

indigo

Documentation Status

Package Links

- **Code API** (http://docs.ros.org/indigo/api/rl_agent/html)
- FAQ (http://answers.ros.org/questions/scope:all/sort:activity-desc/tags:rl_agent/page:1/)
- Change List (/reinforcement_learning/ChangeList)
- Reviews (/rl_agent/Reviews)

Dependencies (7)**Used by (1)****Jenkins jobs (1)**

Package Summary

✓ Documented

rl_agent is a package containing reinforcement learning (RL) agents.

- Maintainer: Todd Hester <[todd.hester AT gmail DOT com](mailto:todd.hester@gmail.com)>
- Author: Todd Hester
- License: BSD
- Source: git <https://github.com/toddhester/rl-texplore-ros-pkg.git> (<https://github.com/toddhester/rl-texplore-ros-pkg>) (branch: master)

目录

1. Documentation
 1. Running the agent
 1. Example
 2. The General Model-Based Agent
 3. How the RL Agent interacts with the Environment
 1. Using rl_msgs
 2. Calling methods directly
 4. Running the various algorithms
 1. Running Q-Learning
 2. Running Sarsa
 3. Running Dyna
 4. Running R-Max
 5. Running TEXPLORE
 6. Running the general Model-Based agent

This package provides some reinforcement learning (RL) agents.

1. Documentation


Please take a look at the tutorial (/reinforcement_learning/Tutorials/Reinforcement%20Learning%20Tutorial) on how to install, compile, and use this package.

Check out the code at: <https://github.com/toddhester/rl-texplore-ros-pkg> (<https://github.com/toddhester/rl-texplore-ros-pkg>)

This package includes a number of reinforcement learning agents that can be used for learning on robots, or learning with the environments in the accompanying rl_env (/rl_env) package.

The package contains the following agents:

- Q-Learning (Watkins 1989 (/rl-texplore-ros-pkg/rl_references))
- Sarsa (Rummery and Niranjana 1994 (/rl-texplore-ros-pkg/rl_references))
- Dyna (Sutton 1990 (/rl-texplore-ros-pkg/rl_references))
- R-Max (Brafman and Tennenholtz 2001 (/rl-texplore-ros-pkg/rl_references))

-  **TEXPLORE** (<http://www.cs.utexas.edu/~pstone/Papers/bib2html/b2hd-ICDL10-hester.html>) (Hester and Stone 2010 (/rl-texplore-ros-pkg/rl_references))

In addition to these methods, the package contains a general model-based architecture that can be used with any combinations of planners and model learning algorithms. For example, the R-Max implementation is simply the general agent with an R-Max model and Value Iteration for planning and the TEXPLORE agent is the general agent with a random forest model (Breiman 2001 (/rl-texplore-ros-pkg/rl_references)) and UCT (Kocsis and Szepesvari 2006 (/rl-texplore-ros-pkg/rl_references)) for planning.

1.1 Running the agent

The agent can be run with the following command. It should be initialized before starting the environment:

```
roslaunch rl_agent agent --agent type [options]
```

where the agent type is one of the following:

```
qlearner sarsa modelbased rmax texplore dyna savedpolicy
```

There are a number of options to specify particular parameters of the algorithms:

- --seed value (integer seed for random number generator)
- --gamma value (discount factor between 0 and 1)
- --epsilon value (epsilon for epsilon-greedy exploration)
- --alpha value (learning rate alpha)
- --initialvalue value (initial q values)
- --actrate value (action selection rate (Hz))
- --lambda value (lambda for eligibility traces)
- --m value (parameter for R-Max)
- --k value (For Dyna: # of model based updates to do between each real world update)
- --history value (# steps of history to use for planning with delay)
- --filename file (file to load saved policy from for savedpolicy agent)
- --model type (tabular,tree,m5tree)
- --planner type (vi,pi,sweeping,uct,parallel-uct,delayered-uct,delayered-parallel-uct)


- --explore type (unknown,greedy,epsilongreedy,variancenovelty)
- --combo type (average,best,separate)
- --nmodels value (# of models)
- --nstates value (optionally discretize domain into value # of states on each feature)
- --reltrans (learn relative transitions)
- --abstrans (learn absolute transitions)
- --v (For TEXPLORE: coefficient for variance bonus intrinsic rewards)
- --n (For TEXPLORE: coefficient for novelty bonus intrinsic rewards)
- --prints (turn on debug printing of actions/rewards)

1.1.1 Example

For example, to run real-time TEXPLORE using 10 continuous trees, at an action rate of 25 Hz, with a discount factor of 0.99, you would call:

```
roslaunch rl_agent agent --agent texplore --planner parallel-uct --nmodels 10 --model m5tree --actrate 25 --gamma 0.99
```

1.2 The General Model-Based Agent

Included in this package is a general model based agent that can use any model learning or planning method that match the interface defined by the  core.hh (http://www.ros.org/doc/indigo/api/rl_common/html/core_8hh.html) file in the rl_common (/rl_common) package.

