : [n] \to [-1, 1]$.
3. Pay expected loss

$$\mathbb{E}_{i \sim p_t}[\ell_t(i)] = \sum_{i=1}^{n} \ell_t(i) p_t(i) = \langle \ell_t, p_t \rangle$$

Multiplicative weights update:

achieves regret

$$\sum_{t=1}^{T} \mathbb{E}_{i \sim p_t}[\ell_t(i)] - \min_{i \in [n]} \sum_{t=1}^{T} \ell_t(i) \leq O(\sqrt{T \log N}).$$

Recall: linear programming

free variables $x_1, \dots, x_n$.

linear constraints $\langle a_i, x \rangle \geq b_i$, $i = 1, \dots, m$

linear objective: $\max \langle c, x \rangle$ subject to all constraints.

This lecture: How to solve linear programs with MW.

Idea: we need to design adversary so that

small regret $\Longleftrightarrow$ good solution to LP.

Step 1: Reduce optimization to feasibility

$$\begin{array}{ll} \min & \langle c, x \rangle \pm \varepsilon \\ \text{s.t.} & Ax \geq b - \varepsilon \end{array} = \min \lambda \pm \varepsilon \text{ s.t. } \begin{array}{l} \langle c, x \rangle \leq b \\ A - x \geq b - \varepsilon \end{array} \text{ is feasible. (i.e. non-empty)}$$

So instead of optimization, we can just ask:

is there $x \in \mathbb{R}^n$ satisfying $Ax \geq b - \varepsilon$?

← for "nice" instances, this also works for finding approximate solutions

Step 2: Restrict the width of the LP.

Assume: we can identify (convex) region K s.t.

width = $\rho := \max\{1, \max_{i \in [m]} |\langle a_i, x \rangle - b|\}$ is bounded.

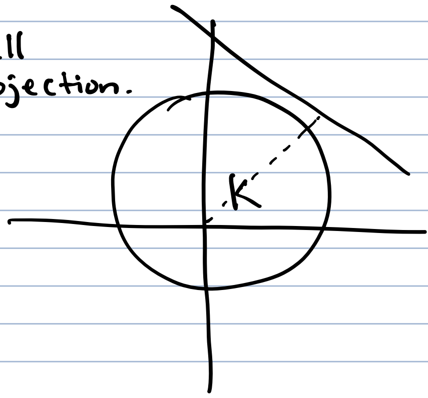
(e.g. large containing ball).

Also, assume that it's easy to solve LP with 1 constraint:

i.e. for any $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}$,

find a point $x \in K$ satisfying $\langle \alpha, x \rangle \geq \beta$, or say no point exists

e.g. If $K = \ell_2$-ball
this is norm of projection.



For all $\varepsilon > 0$,

Thm: Given A and K, there is an algorithm which finds a point $x \in K$ s.t. $\langle a_i, x \rangle \geq b_i - \varepsilon$ $\forall i = 1, \dots, m$, or verifies that there is no point s.t. $Ax \geq b$ and $x \in K$. The algorithm runs in time which is polynomial in $n, m, \frac{1}{\varepsilon}, \rho$.

Idea: Each constraint is an expert

A violated constraint $\rightarrow$ negative loss
A satisfied constraint $\rightarrow$ positive loss

why is this backwards?

(Adversary's)
Algorithm: Let T be fixed later.

For $t = 1, \dots, T$

1. See the player's distribution over constraints

$$p_t(i) : [m] \to \mathbb{R}_{\geq 0}.$$

2. Form the "expected" constraint

$$\sum_{i=1}^{m} p_t(i) \langle a_i, x \rangle \geq \underbrace{\sum_{i=1}^{m} p_t(i) b_i}_{= \tilde{b}_t}$$

$$= \langle \underbrace{\sum_{i=1}^{m} p_t(i) a_i}_{= \tilde{a}_t}, x \rangle$$

3. Solve the 1-constraint LP
$\langle \tilde{a}_t, x \rangle \geq \tilde{b}_t$ s.t. $x \in K$.

4. If no feasible solution, terminate and declare infeasible.

Otherwise, let $\tilde{x}_t$ be s.t.

