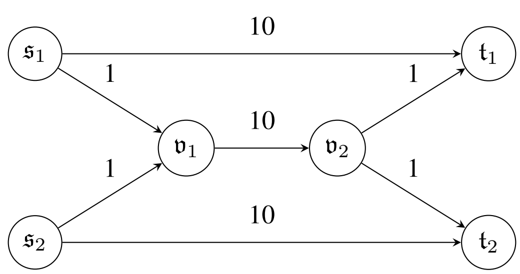
15780: GRADUATE AI (SPRING 2018)
Homework 4: Robust Optimization and Game Theory

Release: April 3, 2018,
Last Updated: April 12, 2018, 5:00pm
Due: April 16, 2018, 11:59pm

1 Transit Games [20 points]

In this problem, we will consider a class of games that we will call **transit games**. A transit game has n players on a directed graph G with non-negative edge weights. Each player i wants to move from a node $\mathfrak{s}_i$ to another node $\mathfrak{t}_i$; the strategy space for i therefore consists of all paths from $\mathfrak{s}_i$ to $\mathfrak{t}_i$.[1] Each edge e also has a cost c_e. The cost of each edge is split between all of the players who are using that edge; the cost for an individual player is the sum of the costs of all edges they are using.

For instance, consider the following graph:

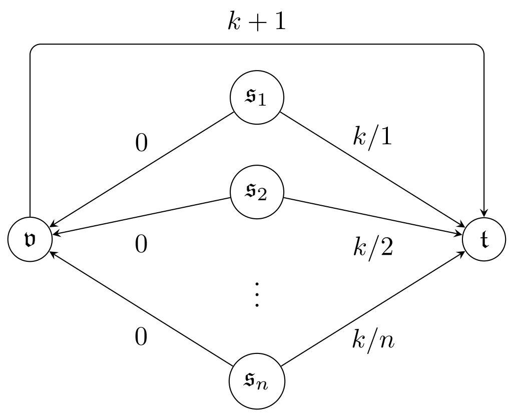


If both players were to use the paths $(\mathfrak{s}_i, \mathfrak{t}_i)$ for $i = 1$ and 2 respectively, they would both incur a total cost of 10, corresponding to the full cost of the single edge used by each. If instead, both players were to use the paths $(\mathfrak{s}_i, \mathfrak{v}_1, \mathfrak{v}_2, \mathfrak{t}_i)$, they would each incur a total cost of 7, because the edge $(\mathfrak{v}_1, \mathfrak{v}_2)$ is being used by both players and would therefore incur a cost of 5 for each player.

[1] You may disregard paths with cycles in them, since these will only have a higher cost than non-cyclic paths.

1.1 Finding Nash Equilibria [8 points]

Consider the following transit game with n players, where each player is at $\mathfrak{s}_i$ and wants to reach the common node $\mathfrak{t}$. Here, k is a strictly positive constant.



Observe that each player has a choice between using their unique edge $(\mathfrak{s}_i, \mathfrak{t})$, or moving to $\mathfrak{v}$ for free and using the shared edge $(\mathfrak{v}, \mathfrak{t})$, the cost of which depends on how many players in total are using that edge.

Find all Nash equilibria (pure or mixed) in this transit game.

1.2 Potential games [12 points]

For the remainder of this question, we will consider only pure strategy profiles. A game is an *exact potential game* if there exists a function $\phi : \prod_{i=1}^{n} S_i \to \mathbb{R}$ such that for all $i \in \mathbb{N}$, for all $\mathbf{s} \in \prod_{i=1}^{n} S_i$, and for all $s_i' \in S_i$, we have:

$$\mathrm{c}_i(s_i', \mathbf{s}_{-i}) - \mathrm{c}_i(\mathbf{s}) = \phi(s_i', \mathbf{s}_{-i}) - \phi(\mathbf{s}) \tag{1}$$

Intuitively, if a player i unilaterally changes their strategy to s_i', then the resulting change in the potential function ϕ is the same as the change in that player's cost. Any local minimum of ϕ therefore corresponds to a point where no player can decrease their cost by changing their strategy, i.e. a Nash equilibrium. Assuming the strategy space is finite, ϕ will always have a minimizer, leading to the following result:

Lemma 1. Every finite exact potential game has a pure Nash equilibrium.