The model learning methods that are available include:

- tabular: normal maximum likelihood tabular model
- tree: discrete C4.5 decision trees (Quinlan 1986 (/rl-texplore-ros-pkg/rl_references)) as used in TEXPLORE (Hester and Stone 2010 (/rl-texplore-ros-pkg/rl_references))
- m5tree: continuous M5 regression trees (Quinlan 1992 (/rl-texplore-ros-pkg/rl_references))

With any of these types of models, multiple models can be combined together using the --nmodels option. For example, a random forest with 10 trees can be created with the options:

```
--model tree --nmodels 10
```

There are also a number of planning methods available:

- Value Iteration (Sutton and Barto 1998 ([/rl-texplore-ros-pkg/rl_references](#)))
- Policy Iteration (Sutton and Barto 1998 ([/rl-texplore-ros-pkg/rl_references](#)))
- Prioritized Sweeping (Moore and Atkeson 1993 ([/rl-texplore-ros-pkg/rl_references](#)))
- UCT: a sampling based planning method (Kocsis and Szepesvari 2006 ([/rl-texplore-ros-pkg/rl_references](#)))
- Parallel UCT: planning and model learning occur in parallel with action selection so agent can select actions in real time (Kocsis and Szepesvari 2006 ([/rl-texplore-ros-pkg/rl_references](#)), Hester et al 2012 ([/rl-texplore-ros-pkg/rl_references](#)))
- Delayed UCT: provides model with last n actions (from --history option) to deal with delayed domains (McCallum 1996 ([/rl-texplore-ros-pkg/rl_references](#)), Kocsis and Szepesvari 2006 ([/rl-texplore-ros-pkg/rl_references](#)))
- Delayed Parallel UCT: combines the delayed and parallel versions (McCallum 1996 ([/rl-texplore-ros-pkg/rl_references](#)), Kocsis and Szepesvari 2006 ([/rl-texplore-ros-pkg/rl_references](#)), Hester et al 2012 ([/rl-texplore-ros-pkg/rl_references](#)))

Any of these model learning methods can be combined with any of the planners. It is also easy to write new model learning and planning methods that match the interface defined in `rl_common` ([/rl_common](#)) and use those as well. In addition, there are multiple ways of performing exploration:


- epsilon-greedy: take a random action epsilon of the time, act greedily otherwise
- greedy: always be greedy
- unknown: provide r-max like bonuses to unknown state-action pairs
- variancenovelty: explore using variance and novelty exploration bonuses as in `TEXPLORE-VANIR` (Hester and Stone 2012 ([/rl-texplore-ros-pkg/rl_references](#)))



1.3 How the RL Agent interacts with the Environment

The RL agent can interact with the environment in two ways: it can use the ROS messages defined in the `rl_msgs` ([/rl_msgs](#)) package, or another method can call the agent and environment methods directly, as done in the `rl_experiment` ([/rl_experiment](#)) package.



1.3.1 Using `rl_msgs`

The `rl_msgs` package defines a set of ROS messages for the agent and environment to communicate. These are similar to the messages used in `RL-Glue` (Tanner and White 2009 ([/rl-texplore-ros-pkg/rl_references](#))), but simplified and defined in the ROS message format. The environment publishes three types of messages for the agent:

-  `rl_msgs/REnvDescription` (http://www.ros.org/doc/indigo/api/rl_msgs/html/msg/REnvDescription.html): this message describes the environment, number of actions, number of features, if its episodic, etc.


-  `rl_msgs/REnvSeedExperience` (http://www.ros.org/doc/indigo/api/rl_msgs/html/msg/REnvSeedExperience.html): this message provides an experience seed for the agent to use for learning.
-  `rl_msgs/RLStateReward` (http://www.ros.org/doc/indigo/api/rl_msgs/html/msg/RLStateReward.html): this is a message from the environment with the agent's new state and reward received on this time step.

The environment subscribes to one type of message from the agent:

-  `rl_msgs/RLAction` (http://www.ros.org/doc/indigo/api/rl_msgs/html/msg/RLAction.html): this message sends the environment the action that the agent has selected.
-  `rl_msgs/RLExperimentInfo` (http://www.ros.org/doc/indigo/api/rl_msgs/html/msg/RLExperimentInfo.html): this message provides information on the results of the latest episode of the experiment.

