GRL: a generic C++ reinforcement learning library

Wouter Caarls <mailto:wouter@caarls.org>

October 5th, 2015

1 Introduction

GRL is a C++ reinforcement learning library that aims to easily allow evaluating different algorithms through a declarative configuration interface.

2 Directory structure

```
.
|-- base                      Base library
|   |-- include               Header files
|   `-- src                   Source files
|       |-- agents            Agents (fixed, black box, td)
|       |-- discretizers      Action discretizers
|       |-- environments      Environments (pendulum, cart-pole)
|       |-- experiments       Experiments (online, batch)
|       |-- policies          Control policies (PID, Q-based)
|       |-- predictors        Value function predictors (SARSA, AC)
|       |-- projectors        State projectors (tile coding, fourier)
|       |-- representations   Representations (linear, ann)
|       |-- samplers          Action samplers (greedy, e-greedy)
|       |-- solvers           MDP solvers (VI, rollout-based)
|       |-- traces            Elibility traces (accumulating, replacing)
|       `-- visualizations    Visualizations (value function, policy)
|-- addons                    Optional modules
|   |-- cma                   CMA-ES black-box optimizer
|   |-- gl                    OpenGL-based visualizations
|   |-- glut                  GLUT-based visualizer
|   |-- llr                   Locally linear regression representation
|   |-- lqr                   Linear Quadratic Regulator solver
|   |-- matlab                Matlab interoperability
|   |-- muscod                Muscod interoperability
|   |-- odesim                Open Dynamics Engine environment
|   |-- rbdl                  Rigid Body Dynamics Library dynamics
|   `-- ros                   ROS interoperability
|-- bin                       Python binaries (configurator)
|-- externals                 Imported external library code
|-- cfg                       Sample configurations
|-- share                     Misc files
|   `-- taskmaster            Taskmaster parameter study example
|-- tests                     Unit tests
|-- CMakeLists.txt            CMake instructions to build everything
`-- grl.cmake                 CMake helper functions
```

3 Prerequisites

GRL requires some libraries in order to compile. Which ones exactly depends on which agents and environments you would like to build, but the full list is

- Git
- GCC (including g++)
- Eigen
- GLUT
- ZLIB
- QT4 (including the OpenGL bindings)
- TinyXML
- MuParser
- ODE, the Open Dynamics Engine
- Python (including Tkinter and the yaml reader)
- Lua

On Ubuntu 16.04, these may be installed with the following command:

```
wcaarls@vbox:~$ git cmake g++ libeigen3-dev \
libgl1-mesa-dev freeglut3-dev libz-dev libqt4-opengl-dev \
libtinyxml-dev libmuparser-dev libode-dev python-yaml python-tk \
liblua5.1-dev
```

4 Building

GRL may be built with or without ROS's catkin. When building with, simply merge `grl.rosinstall` with your catkin workspace

```
wcaarls@vbox:~$ mkdir indigo_ws
wcaarls@vbox:~$ cd indigo_ws
wcaarls@vbox:~/indigo_ws$ rosws init src /opt/ros/indigo
wcaarls@vbox:~/indigo_ws$ cd src
wcaarls@vbox:~/indigo_ws/src$ rosws merge /path/to/grl.rosinstall
wcaarls@vbox:~/indigo_ws/src$ rosws up
wcaarls@vbox:~/indigo_ws/src$ cd ..
wcaarls@vbox:~/indigo_ws$ catkin_make
```

Otherwise, follow the standard CMake steps of (in the `grl` directory)

```
wcaarls@vbox:~/src/grl$ mkdir build
wcaarls@vbox:~/src/grl$ cd build
wcaarls@vbox:~/src/grl/build$ cmake ..
-- The C compiler identification is GNU 4.8.2
...
wcaarls@vbox:~/src/grl/build$ make
Scanning dependencies of target yaml-cpp
...
```

