15780: GRADUATE AI (SPRING 2019)

Homework 4: Robust Optimization and Game Theory

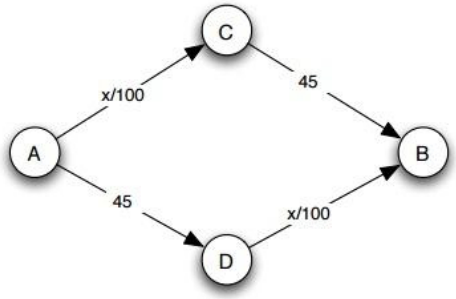
Release: April 11, 2019,
Due: April 25, 2019, 5:00pm

We will use scientific Python for the implementation portions in this course. If you have not used Python before, we recommend downloading the Anaconda distribution (`https://www.continuum.io/downloads`) and looking through introductory resources like Google's Python Class (`https://developers.google.com/edu/python/`) and the Python Beginner's Guide (`https://wiki.python.org/moin/BeginnersGuide`). If you have not used scientific Python before, we recommend following introductions to NumPy (`http://www.numpy.org/`) and matplotlib (`http://matplotlib.org/`).

Please make sure your code submission works with the grading environment, which uses Python 3.7 and numpy 1.15.0.

1 Game Theory (30 points) [Ivan]

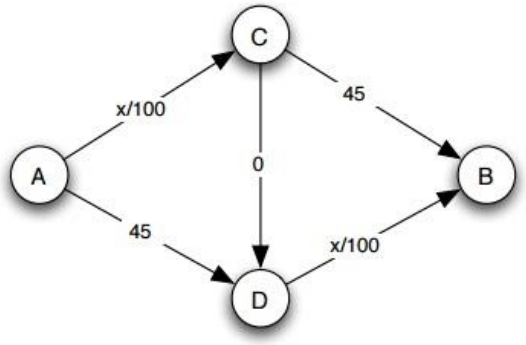
Suppose there are 4000 cars in a city. Every morning, each of the 4000 cars want to go from point A to point B in the city. The figure below shows the possible routes that any car can take:



Here, the road segments are shown by directed edges (note that all the road segments are one ways). The

number written on any arrow represents the cost that will be incurred by any car that travels on that road segment; the cost $x/100$ means that if a total of x cars travel on that road segment then each of these x cars will incur a cost of $x/100$. The total cost incurred by any car in going from A to B is the sum of the costs on each individual road segment taken by the car. For example, if 1000 cars take the route ACB and 3000 cars take the route ADB then each car taking the route ACB incurs a cost (1000/100+45 = 55) and each car taking the route ADB incurs a cost (45+3000/100 = 75).

1. **[10 points]** Find a strategy (route) for each car such that they result in a strict Nash equilibrium. For simplicity in this homework you may assume that the cars can only have pure (deterministic) strategies.
2. **[10 points]** Now consider the set of road segments and associated costs given by the following figure:



   Find a strategy (route) for each car such that they result in a strict Nash equilibrium. Again, for simplicity you may assume that the cars can only have pure (deterministic) strategies.
3. **[10 points]** Comment on the qualitative comparison of the strategies you obtained in the two subquestions.

2 Robust Optimization (35 points) [Chun Kai]

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. **[10 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. **[10 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. **[15 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

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

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:

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

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.)

3 Programming: Stackelberg Strategies (35 points) [Junjue]

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 her/his own strategy using her/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 7 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 (1.2.3)` or `cvxpy (1.0.14)` 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.

4 Submitting to Diderot

Create a tar file containing your writeup for the first two problems and the completed `stackelberg.py` module for the programming problem. 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
```