$$\langle \tilde{a}_t, \tilde{x}_t \rangle \geq b_t \quad , \quad \tilde{x}_t \in K.$$

5. Set our cost vector to be

$$l_t(i) = \frac{(\langle a_i, \tilde{x}_t \rangle - b_i)}{\rho}.$$

if constraint is very satisfied, we downweight it

if it is not satisfied, we upweight it

Output

$$\frac{1}{T}\sum_{t=1}^{T} \tilde{x}_t.$$

<u>Claim 1</u>: If algo terminates early, then LP was infeasible.

pf: We'll prove the contrapositive. Suppose there exists $x \in K$ s.t. $Ax \geq b$. Then

$$\sum p_t(i) \underbrace{\langle a_i, x \rangle}_{\geq b_i \text{ if } Ax \geq b} \geq \sum p_t b_i = \tilde{b}, \text{ so}$$

any 1-constraint LP we encounter will be feasible.

<u>Claim 2</u>: If the algo doesn't terminate early, then output satisfies

$x \in K$ and $Ax \geq b - \varepsilon$, as long as $T = O(\qquad)$

pf: Let's plug in the regret guarantee. Notice that $|l_t(i)| \leq 1$ by def of ρ. So this is ok to do.

Regret guarantee:

$$\sum_{t=1}^{T} \mathbb{E}_{i \sim p_t}[l_t(i)] - \min_{i \in [m]} \sum_{t=1}^{T} l_t(i) \leq \sqrt{T \log m}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{m} p_t(i) \frac{(\langle a_i, \tilde{x}_t \rangle - b_i)}{\rho} \qquad \left(\sum_{t=1}^{T} \frac{\langle a_i, \tilde{x}_t \rangle - b_i}{\rho}\right)$$

$$\Rightarrow \underbrace{\sum_{t=1}^{T} \sum_{i=1}^{m} p_t(i)(\langle a_i, \tilde{x}_t \rangle - b_i)}_{= \langle \tilde{a}_t, \tilde{x}_t \rangle - \tilde{b}_t \geq 0} - \rho\sqrt{T \log m} \leq \min_{i \in [m]} \sum_{t=1}^{T} (\langle a_i, \tilde{x}_t \rangle - b_i)$$

$$\Rightarrow \min_{i \in [m]} \frac{1}{T}\sum_{t=1}^{T} \langle a_i, \tilde{x}_t \rangle - b_i \geq -\rho\sqrt{\frac{\log m}{T}} \qquad \leq \varepsilon \text{ if } T = \frac{\rho^2 \log m}{\varepsilon^2}$$

$$\Rightarrow \forall i, \quad \langle a_i, \frac{1}{T}\sum_{t=1}^{T} \tilde{x}_t \rangle \geq b_i - \varepsilon$$

our final output!

Remarks:

- In general, ρ will be polynomially large $\rightarrow$ poly-time approximation algorithm.

  In some settings, ρ is naturally $O(1)$ $\rightarrow$ nearly linear time!

  e.g. some packing LPs.

- Dependence on ε is unfortunate: ideally would be $\log(1/\varepsilon)$, ours is poly$(1/\varepsilon)$.

What did this method actually need?

1. Reduce to feasibility
2. A decent width bound
3. The ability to solve 1 constraint version of problem.

example: solving SDPs: $X \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{n\times n}$, $A_i \in \mathbb{R}^{n\times n}$

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \langle A_i, X \rangle \geq b \\ & X \succeq 0 \end{aligned}$$

1 + 2 follow for essentially same reason as LPs.

3 becomes: does there exist $X \in K$ s.t.

$$\langle A, X \rangle \geq b$$
$$X \succeq 0. \quad ?$$

Not as immediate but also solvable.

<u>Thm:</u> MW gives an algorithm for solving SDPs in time poly$(n, m, \rho, \frac{1}{\varepsilon})$.

Somewhat better: <u>matrix multiplicative weights</u>