Design and Analysis of Algorithms
Massachusetts Institute of Technology 6.046J/18.410J
Debayan Gupta, Aleksander Madry, and Bruce Tidor Handout 5

Recitation 6: Game Theory, Learning from Experts

1 Game Theory and Min-Max Theorem

In lecture we introduced the concept of a Nash equilibrium, a configuration where no player can improve his expected payoff by changing only his strategy.

Nash proved that every finite game has a Nash equilibrium. In lecture we proved that every finite *zero-sum* two-player game has a Nash equilibrium and we're going to do it again now.

Suppose there are m strategies for player A and n strategies for player B. Let A's m by n payout matrix be $A[i, j]$. Because the game is zero sum, we can capture all the information in the problem with player A's payout matrix– B's payout is always just the negative of A's payout.

From Player A's perspective, he wants to play the mixed strategy [1] denoted by $[p_i] = [p_1, p_2, ..., p_m]$ (with probability p_i he will play his i^{th} strategy) such that his payout is maximized. But A knows B will choose a (mixed) strategy $[q_j]$ to maximize B's payout, or equivalently to minimize A's payout. That is, presented with A playing some fixed strategy $[p_i]$, B chooses

$$[q_j^*] = \arg\min_{[q_j]} \sum_{i,j} p_i q_j A[i, j]$$

A's choice then boils down to maximizing his payout given the above, and his optimal payout is

$$a^* = \max_{[p_i]} \min_{[q_j]} \sum_{i,j} p_i q_j A[i, j]$$

Now from Player B's perspective, we have just the opposite, because the game is zero-sum. B's optimal payout is

$$b^* = -\min_{[q_j]} \max_{[p_i]} \sum_{i,j} p_i q_j A[i, j]$$

The Min-Max theorem says that these two payouts are exactly opposite, i.e. $a^* = -b^*$. Note that this immediately proves the existence of a Nash equilibrium - the best strategy that B can use against A's optimal strategy achieves the same as B's own optimal strategy (and vice versa). Hence, neither player has an incentive to change strategy once both are playing the strategies that result in (a^*, b^*).

[1] as opposed to a pure strategy

Proof of the Min-Max theorem:

Given that A is playing the strategy $[p_i]$, A's payout when B plays his j^{th} strategy is

$$A[1,j]p_1 + A[2,j]p_2 + ... + A[m,j]p_m$$

We can then represent A's payoff maximization as the following linear program:

$$\begin{aligned} \max \ & a \\ \text{subject to } & A[1,1]p_1 + A[2,1]p_2 + ... + A[m,1]p_m \geq a \\ & A[1,2]p_1 + A[2,2]p_2 + ... + A[m,2]p_m \geq a \\ & ... \\ & A[1,n]p_1 + A[2,n]p_2 + ... + A[m,n]p_m \geq a \\ & \sum_i p_i = 1 \\ & p_i \geq 0 \end{aligned}$$

There is a constraint for each of B's strategies; together they express the idea that no matter which strategy B chooses, A's payout must be at least a.

Similarly, if B is playing the strategy $[q_j]$, B's payout when A plays his i^{th} strategy is

$$-A[i,1]q_1 - A[i,2]q_2 - ... - A[i,n]q_n$$

So B's payoff maximization problem is

$$\begin{aligned} \max \ & b \\ \text{subject to } & -A[1,1]q_1 - A[1,2]q_2 - ... - A[1,n]q_n \geq b \\ & -A[2,1]q_1 - A[2,2]q_2 - ... - A[2,n]q_n \geq b \\ & ... \\ & -A[m,1]q_1 - A[m,2]q_2 - ... - A[m,n]q_n \geq b \\ & \sum_j q_j = 1 \\ & q_j \geq 0 \end{aligned}$$

Or equivalently with the substitution $z = -b$:

$$
\begin{aligned}
\min \ & z \\
\text{subject to } & A[1,1]q_1 + A[1,2]q_2 + ... + A[1,n]q_n \leq z \\
& A[2,1]q_1 + A[2,2]q_2 + ... + A[2,n]q_n \leq z \\
& ... \\
& A[m,1]q_1 + A[m,2]q_2 + ... + A[m,n]q_n \leq z \\
& \sum_j q_j = 1 \\
& q_j \geq 0
\end{aligned}
$$

These linear programs are duals of each other[2], and so by LP duality their optimal values coincide. Thus $a^* = z^* = -b^*$, i.e. A's optimal payout is equal to the negative of B's optimal payout.

