# MS&E 338

## Q-Learning in OpenAI gym[*]

In this homework, you will implement the off-policy and on-policy Q-learning algorithms (a.k.a. episodic Q-learning and SARSA respectively) with $\epsilon$-greedy and Boltzmann exploration in order to solve a modified version of the cartpole problem from OpenAI gym.

**Problem**:
A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 (right), 0 or -1 (left) to the cart. The non-zero forces have a cost of 0.1. The cart is old and with probability 0.1 any non-zero force moves the cart to the opposite direction than it is supposed to. The pendulum starts upright, and the goal is to prevent it from falling over. For every timestep that the pole remains upright, a reward of +1 is generated with probability 0.75 and reward 0 is generated with probability 0.25. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

**Homework**:

1. Implement (a) episodic Q-learning with Boltzmann exploration, (b) episodic Q-learning with $\epsilon$-greedy exploration, (c) SARSA with Boltzmann exploration, (d) SARSA with $\epsilon$-greedy exploration.
2. Run simulations of the above algorithms with different values for the $Q$-learning rate and the exploration parameters $\epsilon$ and $\beta$. Use as episode horizon $H = 100$ and simulate up to $L = 10,000$ episodes. Plot the reward per episode in separate plots for each one of the (a), (b), (c), and (d) for the different $Q$-learning rates and exploration parameters you tried. To get nice plots without too much noise, make sure you run enough simulations.
3. Discuss your findings regarding the performance of different algorithms and parameterizations. Which ones perform best?
4. (Extra Credit) Pick a parameterization of your choice for each (a), (b), (c), and (d) and

[*]Material prepared by Maria Dimakopoulou for the course of Professor Benjamin Van Roy. For questions, shoot an email to madima@stanford.edu or bvr@stanford.edu.

animate how each algorithm controls the cartpole after 100, 500, 1,000, 5,000, 10,000 learning episodes.

Please submit a zip with your code, your report with your plots and the discussion of your results, and (optionally) your animations.

**Code**:
For this homework, you will need to have Python 3.5+ installed. Simply install gym using pip, `pip install gym`.
We have provided you with some code:

- `environments.py`: This file contains the implementation of the cartpole environment as described above, including the rendering method you may need for your animations in (4).
- `agents.py`: This file contains a template for the reinforcement learning agents. The random agent and the constant action agent are implemented. This file also contains a feature extractor implementation that converts an observation returned by the environment to a tabular state (see usage in lines 72, 87).
- `example.py`: This file demonstrates a single simulation of the random agent, as a tutorial of how to use the provided code. Play with it to familiarize before diving in.