Now, for any instance of the transit game, consider the following potential function. Let $n_e(\mathbf{s})$ be the number of players using the edge e in the strategy profile $\mathbf{s}$. Then:

$$\phi(\mathbf{s}) = \sum_{e} \sum_{k=1}^{n_e(\mathbf{s})} \frac{c_e}{k}$$

Prove that ϕ satisfies the property described in Equation 1, and conclude that *every* possible transit game always has a pure Nash equilibrium.

2 Stackelberg Security Games [35 points]

In class, we show a polynomial-time algorithm for computing the strong Stackelberg equilibrium (SSE) of security games with singleton schedules. (As a reminder, in an SSE, ties are broken in favor of the defender.) In this problem, we will derive a linear time algorithm to compute the SSE of a more restricted class of security games in which the attacker's utilities for all covered targets are identical, and in which any resource can cover any target. [2]

As a reminder, in this setting, there is a set of n targets $T = \{t_1, t_2, \ldots, t_n\}$, and the defender has m identical security resources (to protect these targets). Assume $m < n$. Each of these resources can protect any one target at a time. The defender pursues a randomized strategy when allocating these resources to protect targets. Such a strategy can be equivalently represented as a vector of coverage probabilities $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ over the n targets, where $\sum_{i=1}^{n} c_i = m$.

For this problem, we further assume that the attacker's utility for attacking a protected target is 0 and for attacking an unprotected target is strictly positive, i.e. $u_a^+(t) = 0$ and $u_a^-(t) > 0$ for any target t. When the attacker attacks target t, the expected utilities of the defender and attacker are then:

$$u_d(t, \mathbf{c}) = u_d^+(t) \cdot c_t + u_d^-(t) \cdot (1 - c_t),$$
$$u_a(t, \mathbf{c}) = u_a^+(t) \cdot c_t + u_a^-(t) \cdot (1 - c_t) = u_a^-(t) \cdot (1 - c_t).$$

For this analysis, you can use the following fact without proof:
Fact 1: The SSE is achieved for the restricted class of games described above when the coverage vector $\mathbf{c}$: (1) minimizes the attacker's maximum expected utility, and (2) maximizes the number of targets that have the same maximum utility for the attacker.[3] Observe that the solution does not depend on the defender's utilities (but only uses the assumption that $u_d^+(t) > u_d^-(t)$).

Form of optimal defender strategy. Let $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ denote the *optimal* defender strategy.

1. (6 points) Show that for some constant q_o and for all targets t_i with coverage probability $c_i > 0$, we have that
$$u_a^-(t_i) \cdot (1 - c_i) = q_o.$$
Hint: Consider some target t_o that is optimal for the attacker to attack. Using Fact 1, why must all targets t_i with $c_i > 0$ have $u_a(t_i, \mathbf{c}) = u_a(t_o, \mathbf{c})$?

2. (5 points) Show that the above statement (along with the optimality of $\mathbf{c}$)[4] implies
$$c_i = \max\left(1 - \frac{q_o}{u_a^-(t_i)}, 0\right)$$
for all targets t_i.

Together, these statements show that to compute $\mathbf{c}$, we need only obtain the value of q_o.

[2] This was added to reflect that in the general setting, resources may be restricted in the set of targets they can protect.

[3] This was first proved in: C. Kiekintveld, M. Jain, J. Tsai, J. Pita, M. Tambe, and F. Ordóñez. Computing Optimal Randomized Resource Allocations for Massive Security Games. In *AAMAS-09*, 2009

[4] Added phrase in brackets to indicate that optimality of $\mathbf{c}$ is also required here.

What does $\mathbf{q_o}$ look like? Let $t_{(1)}, t_{(2)}, \ldots, t_{(n)}$ denote the targets sorted in decreasing order of attacker utilities (when the target is uncovered), i.e.,

$$u_a^-(t_{(1)}) \geq u_a^-(t_{(2)}) \geq \cdots \geq u_a^-(t_{(n)}).$$

Let k_o be the largest index for which the corresponding target $t_{(k_o)}$ has positive coverage probability under the optimal defender strategy $\mathbf{c}$.

3. (4 points) Show that

$$q_o = \frac{k_o - m}{S_{k_o}},$$

where $S_{k_o} = \sum_{i=1}^{k_o} \frac{1}{u_a^-(t_{(i)})}$.

This statement indicates that to find q_o, we need only find the corresponding value of k_o.