2 Learning from Expert Advice

2.1 Review

Recall the problem, as seen in class, of learning from expert advice. In this problem, we have n experts, each of which will give their opinion every day on a "Yes/No" question. Our goal is to develop a way of using the experts' advice to answer the questions minimizing the number of mistakes.

In class, we have considered options to approach this problem that included deterministic and randomized algorithms. In this recitation, we will focus on the randomized methods and prove that they perform well in expectation.

2.2 Randomized Halving

Let us first assume that we are back to the case where the best expert does not do any mistake. We will now try to apply randomization to improve the guarantee given by the Halving Algorithm. The new algorithm that we employ is called the Randomized Halving Algorithm since it is a variation of the deterministic Halving Algorithm from class in which majority selection is replaced by a random choice among the experts that are still in the pool.

More precisely, we consider the following algorithm:

[2]exercise for the reader. You may want to apply the trick to allow z and y to be negative to convert the programs to standard form.

1. At the beginning, each expert i is in the pool of trustworthy experts denoted by S
2. In each round t:
   (a) Choose a random expert from S and follow his/her prediction
   (b) After seeing the answer to the question at the end of t, we remove from S all the experts that made an incorrect prediction in the round

Lemma 1. *The expected number of mistakes $E[m]$ of the Randomized Halving Algorithm is at most $\ln(n)$.*

Proof. Let Ft be the fraction of alive experts (i.e., the ones still in the pool S) that are wrong in round t. This means that at round t, we will make a mistake with probability F_t. Hence, summing over all the rounds, we get that the expected number of mistakes we will do is

$$E[m] = \sum_t F_t \tag{1}$$

Now, it is not hard to see that the fraction of the initial experts that are still alive after t steps is

$$P_t := \prod_t (1 - F_t) \geq \frac{1}{n} \tag{2}$$

where the last inequality holds as the best (perfect) expert will never be removed from the pool.

Taking a natural logarithm of this equation, we get

$$\sum_t \ln(1 - F_t) \geq -\ln(n) \tag{3}$$

We can check with the Taylor expansion of $\ln(n)$ that for $0 < x < 1$ it must hold that $\ln(1 - x) < -x$. Hence,

$$E[m] = \sum_t F_t \leq \ln(n) \tag{4}$$

As we desired to prove. Note that $\ln(n) \approx 0.7 \log_2(n)$. So, the performance of this algorithm is strictly better than the best possible performance of a deterministic solution. Of course, the price of that is that now the mistake guarantee holds only in expectation. Also, remember that this analysis works only under the assumption that there is an infallible expert. We will remove that assumption in the following section.

2.3 Randomized Weighted Majority Algorithm

We proceed now to developing a randomized algorithm for the general case, i.e., a one in which the best expert is making m mistakes. Once again, our algorithm will be a simple modification of its deterministic equivalent. We update the Weighted Majority Algorithm by making it take randomized weighted majority vote (instead of a deterministic one). Furthermore, instead of decreasing the weights of incorrect experts by a factor $1/2$ we consider a more general update by a factor of $(1-\epsilon)$, where ϵ is a tuning parameter of the algorithm. (Clearly, taking $\epsilon = 1/2$ recovers the original update.)

