CS 170 HW 10

Due **2021-11-08, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

2 How to Gamble With Little Regret

Suppose that you are gambling at a casino. Every day you play at a slot machine, and your goal is to minimize your losses. We model this as the experts problem. Every day you must take the advice of one of n experts (i.e. play at a slot machine). At the end of each day t, if you take advice from expert i, the advice costs you some c_i^t in $[0, 1]$. You want to minimize the regret R, defined as:

$$R = \frac{1}{T}\left(\sum_{t=1}^{T} c_{i(t)}^t - \min_{1\le i\le n}\sum_{t=1}^{T} c_i^t\right)$$

where $i(t)$ is the expert you choose on day t. Notice that in this definition, you are comparing your losses to the best expert, rather than the best overall strategy.

Your strategy will be probabilities where p_i^t denotes the probability with which you choose expert i on day t. Assume an all powerful adversary (i.e. the casino) can look at your strategy ahead of time and decide the costs associated with each expert on each day. Let C denote the set of costs for all experts and all days. Compute $\max_C(\mathbb{E}[R])$, or the maximum possible (expected) regret that the adversary can guarantee, for each of the following strategies.

(a) Any deterministic strategy, i.e. for each t, there exists some i such that $p_i^t = 1$.

(b) Always choose an expert according to some fixed probability distribution at every time step. That is, fix some $p_1, \dots, p_n$, and for all t, set $p_i^t = p_i$.

What distribution minimizes the regret of this strategy? In other words, what is $\operatorname{argmin}_{p_1,\dots,p_n} \max_C(\mathbb{E}[R])$?

This analysis should conclude that a good strategy for the problem must necessarily be randomized and adaptive.

Solution:

(a) $\frac{n-1}{n}$. Consider the case where the cost of the chosen expert is always 1, and the cost of each other expert is 0. Let k be the least-frequently chosen expert, and let m_k be the number of times that expert is chosen. This will result in a regret of $\frac{1}{T}(T - m_k)$

Since the best expert is the one that is chosen least often, the best strategy will try to maximize the number of times we choose the expert that is chosen least often. This means we want to choose all the experts equally many times, so expert k is chosen in at most T/n of the rounds. Therefore, $m_k \leq \frac{T}{n}$, thus the regret is at least $\frac{1}{T}(T - \frac{T}{n}) = \frac{n-1}{n}$. (Note here that even if our strategy is adaptive, i.e. it chooses an expert on day i based on the losses from days 1 to $i-1$, rather than committing to an expert for day i before seeing the loss for day 1, it still can't achieve regret better than $\frac{n-1}{n}$.)

(b) $(1 - \min_i p_i)$. Like in part (a), the distribution is fixed across all days, so we know ahead of time which expert will be chosen least often in expectation. Let $k = \text{argmin}_i p_i$ be the expert with least cost. Let $c_k^t = 0$ for all t, and let $c_i^t = 1$ for all $i \neq k$ and for all t. This way, $\mathbb{E}\left[\sum_{t=1}^T c_{i(t)}^t\right] = T\left(\sum_{i \neq k} p_i \cdot 1 + p_k \cdot 0\right) = T(1 - p_k)$. We also have $\min_i \sum_{t=1}^T c_i^t = 0$, so we end up with an expected regret of $\frac{1}{T}(T(1 - p_k) - 0) = 1 - p_k$.

To minimize the expectation of R is the same as maximizing $\min_i p_i$, which is achieved by the uniform distribution. This gives us regret $\frac{n-1}{n}$ (this is the same worst case regret as in part (b)).

3 Zero-Sum Battle

This is a solo question.

Two Pokemon trainers are about to engage in battle! Each trainer has 3 Pokemon, each of a single, unique type. They each must choose which Pokemon to send out first. Of course each trainer's advantage in battle depends not only on their own Pokemon, but on which Pokemon their opponent sends out.

The table below indicates the competitive advantage (payoff) Trainer A would gain (and Trainer B would lose). For example, if Trainer B chooses the fire Pokemon and Trainer A chooses the rock Pokemon, Trainer A would have payoff 2.

