15
Solving Linear Programs using Experts

We can now use the low-regret algorithms for the experts problem to show how to approximately solve linear programs (LPs). As a warm-up, we use it to solve two-player zero-sum games, which are a special case of LPs.

In fact, zero-sum games are equivalent to linear programming, see this work of Ilan Adler. Is there an earlier reference?

15.1 (Two-Player) Zero-Sum Games

There are two players in such a game, traditionally called the "row player" and the "column player". Each of them has some set of actions: the row player with m actions (associated with the set $[m]$), and the column player with the n actions in $[n]$. Finally, we have a payoff matrix $M \in \mathbb{R}^{m \times n}$. In a play of the game, the row player chooses a row $i \in [m]$, and simultaneously, the column player chooses a column $j \in [n]$. If this happens, the row player gets $M_{i,j}$, and the column player loses $M_{i,j}$. The winnings of the two players sum to zero, and so we imagine that the payoff is *from* the row player *to* the column player.

Henceforth, when we talk about payoffs, these will always refer to payoffs to the row player from the column player. This payoff may be negative, which would capture situations where the column player does better.

15.1.1 Strategies, and Best-Response

Each player is allowed to have a randomized strategy. Given strategies $p \in \Delta_m$ for the row player, and $q \in \Delta_n$ for the column player, the expected payoff (to the row player) is

$$\mathbb{E}[\text{payoff to row}] = p^\intercal M q = \sum_{i,j} p_i q_j M_{i,j}.$$

The row player wants to maximize this value, while the column player wants to minimize it.

Suppose the row player fixes a strategy $p \in \Delta_m$. Knowing p, the column player can choose an action to minimize the expected payoff:

$$C(p) := \min_{q \in \Delta_n} p^\intercal M q = \min_{j \in [n]} p^\intercal M e_j.$$

The equality holds because the expected payoff is linear, and hence the column player's best strategy is to choose a column that minimizes the expected payoff. The column player is said to be playing their *best response*. Analogously, if the column player fixes a strategy $q \in \Delta_n$, the row player can maximize the expected payoff by playing their own best response:

$$R(q) := \max_{p \in \Delta_m} p^\intercal M q = \max_{i \in [m]} e_i^\intercal M q.$$

Now, the row player would love to play the strategy p such that even if the column player plays best-response, the payoff is as large as possible: i.e., it wants to achieve

$$\max_{p \in \Delta_m} C(p).$$

Similarly, the column player wants to choose q to minimize the payoff against a best-response row player, i.e., to achieve

$$\min_{q \in \Delta_n} R(q).$$

Lemma 15.1. *For any $p \in \Delta_m, q \in \Delta_n$, we have*

$$C(p) \leq R(q) \tag{15.1}$$

Proof. Intuitively, since the column player commits to a strategy q, it hence gives more power to the row player. Formally, the row player could always play strategy p in response to q, and hence could always get value $C(p)$. But $R(q)$ is the best response, which could be even higher. □

Interestingly, there always exist strategies $p \in \Delta_m, q \in \Delta_n$ which achieve equality. This is formalized by the following theorem:

Theorem 15.2 (Von Neumann's Minimax Theorem)**.** *For any finite zero-sum game $M \in \mathbb{R}^{m \times n}$,*

$$\max_{p \in \Delta_m} C(p) = \min_{q \in \Delta_n} R(q).$$

This common value V is called the value of the game M.

Proof. We assume for the sake of contradiction that $\exists M \in [-1, 1]^{m \times n}$ such that $\max_{p \in \Delta_m} C(p) \leq \min_{q \in \Delta_n} R(q) - \delta$ for some $\delta > 0$. (The assumption that $M_{ij} \in [-1, 1]$ follows by scaling.) Now we use the fact that the average regret of the Hedge algorithm tends to zero to construct strategies $\widehat{p}$ and $\widehat{q}$ that have $R(\widehat{q}) - C(\widehat{p}) < \delta$, thereby giving us a contradiction.

