TEXPLORE: Real-Time Sample-Efficient Reinforcement Learning for Robots

Todd Hester and Peter Stone

Learning Agents Research Group
Department of Computer Science
The University of Texas at Austin

Journal Track: appeared in Machine Learning, 2013
Robots have the potential to solve many problems

We need methods for them to learn and adapt to new situations
Value function RL has string of positive theoretical results [Watkins 1989, Brafman and Tennenholtz 2001]

Could be used for learning and adaptation on robots
Reinforcement Learning

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)
Velocity Control

- **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:** $-10.0 \times$ velocity error (m/s)
Desiderata

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
Desiderata

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
## Common Approaches

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Citation</th>
<th>Sample Efficient</th>
<th>Real Time</th>
<th>Continuous</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-Max</td>
<td>Brafman 2001</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Q-Learning with F.A.</td>
<td>Watkins 1989</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SARSA</td>
<td>Sutton &amp; Barto 1998</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>GPRL</td>
<td>Rummery &amp; Niranjan 1994</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BOSS</td>
<td>Deisenroth &amp; Rasmussen 2011</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Bayesian DP</td>
<td>Asmuth et al 2009</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MBBE</td>
<td>Dearden et al 1999</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MBS</td>
<td>Walsh et al 2009</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Dyna</td>
<td>Sutton 1990</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
# Common Approaches

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Citation</th>
<th>Sample Efficient</th>
<th>Real Time</th>
<th>Continuous</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-Max</td>
<td>Brafman 2001</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Q-Learning</td>
<td>Watkins 1989</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SARSA</td>
<td>Sutton &amp; Barto 1998</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>GPRL</td>
<td>Rummery &amp; Niranjan 1994</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BOSS</td>
<td>Asmuth et al 2009</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Bayesian DP</td>
<td>Strens 2000</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MBBE</td>
<td>Dearden et al 1999</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MBS</td>
<td>Walsh et al 2009</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Dyna</td>
<td>Sutton 1990</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
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 state space into regions with similar dynamics
Random Forest Model [ICDL 2010]

- Average predictions of $m$ different decision trees
- Each tree represents a hypothesis of the true dynamics of the domain
- 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
Random Forest Model [ICDL 2010]

- Average predictions of $m$ different decision trees
- Each tree represents a **hypothesis** of the true dynamics of the domain
- 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 Selection

- Model update can take too long
- Planning can take too long
Real-Time Model Based Architecture (RTMBA)

- Model learning and planning on parallel threads
- Action selection is not restricted by their computation time
- 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
Autonomous Vehicle

- Upgraded to run **autonomously** by adding shift-by-wire, steering, and braking actuators.
- Vehicle runs at 20 Hz.
- Agent **must** provide commands at this frequency.
Uses ROS [Quigley et al 2009]

http://www.ros.org/wiki/rl_msgs
Simulation Experiments

**Exploration Approaches**
- Epsilon-Greedy
- Boltzmann Exploration
- Use merged BOSS-like model
- Use random model each episode

**Sample Efficient Methods**
- BOSS [Asmuth et al 2009]
- Bayesian DP [Strens 2000]
- Gaussian Process RL [Deisenroth & Rasmussen 2011]
## Simulation Experiments

### Continuous Models
- Tabular Models
- Gaussian Process RL [Deisenroth & Rasmussen 2011]
- KWIK linear regression [Strehl & Littman 2007]

### Real-Time Architectures
- Real Time Dynamic Programming [Barto et al 1995]
- Dyna [Sutton 1990]
- Parallel Value Iteration

### Actuator Delays
- Model Based Simulation [Walsh et al 2009]
Challenge 1: Sample Efficiency

Simulated Car Control Between Random Velocities

Average Reward vs. Episode Number for different algorithms:
- TEXPLORE (Greedy)
- Epsilon-Greedy
- Boltzmann
- Variance-Bonus $b=1$
- Variance-Bonus $b=10$
- BayesDP-like
- BOSS-like

Hester and Stone – UT Austin  
TEXPLORE: Real-Time Sample-Efficient RL for Robots
Challenge 1: Sample Efficiency

Simulated Car Control Between Random Velocities

Average Reward vs. Episode Number

- TEXPLORE (Greedy)
- BOSS
- Bayesian DP
- GPRL
- R-Max
- Q-Learning with Tile-Coding
Challenge 2: Real-Time Action Selection

Simulated Car Control Between Random Velocities

Average Reward vs. Episode Number

- RTMBA (TEXPLORE)
- RTDP
- Parallel VI
- Value Iteration
- RT-Dyna
- Q-Learning Tile-Coding

Hester and Stone – UT Austin

TEXPLORE: Real-Time Sample-Efficient RL for Robots
Challenge 3: Modeling Continuous Domains

Model Accuracy on Next State

- Regression Tree Forest
- Regression Tree
- Decision Tree Forest
- Decision Tree
- Tabular
- KWIK Linear Regression
- GP Regression

Average State Error (Euclidean Distance) vs. Number of State-Actions
Challenge 3: Modeling Continuous Domains

Model Accuracy on Next State

Model Accuracy on Reward

Regression Tree Forest
Regression Tree
Decision Tree Forest
Decision Tree
Tabular
KWIK Linear Regression
GP Regression

Hester and Stone – UT Austin
TEXPLORE: Real-Time Sample-Efficient RL for Robots
Challenge 4: Handling Delayed Actions

Simulated Car Control Between Random Velocities

Average Reward vs. Episode Number

- TEXPLORE k=0
- TEXPLORE k=1
- TEXPLORE k=2
- TEXPLORE k=3
- MBS k=1
- MBS k=2
- MBS k=3
- Tabular k=2
On the physical vehicle

But, does it work on the actual vehicle?
On the physical vehicle

Physical Vehicle Velocity Control from 2 to 5 m/s

Yes! It 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