Formally, the Randomized Weighted Majority Algorithm works as follows:

1. At the beginning, each expert i has a weight 1
2. In each round t:
   (a) Choose a random expert (with probability proportional to his/her weight) and follow his/her prediction
   (b) After seeing the answer to the question at the end of t, we decrease the weight w_i by a factor of $(1-\epsilon)$ for all the experts that made an incorrect prediction in the round

Lemma 2. *If $0 < \epsilon \leq 1/2$, and $E[m]$ is the expected number of mistakes of the Randomized Weighted Majority Algorithm then*

$$E[m] \leq (1+\epsilon)m^* + \frac{\ln(n)}{\epsilon} \tag{5}$$

The proof is very similar to the one of Lemma 4. Let F_t be the weighted fraction of experts that are wrong in round t. Let $w_{t,i}$ be the weight of expert i at the end of round t and let $W_t := \sum_i w_{t,i}$. Obviously, $W_0 = n$. Also, as in the previous proof, one has that $E[m] = \sum_t F_t$. Now, note that

$$W_t \leq n(1-\epsilon F_1)...(1-\epsilon F_t) = n\prod_{t' \leq t}(1-\epsilon F_{t'}) \tag{6}$$

Since the best expert i^* makes at most m^* mistakes, it follows that

$$w_{t,i^*} \geq (1-\epsilon)^{m^*} \tag{7}$$

Obviously, $w_{t,i^*} \leq W_t$. So,

$$(1-\epsilon)^{m^*} \leq w_{t,i^*} \leq W_t \leq n\prod_{t' \leq t}(1-\epsilon F_{t'}) \tag{8}$$

Taking the natural logarithm of both sides and using the same Taylor approximation as the previous proof will yield the following:

$$\ln(1-\epsilon)m^* \leq \ln(n) - \epsilon E[m] \tag{9}$$

Hence, again using Taylor expansion, if $0 < \epsilon \leq 1/2$ then

$$E[m] \leq (1+\epsilon)m^* + \frac{\ln(n)}{\epsilon} \tag{10}$$

As we wanted to show. Note that, as ϵ becomes smaller, our multiplicative overhead over m is approaching 1, but the additive term is growing. So, we usually want to choose ϵ in a way that balances out these two effects. Also, as we typically expect that m grows as the total number of turns T becomes larger, one can see that for sufficiently large T the average number of mistakes of this algorithm is close to optimal.

3 Optional material: Kruskal's algorithm revisited

The following section offers an alternative proof of correctness of Kruskal's algorithm. This is an interesting topic that should show a new (and perhaps unintuitive) usage of LP duality as a proof technique. We encourage students to read through this section on their own, and make sure they understand it, but it will not be covered in full in recitation.

Review

Recall the minimum-weight spanning tree problem: given a connected, undirected, weighted graph with positive weights, find the minimum-weight set of edges connecting every pair of vertices in the graph.[3]

The MST problem can be solved in a variety of ways, including with greedy algorithms. In particular, we saw in lecture Kruskal's algorithm:

Line 7 guarantees that what we return, T, will not have cycles. Because we iterate through all edges, we know that T will span the graph. [4] So T is a spanning tree, and we can prove that it is of minimal weight using the exchange argument described in lecture.

[3]The tree property follows from the fact that no minimum-weight spanning subgraph could contain a cycle, because if it did then an edge on that cycle could be removed to produce a subgraph with lower weight (this relies on the fact that the weights are positive).

[4]Suppose T does not span the graph. Because G is connected, there exists a path in G between disconnected components of T. At least one of the edges in that path would have passed the test on line 7 and would been added to T.

KRUSKAL$(G = (V, E, w))$

```
1: T = ∅
2: for v ∈ V do
3:     MAKESET(v)
4: end for
5: Sort all edges in E by weight in increasing order
6: for e = (u, v) in E (in sorted order): do
7:     if FINDSET(u) ≠ FINDSET(v) then
8:         T = T ⋃ e
9:         UNION(u, v)
10:    end if
11: end for
12: return T
```

Now, the exchange argument is simple enough, but because it will be interesting to do so, let's prove that Kruskal's algorithm is correct using LP duality.