We consider an instance of the experts problem, with m experts, one for each row of M. At each time step t, the row player produces

$p^t \in \Delta_m$. Initially $p^1 = \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$, which represents that the row player chooses each row with equal probability, when they have no information to work with.

At each time t, the column player plays the best-response to p^t, i.e.,

$$j_t := \arg\max_{j \in [n]} (p^t)^\intercal M e_j.$$

This defines a gain vector for the row player:

$$g_t := M e_{j_t},$$

which is the j^{th} column of M. The row player uses this to update the weights and get p_{t+1}, etc. Define

$$\widehat{p} := \frac{1}{T}\sum_{t=1}^{T} p^t \quad \text{and} \quad \widehat{q} := \frac{1}{T}\sum_{t=1}^{T} e_{j_t}$$

to be the average long-term plays of the row player, and of the best responses of the column player to those plays. We know that

$$C(\widehat{p}) \le R(\widehat{q})$$

by (15.1). But by Corollary 14.10, after $T \ge \frac{4\ln m}{\varepsilon^2}$ steps,

$$\begin{aligned}
\frac{1}{T}\sum_t \langle p^t, g^t\rangle &\ge \max_i \frac{1}{T}\sum_t \langle e_i, g^t\rangle - \varepsilon && \text{(by Hedge)}\\
&= \max_i \left\langle e_i, \frac{1}{T}\sum_t g^t \right\rangle - \varepsilon \\
&= \max_i \left\langle e_i, M\left(\frac{1}{T}\sum_t e_{j_t}\right)\right\rangle - \varepsilon && \text{(by definition of } g_t\text{)}\\
&= \max_i \langle e_i, M\widehat{q}\rangle - \varepsilon \\
&= R(\widehat{q}) - \varepsilon.
\end{aligned}$$

Since p^t is the row player's strategy, and C is concave (i.e., the payoff on the average strategy $\widehat{p}$ is no more than the average of the payoffs:

$$\frac{1}{T}\sum \langle p^t, g^t\rangle = \frac{1}{T}\sum C(p^t) \le C\left(\frac{1}{T}\sum p^t\right) = C(\widehat{p}).$$

Putting it all together:

$$R(\widehat{q}) - \varepsilon \le C(\widehat{p}) \le R(\widehat{q}).$$

Now for any $\delta > 0$ we can choose $\varepsilon < \delta$ to get the contradiction. □

To see this, recall that

$$C(p) := \min_q p^\intercal M q.$$

Let q^* be the optimal value of q that minimizes $C(p)$. Then for any $a, b \in \Delta_m$, we have that

$$\begin{aligned}
C(a+b) &= (a+b)^\intercal M q^* = a^\intercal M q^* + b^\intercal M q^* \\
&\ge \min_q a^\intercal M q + \min_q b^\intercal M q = C(a) + C(b)
\end{aligned}$$

Observe that the proof gives us an explicit algorithm to find strategies $\widehat{p}, \widehat{q}$ that have a small gap. The minimax theorem is also implied by strong duality of linear programs: indeed, we can write $\min_{q\in\Delta_n} R(q)$ as a linear program, take its dual and observe that it computes $\min_{p\in\Delta_m} C(p)$. The natural question is: we can solve linear programs using low-regret algorithms. We now show how to do this. We should get a clean proof of strong duality this way?

15.2 Solving LPs Approximately

Consider an LP with constraint matrix $A \in \mathbb{R}^{m \times n}$:

$$\begin{aligned} \max \ & \langle c, x \rangle \\ Ax &\leq b \\ x &\geq 0 \end{aligned} \tag{15.2}$$

Suppose x^* is an optimal solution, with $OPT := \langle c, x^* \rangle$. Let $K \subseteq \mathbb{R}^n$ be the polyhedron defined by the "easy" constraints, i.e.,

$$K := \{x \in \mathbb{R}^n \mid \langle c, x \rangle = OPT, x \geq 0\},$$

where OPT is found by binary search over possible objective values. Binary search over the reals is typically not a good idea, since it may never reach the answer. (E.g., searching for $1/3$ by binary search over $[0, 1]$.) However, we defer this issue for now, and imagine we know the value of OPT. We now use low-regret algorithms to find $\widehat{x} \in K$ such that $\langle a_i, x \rangle \leq b_i + \varepsilon$ for all $i \in [m]$.