4. (8 points) Argue that the following is true: q_o is in the interval $\left[u_a^-(t_{(k_o+1)}), u_a^-(t_{(k_o)})\right)$. Further, if for some k we have that the corresponding $q = \frac{k-m}{S_k} < u_a^-(t_{(k+1)})$, then it must be that $k_o > k$. Similarly, if $q = \frac{k-m}{S_k} \geq u_a^-(t_{(k)})$, then it must be that $k_o < k$.

Using the above, we can now construct an algorithm to find the value of k_o.

The search for $\mathbf{k_o}$! We find the value k_o corresponding to the SSE solution using *binary search* on the attacker utilities (without sorting the targets). To do this, we maintain three lists of targets:

- T^+: targets t_i that will certainly be covered with positive probability ($c_i > 0$),
- T^-: targets t_i that will certainly not be covered ($c_i = 0$),
- $T^?$: targets t_i that we are undecided about potentially covering with positive probability.

We start with all targets in the "undecided" list, i.e., $T^? = \{t_1, t_2, \ldots, t_n\}$ and $T^+ = T^- = \emptyset$. At each stage, we will update the value $S^+ = \sum_{t_i \in T^+} \frac{1}{u_a^-(t_i)}$, with an initial value of $S^+ = 0$.

We then iterate over the following steps to find the value of k_o:

- Step 1:
  - Compute the median d of the attacker utilities for targets in $T^?$, i.e. $d = \mathrm{median}(\{u_a^-(t_i) \mid t_i \in T^?\})$. [5]
  - Partition $T^?$ into two sublists based on this median: $T^\ell = \{t_i \mid u_a^-(t_i) \geq d, t_i \in T^?\}$ and $T^r = \{t_i \mid u_a^-(t_i) < d, t_i \in T^?\}$.
  - Define t_{k^ℓ} as the target in T^ℓ with the smallest value of u_a^-, and t_{k^r} as the target in T^r with the largest value of u_a^-.
- Step 2: Compute $S^\ell = \sum_{t_i \in T^\ell} \frac{1}{u_a^-(t_i)}$, $S_{k^\ell} = S^+ + S^\ell$, and $q = \frac{(|T^+|+|T^\ell|)-m}{S_{k^\ell}}$.[6]

[5] The median d can be computed in time linear in $|T^?|$ even if the targets are not sorted (by the u_a^- values).

[6] The previous formula for q was $q = \frac{k^\ell - m}{S_{k^\ell}}$ which is incorrect.

- <u>Step 3</u>: Update $T^+, T^?, T^-$, and S^+ as follows:[7]
  - If $q < u_a^-(t_{k^r})$, then: $\boxed{\text{A}}$.
  - If $q \geq u_a^-(t_{k^\ell})$, then: $\boxed{\text{B}}$.
  - If $u_a^-(t_{k^r}) \leq q < u_a^-(t_{k^\ell})$, then: $\boxed{\text{C}}$.

5. (8 points) Finish the above algorithm by describing cases $\boxed{\text{A}}$, $\boxed{\text{B}}$, and $\boxed{\text{C}}$. Specifically, for each case, write the update equations for $T^+, T^?, T^-$, and S^+, argue if we should terminate, and briefly justify based on our above analysis.
6. (4 points) Argue that our algorithm runs in time linear in the number of targets, i.e. the time complexity is $O(n)$.

3 Robust Optimization [20 points]

In this problem, we will consider variations of the robust optimization problem discussed in class. Assume we are given m training points $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)}) \in \mathbb{R}^D \times \{-1, +1\}$. Consider a monotonically decreasing classification loss $\mathcal{L} : \mathbb{R} \to \mathbb{R}$ and a hypothesis function $h_\theta(x) = \theta^T x$ mapping from $\mathbb{R}^D$ to $\mathbb{R}$ for $\theta \in \mathbb{R}^D$.

1. (5 points) **Robust optimization with respect to ℓ_1 balls.** Reduce the following optimization problem over the variables θ and $\Delta^{(1)}, \Delta^{(2)}, \ldots, \Delta^{(m)}$ to an optimization problem over only θ:

$$\underset{\theta}{\text{minimize}} \ \frac{1}{m} \sum_{i=1}^{m} \max_{\|\Delta^{(i)}\|_1 \leq \epsilon} \mathcal{L}(h_\theta(x^{(i)} + \Delta^{(i)}) \cdot y^{(i)})$$