When the environment is created, it sends an `REnvDescription` message to the agent. Then it will send any experience seeds for the agent in a series of `REnvSeedExperience` messages. Then it will send the agent an `RLStateReward` message with the agent's initial state in the domain. It should then receive an `RLAction` message, which it can apply to the domain and send a new `RLStateReward` message. When the episode has ended, the environment will receive an `RLExperimentInfo` message from the agent, and it will reset the domain and send the agent a new `RLStateReward` message with its initial state in the new episode.


1.3.2 Calling methods directly

Experiments can also be run by calling the agent methods directly (as done in the `rl_experiment (/rl_experiment)` package). The methods that all Agents must implement are defined in the Agent interface in the `rl_common (/rl_common)` package ( API (http://www.ros.org/doc/indigo/api/rl_common/html/classAgent.html)). Seeds can be given to the method by calling the `seedExp` method. The agent can be queried for an action after getting a new state and reward by calling `next_action(reward, state)`.

1.4 Running the various algorithms

In this section, I provide directions on running each of the various algorithms available in the package, as well as what options each of the algorithms have. The package contains 6 algorithms:

- Q-Learning (Watkins 1989 ([/rl-texplore-ros-pkg/rl_references](#)))
- Sarsa (Rummery and Niranjan 1994 ([/rl-texplore-ros-pkg/rl_references](#)))
- Dyna (Sutton 1990 ([/rl-texplore-ros-pkg/rl_references](#)))
- R-Max (Brafman and Tennenholtz 2001 ([/rl-texplore-ros-pkg/rl_references](#)))

-  [TEXPLORE](http://www.cs.utexas.edu/~pstone/Papers/bib2html/b2hd-ICDL10-hester.html) (<http://www.cs.utexas.edu/~pstone/Papers/bib2html/b2hd-ICDL10-hester.html>) (Hester and Stone 2010 ([/rl-texplore-ros-pkg/rl_references](#)))
- General Model-Based algorithm

1.4.1 Running Q-Learning

To run the basic Q-Learning (Watkins 1989 ([/rl-texplore-ros-pkg/rl_references](#))) agent, type the following:

```
roslaunch rl_agent agent --agent qlearner
```

By default, Q-Learning will be run with greedy exploration, a learning rate alpha of 0.3, and initial Q-values of 0.0.

The following options are available for the Q-Learning agent:

- --seed value (integer seed for random number generator)
- --gamma value (discount factor between 0 and 1)
- --epsilon value (epsilon for epsilon-greedy exploration)
- --alpha value (learning rate alpha)
- --initialvalue value (initial q values)
- --filename file (file to save a policy from the agent)
- --explore type (greedy,epsilongreedy)
- --nstates value (optionally discretize domain into value # of states on each feature)
- --prints (turn on debug printing of actions/rewards)

1.4.2 Running Sarsa

To run the basic Sarsa (Rummery and Niranjan 1994 ([/rl-texplore-ros-pkg/rl_references](#))) agent, type the following:

```
roslaunch rl_agent agent --agent sarsa
```

By default, Sarsa will be run with greedy exploration, a learning rate alpha of 0.3, initial action-values of 0.0, and lambda set to 0.1.

The following options are available for the Sarsa agent:

- --seed value (integer seed for random number generator)
- --gamma value (discount factor between 0 and 1)

- --epsilon value (epsilon for epsilon-greedy exploration)
- --alpha value (learning rate alpha)
- --initialvalue value (initial q values)
- --lambda value (lambda for eligibility traces)
- --filename file (file to save a policy from the agent)
- --explore type (greedy,epsilongreedy)
- --nstates value (optionally discreteize domain into value # of states on each feature)
- --prints (turn on debug printing of actions/rewards)

1.4.3 Running Dyna

To run the basic Dyna (Sutton 1990 (/rl-texplore-ros-pkg/rl_references)) agent, type the following:

```
roslaunch rl_agent agent --agent dyna
```

By default, Dyna will be run with greedy exploration, a learning rate alpha of 0.3, initial action-values of 0.0, and k set to 1000.

The following options are available for the Dyna agent:

- --seed value (integer seed for random number generator)
- --gamma value (discount factor between 0 and 1)
- --epsilon value (epsilon for epsilon-greedy exploration)
- --alpha value (learning rate alpha)
- --initialvalue value (initial q values)
- --k value (# of model based updates to do between each real world update)
- --filename file (file to save a policy from the agent)
- --explore type (greedy,epsilongreedy)
- --nstates value (optionally discreteize domain into value # of states on each feature)
- --prints (turn on debug printing of actions/rewards)

1.4.4 Running R-Max

To run the basic R-Max (Brafman and Tenenbholz 2001 (/rl-texplore-ros-pkg/rl_references)) agent, type the following:

```
roslaunch rl_agent agent --agent rmax
```


R-Max uses a tabular model and gives exploration bonuses to any state-actions with fewer than M visits. By default, M is set to 5, and R-Max uses value iteration for planning.