		Trainer B:		
		ice	water	fire
Trainer A:	dragon	-10	3	3
	steel	4	-1	-3
	rock	6	-9	2

Feel free to use an online LP solver to solve your LPs in this problem.
Here is an example of an online solver that you can use: `https://online-optimizer.appspot.com/`.

1. Write an LP to find the optimal strategy for Trainer A. What is the optimal strategy and expected payoff?

2. Now do the same for Trainer B. What is the optimal strategy and expected payoff? How does the expected payoff compare to the answer you get in part (a)?

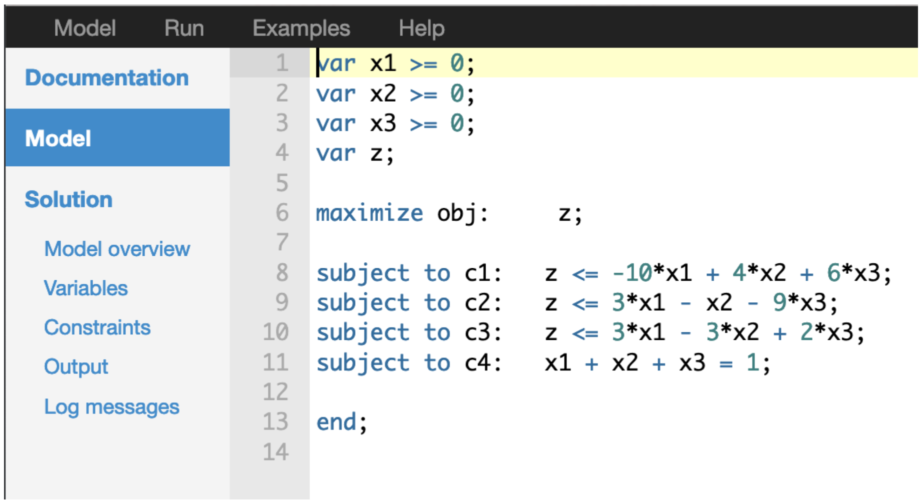
Solution:

1. d = probability that A picks the dragon type
   s = probability that A picks the steel type
   r = probability that A picks the rock type

$$\begin{aligned} &\max \quad z \\ -10d + 4s + 6r &\geq z && \text{(B chooses ice)} \\ 3d - s - 9r &\geq z && \text{(B chooses water)} \\ 3d - 3s + 2r &\geq z && \text{(B chooses fire)} \\ d + s + r &= 1 \\ d, s, r &\geq 0 \end{aligned}$$

   The optimal strategy is $d = 0.3346,\ s = 0.5630,\ r = 0.1024$ for an optimal payoff of -0.48.

   If you are using the website suggested in this problem, here is what you should put in the model tab:



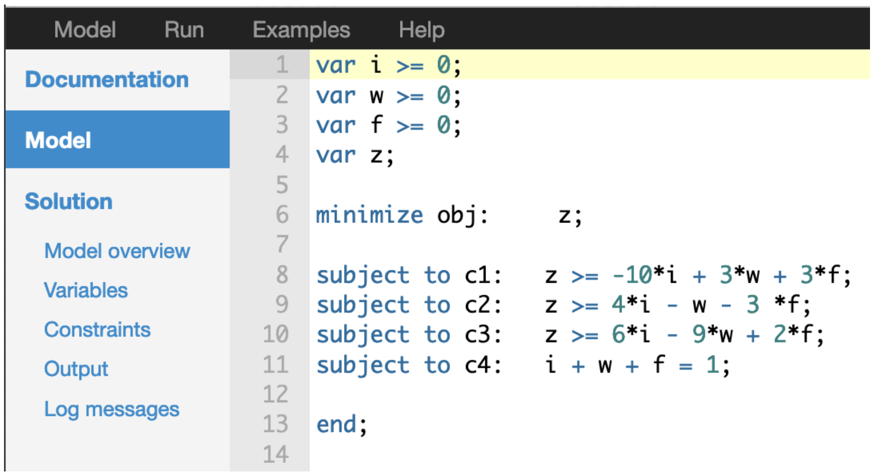
```
var x1 >= 0;
var x2 >= 0;
var x3 >= 0;
var z;

maximize obj:     z;

subject to c1:   z <= -10*x1 + 4*x2 + 6*x3;
subject to c2:   z <= 3*x1 - x2 - 9*x3;
subject to c3:   z <= 3*x1 - 3*x2 + 2*x3;
subject to c4:   x1 + x2 + x3 = 1;

end;
```