The fix to the "binary search over reals" problem is this: the optimal value of a linear program in n dimensions where all numbers integers using at most b bits is a rational p/q, whereboth p, q use at most $\text{poly}(nb)$ bits. So once we the granularity of the search is fine enough, there is a unique rational close the query point, and we can snap to it. See, e.g., the problem on finding negative cycles in the homeworks.

15.2.1 The Oracle

The one assumption we make is that we can solve the feasibility problem obtained by intersecting the "easy" constraints K with a single linear constraint. Suppose $\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}$, then we want to solve the problem:

$$\textsc{Oracle}\text{: find a point } x \in K \cap \{x \mid \langle \alpha, x \rangle \leq \beta\}. \tag{15.3}$$

Proposition 15.3. *There is an $O(n)$-time algorithm to solve (15.3), when $K = \{x \geq 0 \mid \langle c, x \rangle = OPT\}$.*

Proof. We give the proof only for the case where $c_i > 0$ for all i; the general case is left as an exercise. Let $j^* := \arg\min_j \alpha_j / c_j$, and define $x = (OPT/c_{j^*})\mathbf{e}_{j^*}$. Say "infeasible" if x does not satisfy $\langle \alpha, x \rangle \leq \beta$, else return x. □

Of course, this problem can be solved in time linear in the number of variables (as Proposition 15.3 above shows), but the situation can be more interesting when the number of variables is large. For instance, when we solve flow LPs, the number of variables will be exponential in the size of the graph, yet the oracle will be implementable in time $\text{poly}(n)$.

15.2.2 The Algorithm

The key idea to solving general LPs is then similar to that for zero-sum games. We have m experts, one corresponding to each constraint. In each round, we combine the multiple constraints using a

weighted sum, we call the above oracle on this single-constraint LP to get a solution, we construct a gain vector from this solution and feed this to Hedge, which then updates the weights that we use for the next round. The gain of an expert in a round is based based on how badly the constraint was violated by the current solution. The intuition is simple: greater violation means more gain, and hence more weight in the next iteration, which forces us to not violate the constraint as much.

An upper bound on the maximum possible violation is the *width* ρ of the LP, defined by

$$\rho := \max_{x \in K, i \in [m]} \{|\langle a_i, x\rangle - b_i|\}. \tag{15.4}$$

We assume that $\rho \geq 1$.

Algorithm 13: LP-Solver

13.1 $p^1 \leftarrow (1/m, \ldots, 1/m)$. $T \leftarrow \Theta(\rho^2 \ln m/\varepsilon^2)$
13.2 **for** $t = 1$ *to* T **do**
13.3 Define $\alpha^t := \sum_{i=1}^m p_i^t a_i \in \mathbb{R}^n$ and $\beta^t = \sum_{i=1}^m p_i^t b_i \in \mathbb{R}$.
13.4 Use ORACLE to find $x \in K \cap \{\langle \alpha^t, x^t\rangle \leq \beta^t\}$.
13.5 **if** *oracle says infeasible* **then**
13.6 **return** infeasible
13.7 **else**
13.8 $g_i^t \leftarrow \langle a_i, x^t\rangle - b_i$ for all i.
13.9 feed g^t to Hedge(ε) to get p^{t+1}.
13.10 **return** $\widehat{x} \leftarrow (x^1 + \cdots + x^T)/T$.