2. (5 points) **Robust optimization with respect to ℓ_2 balls.** Reduce the following optimization problem over the variables θ and $\Delta^{(1)}, \Delta^{(2)}, \ldots, \Delta^{(m)}$ to an optimization problem over only θ:

$$\underset{\theta}{\text{minimize}} \ \frac{1}{m} \sum_{i=1}^{m} \max_{\|\Delta^{(i)}\|_2 \leq \epsilon} \mathcal{L}(h_\theta(x^{(i)} + \Delta^{(i)}) \cdot y^{(i)})$$

3. (10 points) **Robust optimization with respect to ℓ_∞ balls for multi-class classification.** Consider a K-class classification problem where the training points $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$ belong to $\mathbb{R}^D \times \{1, 2, \ldots, K\}$. The hypothesis function

$$h_\Theta(x) = \Theta x = \begin{bmatrix} \theta_1^T \\ \theta_2^T \\ \vdots \\ \theta_K^T \end{bmatrix} x$$

[7] We can define false targets $t_{(0)}$ and $t_{(n+1)}$ with $u_a^-(t_{(0)}) \equiv \infty$ and $u_a^-(t_{(n+1)}) \equiv 0$ to avoid edge cases here. As such, you will not need to deal with edge cases in your argument.

maps from $\mathbb{R}^D$ to $\mathbb{R}^K$ for $\Theta \in \mathbb{R}^{K\times D}$. We consider the softmax loss

$$\mathcal{L}(h_\Theta(x), y) = \log\left(\sum_{k=1}^{K} e^{h_\Theta(x)_k}\right) - h_\Theta(x)_y.$$

We will now upper bound the robust optimization loss. Specifically, we will define a new function $\tilde{h}_\Theta$ on the training points such that for every $x^{(i)}$, the kth co-ordinate of $\tilde{h}_\Theta(x^{(i)})$ is defined as follows:

$$\tilde{h}_\Theta(x^{(i)})_k = \theta_k^T x + \epsilon\|\theta_k - \theta_{y^{(i)}}\|_1.$$

That is, the output of the classifier is increased by a small amount on all but the correct class. Show that we can upper bound the robust optimization loss by computing softmax loss on the output of $\tilde{h}_\Theta$ for the training datapoints, i.e., show that:

$$\frac{1}{m}\sum_{i=1}^{m} \max_{\|\Delta^{(i)}\|_\infty \le \epsilon} \mathcal{L}(h_\Theta(x^{(i)} + \Delta^{(i)}), y^{(i)}) \le \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\tilde{h}_\Theta(x^{(i)}), y^{(i)}).$$

Hints:

(a) First, bring the second term in the softmax loss into the $\log$ term.

(b) Next, to show an upper bound, bring the $\max_{\Delta^{(i)}}$ to the front of each term in the summation $\sum_{k=1}^{K}$ inside the $\log$. (Remember to justify why you can do this to show the upper bound.)

4 Programming: Stackelberg Strategies [25 points]

In a 2-player normal form game, a Stackelberg strategy is where one of the players is a leader and the other is a follower. In contrast to the default situation where both players pick their respective strategies at the same time, a Stackelberg strategy is when the leader, which is identified as player 1, first commits to a (mixed) strategy which the follower, player 2, knows. Then player 2 commits to his own strategy using his knowledge of player 1's strategy.

An optimal Stackelberg strategy would be a Stackelberg strategy where player 1's expected utility is maximized. The optimal Stackelberg strategy can be computed in polynomial time by solving multiple LPs. See Slide 6 of Lecture 20 for a description of the algorithm.

Given a 2-player normal form game, your task is to implement the function `stackelberg(u1, u2)` in `stackelberg.py` which will return the optimal Stackelberg strategy for the given game. You can use `cvxopt` or `cvxpy` to solve the LPs. Your function should return numpy arrays, not datatypes from these libraries. If there is a tie in the expected utility between two strategies of player one (i.e. the expected utility is off by an absolute error of 1e-5), you should return the optimal strategy induced by the lowest indexed pure strategy of player 2.

5 Submitting to Autolab

Create a tar file containing your writeup and the completed `stackelberg.py` modules for the programming problems. Make sure that your tar has these files at the root and not in a subdirectory. Use the following commands from a directory with your files to create a `handin.tgz` file for submission.

```
$ ls
stackelberg.py  writeup.pdf  [...]
$ tar cvzf handin.tgz writeup.pdf stackelberg.py
a writeup.pdf
a stackelberg.py
$ ls
handin.tgz  stackelberg.py  writeup.pdf  [...]
```