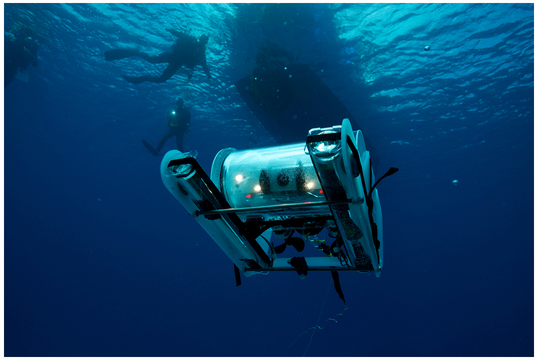
Reinforcement Learning With Imperfect Safety Constraints

Jin Woo Ro, Gerald Lüttgen, Diedrich Wolter
University of Bamberg
March 1, 2022

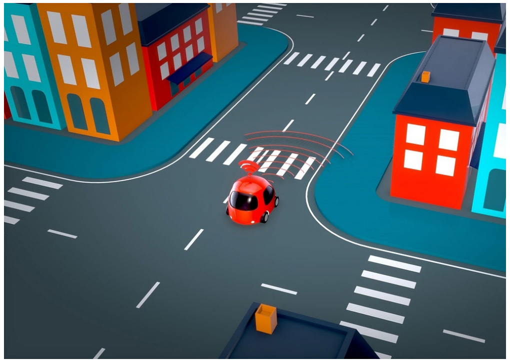
Reinforcement Learning

RL is useful for dealing with the environment that changes dynamically or is not fully known in the system design time

Exploration Robots (NASA: Mars Rover)



Underwater robots



Self-driving Cars

Mission-critical or Safety-critical Applications!!

Safety in Reinforcement Learning

- Ensuring safety in RL can be formulated as a constraint satisfaction problem

- Various existing approaches include:
  1. Reward shaping [1]
     - Multiple rewards to indicate different system quality (including safety)
  2. Safe Policy Extraction [2]
     - Safety monitors (runtime verification) to detect unsafe actions
  3. Constrained Markov Decision Process (CMDP) [3]
     - Choose from the actions that satisfy constraints

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. *Human-level control through deep reinforcement learning*. Nature, 518(7540), pp. 529-533, 2018

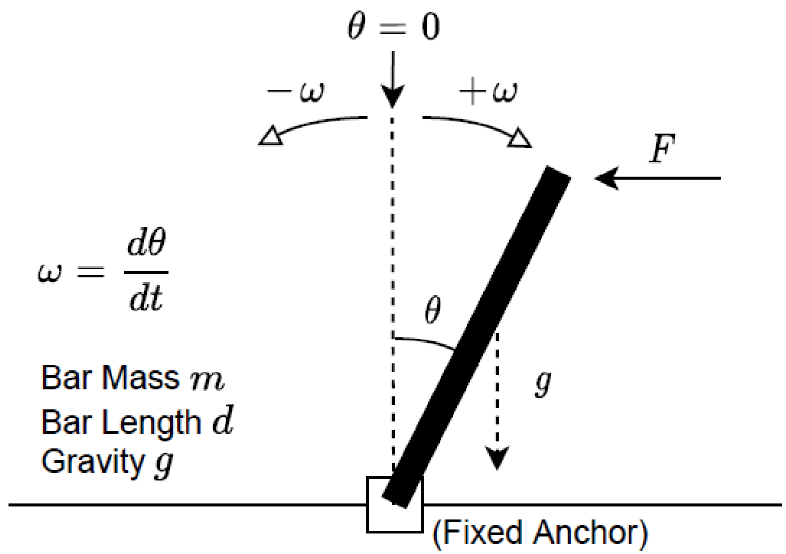
[2] Mirchevska, B., Pek, C., Werling, M., Althoff, M., & Boedecker, J., *High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning*. In 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2156-2162. IEEE, 2018

[3] Altman, E. *Constrained Markov decision processes* (Vol. 7). CRC Press, 1999

Research Problem

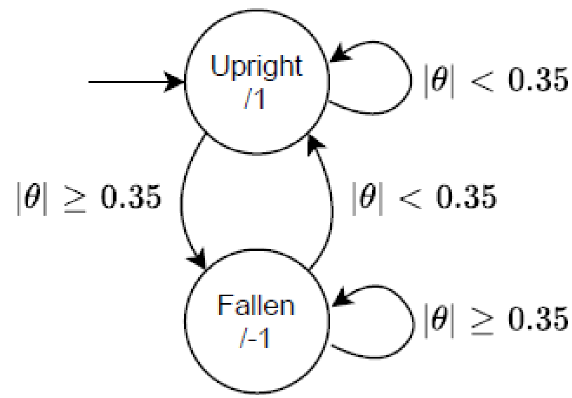
- Safety constraints are invariant in existing approaches
  - Constraints need to be specified correctly
- But, what if the safety constraints are *imperfect*?
  - Can be due to unexpected environment changes or system changes such as damages
  - i.e., imperfect constraints can result in false-negative detections (constraints indicate the state is safe, but actually is not safe)
- Need to correctly detect the unsafe system states that are not captured by the constraints (namely, *hidden* unsafe states)

