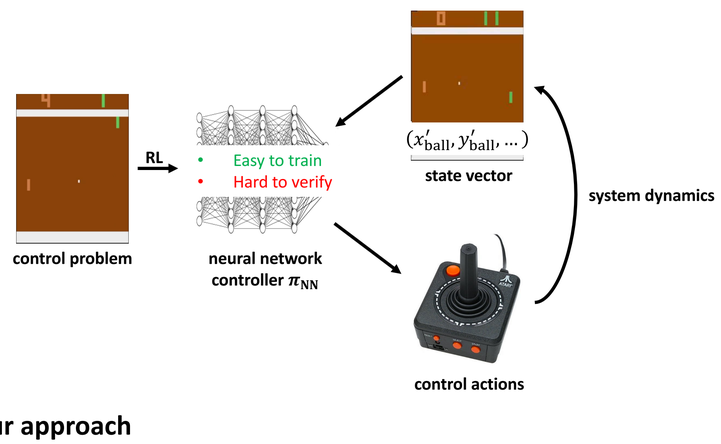
Verifiable Reinforcement Learning via Policy Extraction

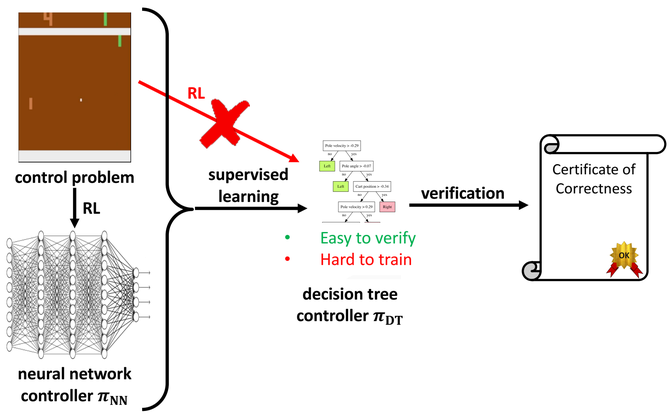
Osbert Bastani[1,2], Yewen Pu[1], Armando Solar-Lezama[1]
[1]Massachusetts Institute of Technology
[2]University of Pennsylvania

Overview

- **Motivation**
  - Deep reinforcement learning (RL) has been successfully applied to solve a number of challenging control tasks
  - However, it's real-world applicability remains limited due to safety concerns in using learned, blackbox controllers



- **Our approach**
  - Decision tree controllers are easy to verify but hard to train
  - Use imitation learning to train a decision tree controller

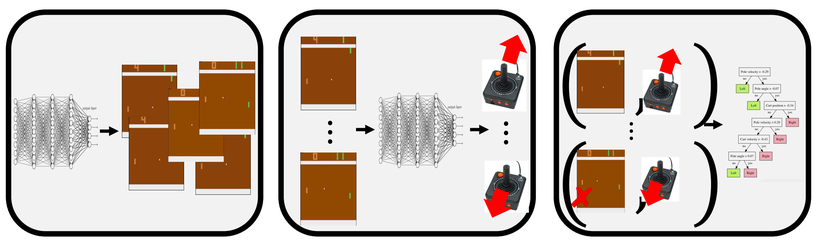


Problem Formulation

- **Input**
  - Markov decision process (MDP) $M = (S, A, T, R)$
  - Neural network (NN) controller $\pi_{\text{NN}}: S \to A$
  - Q function, where $Q(s, a)$ measures how good action a is in state s (obtained from deep RL algorithms)
- **Output**
  - Decision tree (DT) controller $\pi_{\text{DT}}: S \to A$

Background on Imitation Learning

- **Naïve algorithm**
  - **Step 1:** Use NN to generate states
  - **Step 2:** Use NN to label action for each state
  - **Step 3:** Use supervised learning to train DT

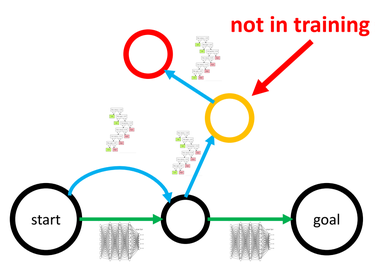


Step 1: Use NN to generate states

Step 2: Use NN to obtain actions

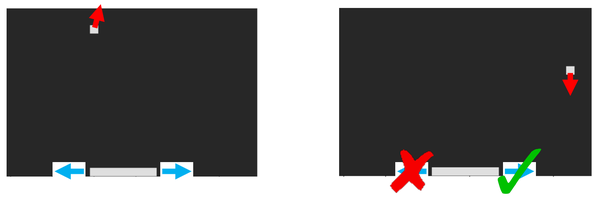
Step 3: Use supervised learning to train a decision tree

- **DAgger Algorithm (Ross 2011)**
  - **Problem:** DT makes mistakes and sees new states
  - **Solution:** Use NN to label states



VIPER Algorithm

- **Insight:** Want to prioritize accuracy on "critical states" where the gap between the optimal action and the remaining actions is large
- **Idea:** Weight state-action pairs in the loss using the Q function



actions are similar (non-critical state)

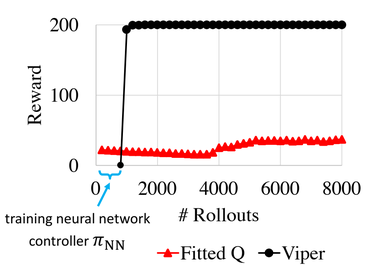
$$\underbrace{Q(s, \pi_{\text{NN}}(s))}_{\text{optimal } Q \text{ value}} \approx \underbrace{\min_{a \in A} Q(s, a)}_{\text{worst-case } Q \text{ value}}$$

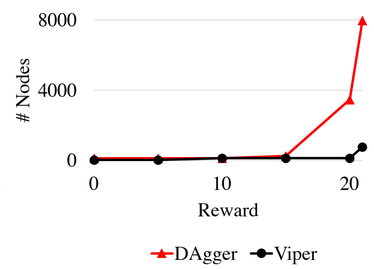
must move right! (critical state)

$$\underbrace{Q(s, \pi_{\text{NN}}(s))}_{\text{optimal } Q \text{ value}} \gg \underbrace{\min_{a \in A} Q(s, a)}_{\text{worst-case } Q \text{ value}}$$

Evaluation

- **Comparison to reinforcement learning for DTs (below, left)**
  - Fitted Q iteration (RL algorithm for learning decision trees)
  - Cart-pole control problem
- **Comparison to Dagger (below, right)**
  - Atari Pong (symbolic state space)





Case Study: Verifying Toy Pong

- **Toy pong**
  - **Problem:** $S = \mathbb{R}^5$, $A = \{\text{left}, \text{right}, \text{stay}\}$
  - **NN:** Trained using policy gradients, 600 neurons
  - **DT:** Extracted using VIPER, 31 nodes
  - **Correctness:** Never lets the ball leave the arena
- **Verification**
  - **Inductive invariant:** $s(0) \in \text{blue} \Rightarrow s(T) \in \text{blue}$ (below, left)
  - **Algorithm:** Dynamics and DT controller are piecewise linear, so we can encode correctness as an SMT formula
- **Results**
  - Solved by Z3 in < 5 seconds
  - Finds an error when ball starts on the right (below, right)
  - Fixed when paddle is slightly longer!

References

Ross, Gordon, & Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. AISTATS 2011.