Proof of correctness using LP duality

Outline

- Given any valid input graph G (connected, undirected, positive weights)...
- Find a minimization linear program whose optimal objective value is at most the weight of the MST of G.
- Find the dual: a maximization program whose objective value is always at most the weight of the MST of G.
- Show that the spanning tree Kruskal's algorithm produces has a weight that equal to the value of the objective function for some feasible solution to this dual.

Take a moment to convince yourself that this is sufficient to prove optimality of Kruskal's algorithm.

Formulate as an ILP P

First, we can formulate the MST problem as an integer LP. We can represent a tree as a set of edges $T \subseteq E$, and we can represent a set of edges T using $|E|$ indicator variables $\{x_e\}$, each indicating whether edge e is included in T. Now, we must constrain the variables x_e to ensure that they actually represent a spanning tree.

Because any acyclic set of $|V|-1$ edges is a spanning tree [5], it will suffice to enforce the constraints that $|T| = |V| - 1$ and that T has no cycles. Now, because a cycle requires as many edges as vertices, we can enforce the latter constraint by requiring that any $S \subset V$ induces a subgraph containing at most $|S| - 1$ edges (ignoring the empty set). (This is a necessary condition as well.)

Finally, our objective function will be the sum of the weights of the edges in T, and we will want to minimize this.

So we have the program P:

$$\begin{aligned}
\min \quad & \sum_{e \in E} w(e) x_e \\
\text{subject to} \quad & \sum_{e \in E} x_e = |V| - 1 \\
& \sum_{e=(u,v) \in E | u,v \in S} x_e \leq |S| - 1 \qquad \forall S \subset V \\
& 0 \leq x_e \leq 1 \qquad \forall e \in E
\end{aligned}$$

Note that this LP has an exponential number of constraints, because the number of vertex subsets $S \subset V$ is exponential in $|V|$. We wouldn't want to use this to actually find the MST.

Every spanning tree (when represented with the variables x_e) satisfies the constraints of P. Thus the weight of the MST is at least $opt(P)$. [6]

By (weak) LP duality, every solution to the dual of P will have value less than the value of any solution to the primal P, and in particular less than $opt(P)$. So it will suffice to show that there exists a feasible solution to the dual of P with objective value equal to the weight of the tree K that Kruskal's algorithm produces. That is, what we want to show is the first equality in

$$w(K) = \text{a feasible value of the objective function of the dual of P} \leq opt(P) \leq \text{weight of MST}$$

[5] Proof by induction on $|V|$:
Base case: True for $|V| = 2$.
Inductive step: Now consider a graph G of size $|V|$ and an acyclic set T of $|V| - 1$ of the edges of G. There must be (at least) one vertex that exactly 1 edge of T is incident on (otherwise T would either not be spanning or would contain a cycle). Remove this vertex from G to get G' of size $|V| - 1$, and remove the edge in T incident on v from T to get T', an acyclic set of $|V| - 2$ edges. By the inductive hypothesis, T' is a spanning tree of G'. Now add the edge and vertex back and we evidently have that T is a spanning tree of G.

(This should also be intuitively obvious by now– every time we select an edge in Kruskal's algorithm, we connect two trees, and because we start with $|V|$ trees this will be done exactly $|V| - 1$ times. Or, in Prim's algorithm, every time we add an edge we connect a new vertex. Since we start with one vertex and no edges, when we have added $|V| - 1$ edges we will have $|V|$ vertices in our tree.)

[6] Although we are only proving one direction here, this program P is in fact equivalent to the MST problem (otherwise the rest of our proof would not work out).