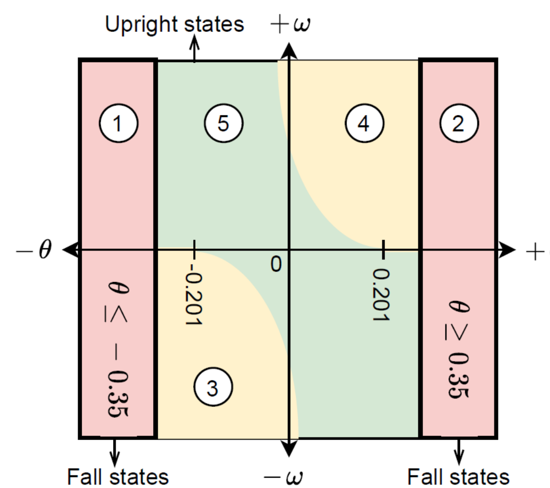
Motivating Example: *Inverse Pendulum*



θ : angle from the upright position
ω : angular velocity (clockwise = positive)
m : bar mass (2kg)
d : bar length (1m)
g : gravity (9.8ms^{-2})
F : horizontally force acting in the right or left direction (2N or -2N, respectively)

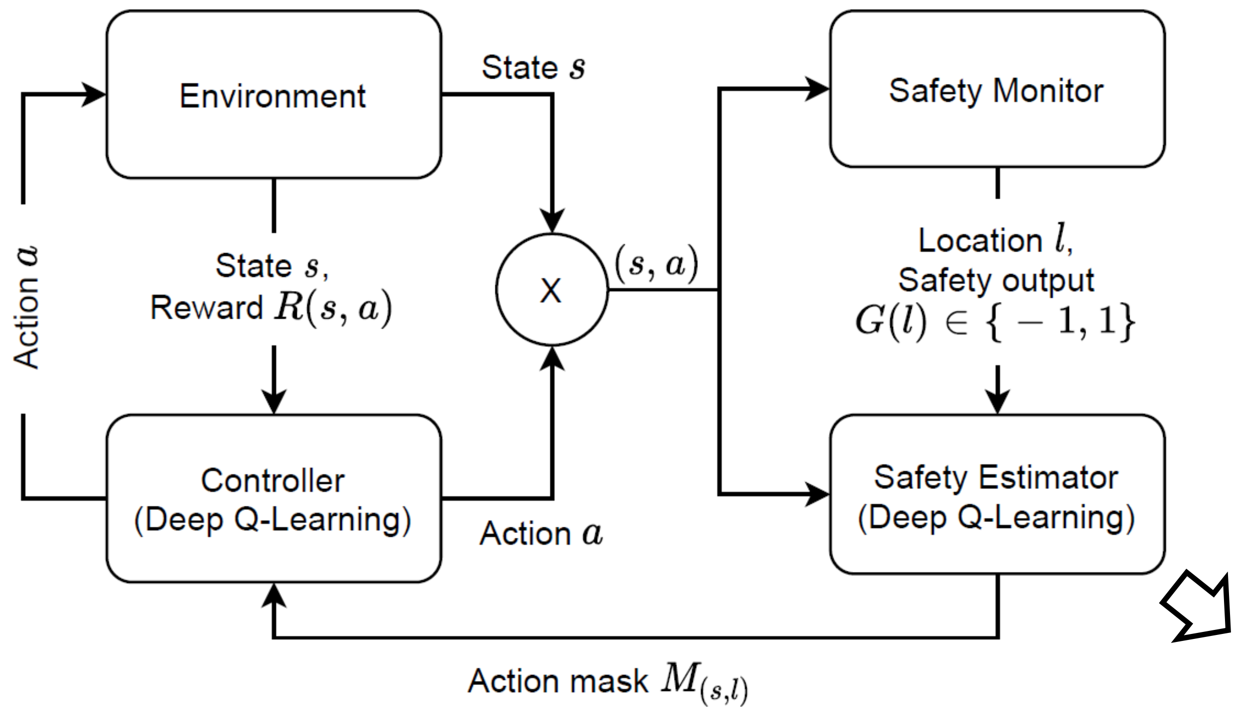
Imperfect Safety Monitor





Proposed Idea in a Nutshell

- *Safety Estimator* learns from safety monitors
  - Estimate *hidden* unsafe states
  - Generate action masks to indicate unsafe actions



Two learning components:
- Controller
- Safety Estimator

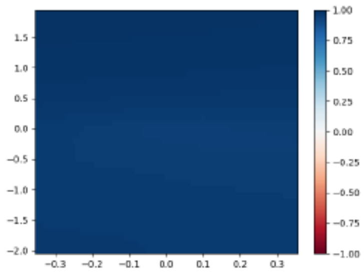
Modified the bellman equation for learning

Inverse Pendulum Result

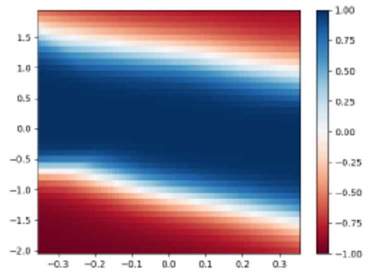
- The controller's learning speed is improved
  - Less safety violations in the early learning phase => faster convergence

*Double buffer case: safe and unsafe system state data are stored in two buffers separately for training Safety Estimator.

