The Open-Source TEXPLORE Code Release for Reinforcement Learning on Robots

TODD HESTER AND PETER STONE
The University of Texas at Austin
Austin, TX 78712 USA
{todd,pstone}@cs.utexas.edu

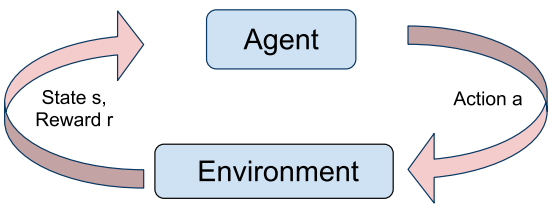
ABSTRACT

- Applying RL to robots could enable them to learn many useful tasks
- Learning on robots presents four specific challenges for RL
- The TEXPLORE algorithm addresses all four challenges
- TEXPLORE is released as an open-source ROS repository
- It is easy to integrate TEXPLORE with robots already running ROS

Motivation

- Robots have the potential to solve many problems
- We need methods for them to learn and adapt to new situations

Reinforcement Learning



- Value function RL has string of positive theoretical results [Watkins 1989, Brafman and Tennenholtz 2001]
- Could be used for learning and adaptation on robots
- **Model-free Methods**:
  - Learn a value function directly from interaction with environment
  - Can run in real-time, **but** not very sample efficient
- **Model-based Methods**:
  - Learn model of transition and reward dynamics
  - Update value function using model (planning)
  - Can update action-values without taking real actions in the world

Velocity Control of an Autonomous Vehicle

- Upgraded to run **autonomously** by adding shift-by-wire, steering, and braking actuators.
- 10 second episodes (at 20 Hz: 200 samples / episode)
- State:
  - Current Velocity
  - Desired Velocity
  - Accelerator Pedal Position
  - Brake Pedal Position
- Actions: Do nothing, Increase/decrease brake position by 0.1, Increase/decrease accelerator position by 0.1
- Reward: -10.0 * velocity error (m/s)

Robot Learning Challenges

1. Learning algorithm must learn in very few actions (be **sample efficient**)
2. Learning algorithm must take actions **continually** in real-time (while learning)
3. Learning algorithm must handle **continuous** state
4. Learning algorithm must handle **delayed** actions

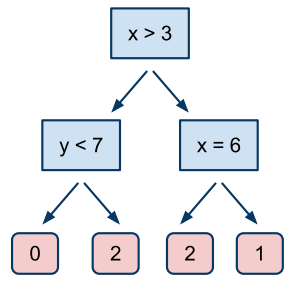
The TEXPLORE Algorithm

1. Limits exploration to be **sample efficient**
2. Selects actions continually in **real-time**
3. Handles **continuous** state
4. Handles actuator **delays**

Available publicly as a **ROS package**:
www.ros.org/wiki/rl-texplore-ros-pkg

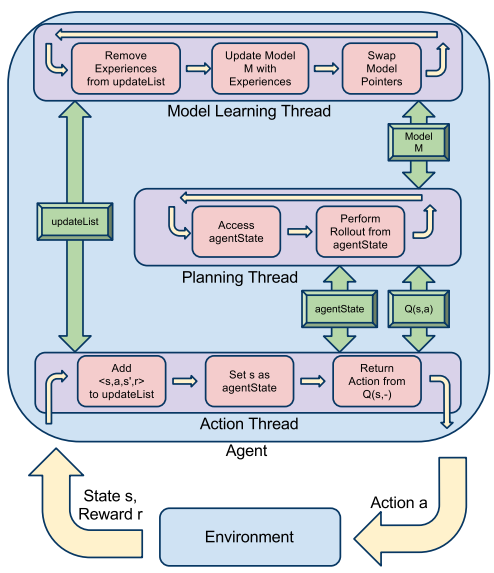
Challenge 1: Sample Efficiency

- Treat model learning as a supervised learning problem
  - **Input:** State and Action
  - **Output:** Distribution over next states and reward
- **Factored** model: Learn a separate model to predict each next state feature and reward
- **Decision Trees**: Split space into regions with similar dynamics



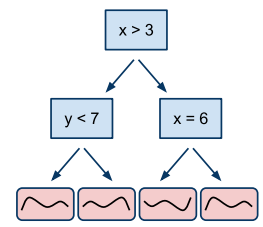
- **Random Forest**: Average predictions of m trees
- Acting greedily w.r.t. the average model balances predictions of optimistic and pessimistic models
- **Limits** the agent's exploration to state-actions that appear promising, while avoiding those which may have negative outcomes

Challenge 2: Real-Time Action



- Model learning and planning on parallel threads
- Use sample-based planning (anytime)
- Mutex locks on shared data

Challenge 3: Continuous State

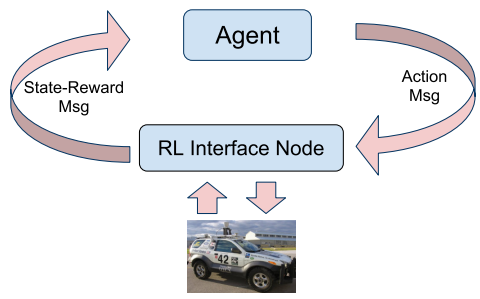


- Use regression trees to model continuous state
- Each tree has a linear regression model at its leaves
- Discretize state space for value updates from UCT, but still plan over continuously valued states

Challenge 4: Actuator Delays

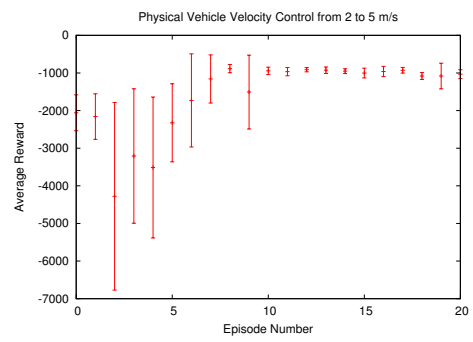
- Delays make domain non-Markov, but k-Markov
- Provide model with previous k actions (Similar to U-Tree [McCallum 1996])
- Trees can learn which delayed actions are relevant
- UCT can plan over augmented state-action histories easily
- Would not be as easy with tabular models or dynamic programming

Using ROS



- TEXPLORE sends **action** messages and receives **state-reward** messages
- Interface node translates **actions** to actuator commands and translates sensor information into **state-reward**
- No need to touch the TEXPLORE code, simply subscribe to and publish the correct messages

Learning on the Autonomous Vehicle



- Learns the task within **2 minutes** of driving time

Conclusion

- TEXPLORE can:
  1. Learn in few **samples**
  2. Act continually in **real-time**
  3. Learn in **continuous** domains
  4. Handle actuator **delays**
- TEXPLORE code has been released as a ROS package: www.ros.org/wiki/rl-texplore-ros-pkg