Josh Palmer and Liyiming Ke

Proposal: Reinforcement Learning Applied to Adversarial Manipulations of MDPs

Introduction:

Reinforcement Learning (RL) is a new subject for some; as such it would be useful to define it by outlining the three characteristics associated with it:[1] (1) "being closed-loop in an essential way" (i.e. "[the] learning system's actions influence its later inputs"), (2) "not having direct instructions as to what actions to take", and (3) "the consequences of actions…[playing] out over extended time periods". RL is different enough from supervised and unsupervised learning --- there is neither labeled data guiding what actions should be taken at a given state nor a hidden structure to discover --- that it belongs in its own proper category of learning.

We would like to apply RL to tackle a class of problems: the manipulation of Markov Decision Problems (MDPs) by an adversarial agent. We are interested because, as our world increasingly becomes an "Internet of things", cyberattacks are more and more debilitating. For example, imagine an adversary hacking a self-driving car; the software needs to have learned what to do in such an attack. Recent related literature has included the following:

- An adversary made minor pixel-wise changes to images that are imperceptible to humans and yet "induced misclassification by a CNN"[2] (2014)
- A response to the above paper that critiqued it as being unrealistic because the adversary would require access to the CNN, whose abilities do differ from those of a human. Furthermore, **all ML algorithms** are susceptible to such manipulations.[3] (2015)

[1] Source: https://webdocs.cs.ualberta.ca/~sutton/book/bookdraft2016sep.pdf; page 2
[2] Source: http://cs.nyu.edu/~zaremba/docs/understanding.pdf
[3] Source: http://www.kdnuggets.com/2015/01/deep-learning-flaws-universal-machine-learning.html

- RL has been applied to zero-sum, symmetric adversarial games --- Othello/Reversi[4] (2010) and an RTS called Wargus.[5] (2012)

We did not encounter the application of RL the case of an asymmetric adversarial MDP, where the player and adversary are distinct in their goals --- the player seeks to maximize their utility from their environment (as if there were no adversary) while the adversary surreptitiously manipulates this environment and/or the player's perception of it.

Approach (Including Steps and Timeline):

Our project will at first focus on application to a toy model:

- an n x m grid that contains the player's initial position, the player's goal, and randomly distributed "holes" in which the player will fall to their death
- at each time step, the player must decide a valid direction to move and then carry out that move. The player is very perceptive, "knowing" where the holes and the goal are
- at each time step, the adversary can manipulate the player's perception of the game state --- altering where they believe the holes to be

We will use Q-learning, one specific flavor of RL, that is favorable because "it doesn't require knowing the dynamics of the environment" and learns quicker through bootstrapping ("estimating over existing estimates").[6] The first point is especially important because the player is unsure of how exactly the game state is changing. This is the breakdown for how the project will be completed throughout the semester:

- **Step 0:** better understand RL and the current literature
    - complete PyTorch Q-Learning tutorial and read lecture slides about RL

[4] Source: www.intelligence.tuc.gr/lib/downloadfile.php?id=433
[5] Source: http://www.aaai.org/ocs/index.php/AIIDE/AIIDE12/paper/viewFile/5515/5734
[6] Source: same as 5

    - **Deadline:** 2/1

- **Step 1:** write RL algorithm for player
    - input: list of tuples of (State, Action, Reward) (offline for simplicity)
    - output: policy function → tells optimum action to take in state
    - **Deadline:** 2/15
- **Step 2:** design adversary
    - Substep A: manually generate noise/manipulation such that player cannot reach their goal
        - **Deadline:** 2/22
    - Substep B: figure out how to best quantify how "successful" the adversary has been with misleading/redirecting player through their manipulations
        - (i.e. deviation of player's policy function from true policy function)
        - **Deadline:** 3/1
    - Substep C: codify way to automatically generate manipulations
        - need to impose restrictions (i.e. upon every move to a new state, can only modify 10% of pixels)
        - **Deadline:** 3/15
- **Step 3:** look into similar flavors/applications of RL, recycling prior work
    - **Deadline:** 4/12
- **Step 4:** analyze results and compile report
    - **Deadline:** 4/24

Note: We plan to meet every Wednesday for 30 minutes for purposes of planning and accountability. And we can meet on the weekend for collaborative work.