Convert to P' in standard form

Before finding the dual of P, let's put it in standard form. We can split the equality constraint into two inequality constraints and then absorb one of them into the (originally) second constraint. Also, note that the constraints $x_e \leq 1$ are redundant with the second constraint because $S = \{u, v\} \subset V$ for each edge u, v. Finally, we will also write it as a maximization by negating the objective function. All these changes give the equivalent program P':

$$\begin{aligned}
\max \quad & \sum_{e \in E} -w(e)x_e & \\
\text{subject to} \quad & \sum_{e \in E} -x_e \leq -(|V| - 1) & \\
& \sum_{e=(u,v) \in E | u,v \in S} x_e \leq |S| - 1 & \forall S \subseteq V \\
& x_e \geq 0 & \forall e \in E
\end{aligned}$$

Write dual of P'

Now let's write the dual of P'. It may help to imagine the constraint coefficient matrix. It has $1 + |\{S \subseteq V, S \neq \emptyset\}|$ rows and $|E|$ columns. The first row is all -1's. The remaining $|\{S \subseteq V\}|$ when considered as a separate matrix are such that the element in position i, j is 1 if the edge corresponding to column j is in the set S corresponding to row i and 0 otherwise. So, looking down the columns of this matrix, we can see that for each edge e we'll have a constraint involving all the vertex sets containing e.

$$\begin{aligned}
\min \quad & -z(|V| - 1) + \sum_{S \subseteq V} (|S| - 1)y_S & \\
\text{subject to} \quad & -z + \sum_{S \subseteq V | u,v \in S} y_S \geq -w_e & \forall e = (u, v) \in E \\
& z, y_S \geq 0 & \forall S \subseteq V
\end{aligned}$$

Again, all we need to do now is find a feasible solution to the dual and show that its value is the value of the spanning tree produced by Kruskal's algorithm. Note that we have changed the sign of the objective function in converting to standard form, but this doesn't matter. Every dual solution is effectively a lower bound on the weight of a spanning tree so if we can show that Kruskal's algorithm achieves this, we will have shown that it is optimal.

You may wish to pause for a moment here and try to come up with an intuitive interpretation of the dual.

Construct solution to dual

We need to consider the substructure of Kruskal's algorithm. Suppose running the algorithm adds the edges $K = \{e_1, e_2, ..., e_{|V|-1}\}$ to the growing forest in that order (so $w(e_1) \leq w(e_2) \leq ... \leq w(e_{|V|-1})$).

Consider the sequence of components that are created as we run Kruskal's. Each edge e that we add to our forest creates a single new connected component which we will denote by its vertex set $C(e)$. Now, define the parent of e to be $p(e)$, the first edge added that leaves the component $C(e)$.

These relationships are enough to define the following dual solution

$$\begin{aligned} y_S &= -w(e) + w(p(e)) & S &= C(e), e \in K \setminus e_{|V|-1} \\ y_S &= 0 & &\text{for all other } S \subseteq V \\ z &= w(e_{|V|-1}) \end{aligned}$$

Show solution is feasible

We simply check each constraint in the dual.
The nonnegativity constraint is satisfied because $p(e)$ is certainly added to the forest after e is, so because Kruskal's takes edges in increasing order, $w(e) \leq w(p(e))$.
Now consider the edge constraint corresponding to an arbitrary edge $e = (u, v)$. We need to sum the variables y_S for each vertex set S containing e. The only nonzero y_S are the ones that are $C(e)$ for some edge $e \in K$ as defined above. The first connected component to contain e is $C(e)$ by definition. Now, the next component to contain e is $C(p(e))$, again by definition. And so on. Note that the parent of the edge corresponding to the last component in this sequence is necessarily $e_{|V|-1}$ because this last edge connects the whole graph.