2. i = probability that B picks the ice type
   w = probability that B picks the water type
   f = probability that B picks the fire type

$$\begin{aligned} \min \quad & z \\ -10i + 3w + 3f &\le z && \text{(A chooses dragon)} \\ 4i - w - 3f &\le z && \text{(A chooses steel)} \\ 6i - 9w + 2f &\le z && \text{(A chooses rock)} \\ i + w + f &= 1 \\ i, w, f &\ge 0 \end{aligned}$$

B's optimal strategy is $i = 0.2677,\ w = 0.3228,\ f = 0.4094$. The value for this is -0.48, which is the payoff for A. The payoff for B is the negative of A's payoff, i.e. 0.48, since the game is zero-sum.

If you are using the website suggested in this problem, here is what you should put in the model tab:



```
var i >= 0;
var w >= 0;
var f >= 0;
var z;

minimize obj:     z;

subject to c1:   z >= -10*i + 3*w + 3*f;
subject to c2:   z >= 4*i - w - 3 *f;
subject to c3:   z >= 6*i - 9*w + 2*f;
subject to c4:   i + w + f = 1;

end;
```

(Note for grading: Equivalent LPs are of course fine. It is fine for part (b) to maximize B's payoff instead of minimizing A's. For the strategies, fractions or decimals close to the solutions are fine, as long as the LP is correct.)

4 Minimum ∞-Norm Cut

In the Minimum Infinity-Norm Cut problem, you are given a connected undirected graph $G = (V, E)$ with positive edge weights w_e, and you are asked to find a cut in the graph where the largest edge in the cut is as small as possible (note that there is no notion of source or target; any cut with at least one node on each side is valid).

Show how to reduce Minimum Infinity-Norm Cut to Minimum Spanning Tree in linear time. **Give a 3-part solution.**

Hint: Minimum Spanning Tree does not require edge weights to be positive.

Solution: Algorithm: First, negate all the edge weights in G, and pass this new graph to whatever minimum spanning tree algorithm we are using; this will give us a maximum spanning tree of the original graph. Remove the smallest-weight edge in this maximum spanning tree, and return the cut induced by its removal.

Proof of Correctness: First note that we correctly find a maximum spanning tree of the graph, since $\max_{\text{tree } T} \sum_{e \in T} w_e = \min_{\text{tree } T} \sum_{e \in T} -w_e$. It is similarly easy to see that we find a minimum infinity-norm cut in the maximum spanning tree of the graph (using any other edges from the spanning tree could not decrease the cut, since we used only the smallest possible edge).

It only remains to show that any cut of the nodes in the graph has exactly the same infinity norm in the graph overall as in the maximum spanning tree; this will prove the correctness of our algorithm (since we have already proven its correctness in the tree). To see this, we will use the cut property for maximum spanning trees (which follows immediately from the cut property for minimum spanning trees, applied to the negated graph). This property is: for any cut in the graph, its largest edge (or one of its largest edges, if the largest edge is not unique) must be contained in the maximum spanning tree. Since the infinity norm of the cut is equal to the weight of its largest edge, this means that the infinity norm of the cut in the tree is at least its infinity norm in the graph; it is also at most the infinity norm in the graph, since no edges exist in the tree which do not exist in the graph.

Runtime Analysis: Creating a new graph with every edge negated takes $O(|V| + |E|)$ time. Once we have the maximum spanning tree, the smallest-weight edge can be found with a simple $O(|E|)$ time search; once it is removed, the nodes in the two resulting components can be enumerated with a $O(|V| + |E|)$ time traversal. So overall, we have taken linear time.