The following options are available for the R-Max agent:

- --seed value (integer seed for random number generator)
- --gamma value (discount factor between 0 and 1)
- --actrate value (action selection rate (Hz) if using UCT for planning)
- --m value (how many visits are required for a state-action to become known)
- --filename file (file to save a policy from the agent)
- --planner type (vi,pi,sweeping,uct,parallel-uct,delayed-uct,delayed-parallel-uct)
- --nstates value (optionally discretize domain into value # of states on each feature)
- --prints (turn on debug printing of actions/rewards)

1.4.5 Running TEXPLORE

To run the basic TEXPLORE or TEXPLORE-VANIR (Hester and Stone 2010, Hester and Stone 2012, Hester et al 2012 ([/rl-texplore-ros-pkg/rl_references](#))) agent, type the following:

```
roslaunch rl_agent agent --agent texplore
```

TEXPLORE plans greedily with respect to the average of a number of decision tree models of the domain. By default, TEXPLORE uses nmodels = 5, C 4.5 discrete decision trees, and plans using the RTMBA real-time architecture (Hester et al 2012 ([/rl-texplore-ros-pkg/rl_references](#))) with an action rate of 10 Hz.

For continuous domains, TEXPLORE can use M5 regression trees instead:

```
--model m5tree
```

To run TEXPLORE with Variance and Novelty Rewards (TEXPLORE-VANIR) (Hester and Stone 2012 ([/rl-texplore-ros-pkg/rl_references](#))), set the coefficients for the variance and novelty explorations:

```
--n 5  
--v 5
```

For domains with possible state and actuator delays, enable TEXPLORE to learn models from the previous k actions:

```
--history 5
```

The following options are available for the TEXPLORE agent:

- --seed value (integer seed for random number generator)
- --gamma value (discount factor between 0 and 1)
- --actrate value (action selection rate (Hz))
- --lambda value (lambda for eligibility traces)
- --history value (# steps of history to use for planning with delay)
- --filename file (file to save a policy from the agent)
- --model type (tabular,tree,m5tree)
- --planner type (vi,pi,sweeping,uct,parallel-uct,delayed-uct,delayed-parallel-uct)
- --explore type (unknown,greedy,epsilongreedy,variancenovelty)
- --combo type (average,best,separate)
- --nmodels value (# of models)
- --nstates value (optionally discretize domain into value # of states on each feature)
- --reltrans (learn relative transitions)
- --abstrans (learn absolute transitions)
- --v (coefficient for variance bonus intrinsic rewards)
- --n (coefficient for novelty bonus intrinsic rewards)
- --prints (turn on debug printing of actions/rewards)

1.4.6 Running the general Model-Based agent

There is also an option to run a general model-based agent, using any combination of models, planners, and exploration that you wish. To run it, type the following:

```
roslaunch rl_agent agent --agent modelbased
```

The following options are available for the model-based agent:

- --seed value (integer seed for random number generator)
- --gamma value (discount factor between 0 and 1)
- --epsilon value (epsilon for epsilon-greedy exploration)

- --actrate value (action selection rate (Hz))
- --lambda value (lambda for eligibility traces)
- --m value (parameter for R-Max type exploration)
- --history value (# steps of history to use for planning with delay)
- --filename file (file to save a policy from the agent)
- --model type (tabular,tree,m5tree)
- --planner type (vi,pi,sweeping,uct,parallel-uct,delayed-uct,delayed-parallel-uct)
- --explore type (unknown,greedy,epsilongreedy,variancenovelty)
- --combo type (average,best,separate)
- --nmodels value (# of models)
- --nstates value (optionally discretize domain into value # of states on each feature)
- --reltrans (learn relative transitions)
- --abstrans (learn absolute transitions)
- --v (coefficient for variance bonus intrinsic rewards)
- --n (coefficient for novelty bonus intrinsic rewards)
- --prints (turn on debug printing of actions/rewards)

References (/rl-texplore-ros-pkg/rl_references)

Except where otherwise noted, the ROS wiki is licensed under the

Creative Commons Attribution 3.0 (<http://creativecommons.org/licenses/by/3.0/>) | Find us on
Google+ (<https://plus.google.com/113789706402978299308>)

Wiki: rl_agent (2015-09-06 01:10:57由ToddHester (/ToddHester)编辑)

Brought to you by:  Open Source Robotics Foundation

(<http://www.osrfoundation.org>)