Denote the set of edges containing e and all of its ancestors by $\{e_j\}$. Then our constraint equation evaluates to

$$\begin{aligned} -z + \sum_{S \subseteq V | u,v \in S} y_S &= -z + \sum_{\{e_j\}} y_{C(e_j)} \\ &= -z + y_{C(e)} + y_{C(p(e))} + y_{C(p(p(e)))} + ... \\ &= -z + [-w(e) + w(p(e))] + [-w(p(e)) + w(p(p(e)))] + ... + [-w(e_?) + w(e_{|V|-1})] \\ &= -z + -w(e) + w(e_{|V|-1}) \\ &= -w(e) \end{aligned}$$

(Note that every one of these edge constraints is tight. This is an example of complementary slackness, which says that in general if you consider any optimal solution x_* to a primal in standard form and any optimal solution y_* to the dual, for every primal variable which is positive in x_*, the corresponding dual constraint is tight and vice versa. This can be proved directly from LP duality:

$$\begin{aligned} c^T x_* =& y_*^T A x_* = b^T y_* \\ c^T x_* = y_*^T A x_* \quad &\text{and} \quad y_*^T A x_* = b^T y_* \\ (c - y_*^T A) x_* = 0 \quad &\text{and} \quad y_*^T (b - A x_*) = 0 \end{aligned}$$

These equations and the zero-product property give us complementary slackness.)

Show objective value of this solution is equal to the (negative) weight of the tree Kruskal's finds

Checking the value of the objective function is a bit trickier, but it still works out nicely because of our choice of dual solution. We want to show that

$$\sum_{e \in K} -w(e) = -z(|V| - 1) + \sum_{S \subseteq V} (|S| - 1) y_S$$

Now, let's take advantage of the work we just did to show that the constraints are satisfied. Sum over all the constraints corresponding to edges in K to get:

$$\begin{aligned} \sum_{e \in K} -w(e) &= \sum_{e=(u,v) \in K} [-z + \sum_{S \subseteq V | u,v \in S} y_S] \\ &= \sum_{e \in K} -z + \sum_{e=(u,v) \in K} \sum_{S \subseteq V | u,v \in S} y_S \\ &= -z|K| + \sum_{e=(u,v) \in K} \sum_{S \subseteq V | u,v \in S} y_S \\ &= -z(|V| - 1) + \sum_{e=(u,v) \in K} \sum_{S \subseteq V | u,v \in S} y_S \end{aligned}$$

Comparing this to the objective function, all we need to show now is that

$$\sum_{S \subseteq V} (|S| - 1) y_S = \sum_{e=(u,v) \in K} \sum_{S \subseteq V | u,v \in S} y_S$$

Now, let's switch the order of summation of the right hand side. For a particular y_S, it is counted once in the double summation for each edge in K contained in S. Alternatively, you can think of

the above double sum as being a sum over all pairs $(e = (u, v) \in K, S \subseteq V)$ such that $u, v \in S$, and of course we can instead sum over all pairs $(S \subseteq V, e = (u, v) \in K)$ such that $u, v \in S$.

$$\sum_{S \subseteq V} (|S| - 1) y_S = \sum_{S \subseteq V} \sum_{e=(u,v) \in K | u,v \in S} y_S$$

So it will suffice to show that

$$(|S| - 1) y_S = \sum_{e=(u,v) \in K | u,v \in S} y_S \quad \forall S \subseteq V$$

Now, consider any $S = C(e)$ for $e \in K \setminus e_{|V|-1}$. $C(e)$ was defined as the vertex set of a connected component formed during Kruskal's algorithm. Now, the subset of K consisting of the edges that are contained in $C(e)$ form a spanning tree of $C(e)$ (by definition of $C(e)$). So this subset has size $|C(e)| - 1 = |S| - 1$.
For all other $S \subseteq V$, y_S is zero so the equality holds in those cases too.

Thus we have shown the first inequality in

$-w(K) \leq$ a feasible value of the objective function of the dual of $P' \geq opt(P') \geq -$weight of MWST

which implies (by multiplying through by -1) that $w(K) \leq$ weight of MWST. So we are done. Note that (as we would expect) the first inequality was actually shown to be tight.

Strictly optional reading suggestion: section 16.4 of CLRS (Matroids).