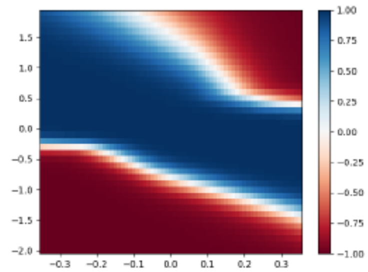
Safety Estimator's Learning over Episodes



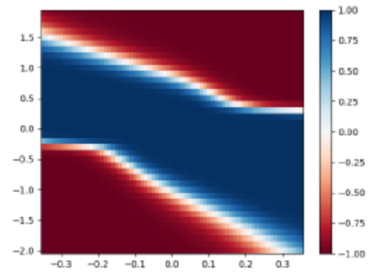
(a) Episode 0



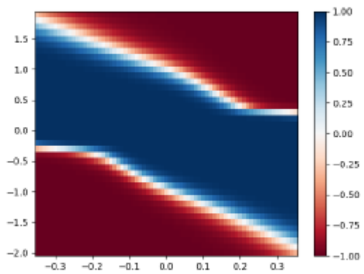
(b) Episode 20



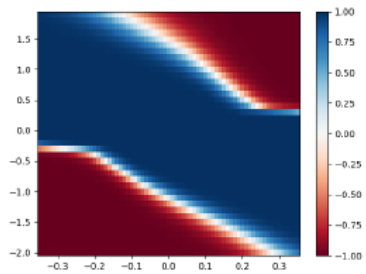
(c) Episode 40



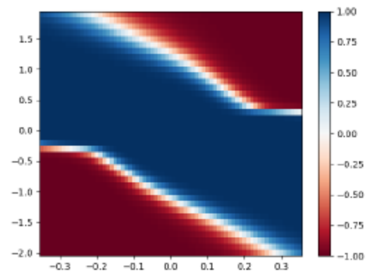
(d) Episode 140



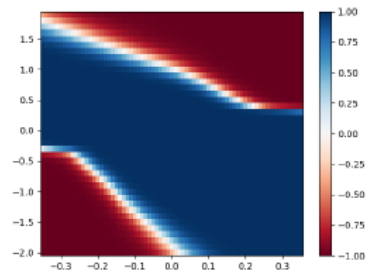
(e) Episode 180



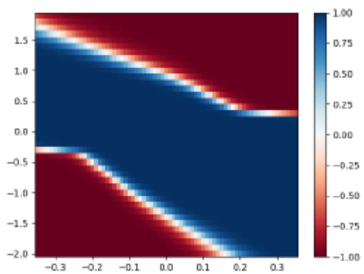
(f) Episode 220



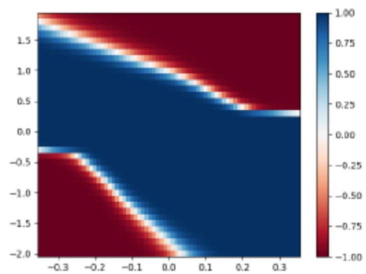
(g) Episode 260



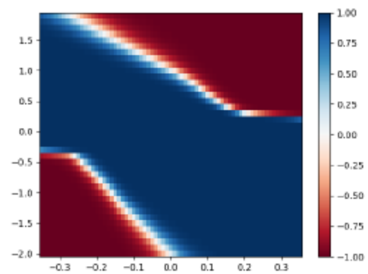
(h) Episode 300



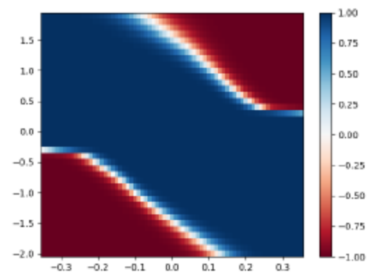
(i) Episode 400



(j) Episode 420



(k) Episode 440



(l) Episode 460

Conclusion & Future Work

- We addressed the problem of dealing with an imperfect safety monitor
- Safety Estimator is used to detect false-negative outputs of the safety monitor via learning
- Experimental result shows that the safety violation occurrence is reduced (faster convergence)
- We are currently working on:
  - How do we ensure that what Safety Estimator learned is an over-approximation of the actual unsafe states?
  - How can we feedback the learned knowledge of Safety Estimator to safety monitor, and update the monitor accordingly?