15.2.3 *The Analysis*

Theorem 15.4. *Fix $0 \leq \varepsilon \leq 1/4$. Then Algorithm 13 calls the oracle $O(\rho^2 \ln m/\varepsilon^2)$ times, and either correctly returns "infeasible", or returns $\widehat{x} \in K$ such that*

$$A\widehat{x} \leq b - \varepsilon \mathbb{1}.$$

Proof. Observe that if x^* is feasible for the original LP (15.2) then it is feasible for any of the calls to the oracle, since it satisfies any positive linear combination of the constraints. Hence, we are correct if we ever return "infeasible". Moreover, $x^t \in K$ in each iteration, and $\widehat{x}$ is an average of x^t's, so it also lies in K by convexity. So it remains to show that $\widehat{x}$ approximately satisfies the other linear constraints.

Recall the guarantee from Corollary 14.10:

$$\frac{1}{T}\sum_t \langle p^t, g^t\rangle \geq \max_i \frac{1}{T}\sum_t \langle e_i, g^t\rangle - \varepsilon, \tag{15.5}$$

for precisely the choice of T in Algorithm 13, since the definition of width in (15.4) ensures that $g^t \in [-\rho, \rho]^m$.

Let $i \in [m]$, and recall the definitions of $a^t = \sum_{i=1}^m p_i^t a_i$, $\beta^t = \sum_{i=1}^m p_i^t b_i$, and $g^t = Ax^t - b$ from the algorithm. Then

$$\begin{aligned}\langle p^t, g^t \rangle &= \langle p^t, Ax^t - b \rangle \\ &= \langle p^t, Ax^t \rangle - \langle p^t, b \rangle \\ &= \langle a^t, x^t \rangle - \beta^t \leq 0,\end{aligned}$$

the last inequality because x^t satisfies the single linear constraint $\langle a^t, x \rangle \leq \beta^t$. Averaging over all times, the left hand side of (15.5) is

$$\frac{1}{T} \sum_{t=1}^T \langle p^t, g^t \rangle \leq 0.$$

However, the average on the RHS in (15.5) for constraint/expert i is:

$$\begin{aligned}\frac{1}{T} \sum_{t=1}^T \langle e_i, g^t \rangle &= \left\langle e_i, \frac{1}{T} \sum_{t=1}^T g^t \right\rangle \\ &= \frac{1}{T} \sum_{t=1}^T \left(\langle a_i, \widehat{x}^t \rangle - b_i \right) \\ &= \langle a_i, \widehat{x} \rangle - b_i.\end{aligned}$$

Substituting into (15.5) we have

$$0 \geq \frac{1}{T} \sum_{t=1}^T \langle p^t, g^t \rangle \geq \max_i \left(\langle a_i, \widehat{x} \rangle - b_i \right) - \varepsilon.$$

This shows that $A\widehat{x} \leq b + \varepsilon \mathbb{1}$. □

15.2.4 *A Small Extension: Approximate Oracles*

Recall the definition of the problem width from (15.4). A few comments:

- In the above analysis, we do not care about the maximum value of $|a_i^\intercal x - b_i|$ over all points $x \in K$, but only about the largest this expression gets over points that are potentially returned by the oracle. This seems a pedantic point, but if there are many solutions to (15.3), we can return one with small width. But we can do more, as the next point outlines.

- We can also relax the oracle to satisfy $\langle a, x \rangle \leq \beta + \delta$ for some small $\delta > 0$ instead. Define the *width* of the LP with respect such a relaxed oracle to be

$$\rho_{\text{rlx}} := \max_{i \in [m], x \text{ returned by relaxed oracle}} \{|a_i^\intercal x - b_i|\}. \tag{15.6}$$

Now running the algorithm with a relaxed oracle gives us a slightly worse guarantee that

$$A\widehat{x} \leq b + (\varepsilon + \delta)\mathbb{1},$$

but now the number of calls to the relaxed oracle can be even smaller, namely $O(\rho_{\text{rlx}}^2 \ln m/\varepsilon^2)$.

- Of course, if we violations can be bounded in some better way, e.g., if we can ensure that violations are always positive or negative, then we can give stronger bounds on the regret, and hence reduce the number of calls even further. Details to come.

All these improvements will be crucial in the upcoming applications.