5 Running

The most important executables in grl are the deployer (`grld`) and configurator (`grlc`). The configurator allows you to generate configuration files easily. To see an example, run

```
wcaarls@vbox:~/src/grl/bin$ ./grlc ../cfg/pendulum/sarsa_tc.yaml
```

More information on the configurator can be found in Section 8. Once you have configured your experiment, you can either run it directly from the configurator, or save it and run it using the deployer. For example:

```
wcaarls@vbox:~/src/grl/build$ ./grld ../cfg/pendulum/sarsa_tc.yaml
```

6 Build environment

The whole `grl` system is built as a single package, with the exception of `mprl_msgs`. This is done to facilitate building inside and outside catkin. There is one `CMakeLists.txt` that is used in both cases. The ROS interoperability is selectively built based on whether `cmake` was invoked by `catkin_make` or not.

Modules are built by calling their respective `build.cmake` scripts, which is done by `grl_build_library`. The include directory is set automatically, as is an `SRC` variable pointing to the library's source directory.

The build system has a simplistic dependency management scheme through `grl_link_libraries`. This calls the `link.cmake` files of the libraries on which the current library depends. Typically they will add some `target_link_libraries` and add upstream dependencies. `grl_link_libraries` also automatically adds the upstream library's include directory.

7 Class structure

Most classes in grl derive from `Configurable`, a base class that standardizes configuration such that the object hierarchy may be constructed declaratively in a configuration file. Directly beneath `Configurable` are the abstract base classes defining the operation of various parts of the reinforcement learning environment, being:

`Agent` RL-GLUE[1] style agent interface, receiving observations in an episodic manner and returning actions.

`Discretizer` Provides a list of discrete points spanning a continuous space.

`Environment` RL-GLUE style environment interface, receiving actions and returning observations.

`Experiment` Top-level interface, which typically calls the agent and environment in the correct manner, but may in general implement any experiment.

`Optimizer` Black-box optimization of control policies, suggesting policies and acting on their cumulative reward.

`Policy` Basic control policy that implements the state-action mapping.

`Predictor` Basic reinforcement learning interface that uses transitions to predict a value function or model.

`Projector` Projects an observation onto a feature vector, represented as a `Projection`.

`Representation` Basic supervised learning interface that uses samples to approximate a function. As such, it generally supports reading, writing and updating of any vector-to-vector mapping.

`Sampler` (Stochastically) chooses an item from a vector of (generally unnormalized) values.

`Trace` Stores a trace of projections with associated eligibilities that can be iterated over.

[1] `http://http://glue.rl-community.org`

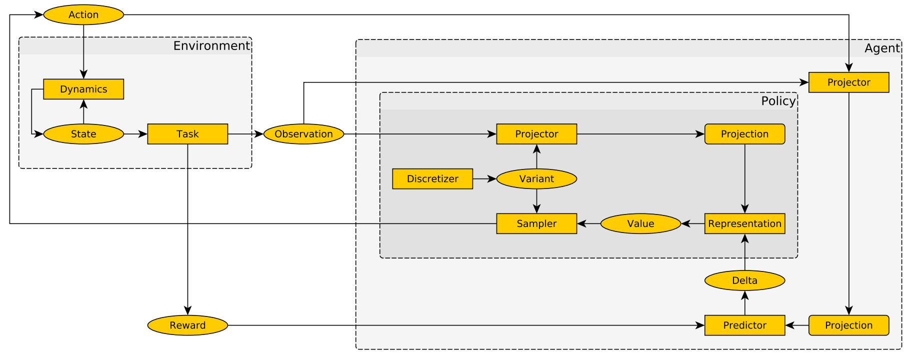


Figure 1: Information flow diagram for regular TD control. Rectangles (and dashed rectangles) are `Configurable` objects, while the others are the data passed between them.

`Visualization` Draws on the screen to visualize some aspect of the learning process.

`Visualizer` Keeps track of visualizations and provides the interface to the graphics subsystem.

Each abstract base class is generally implemented in various concrete classes, with or without additional hierarchy. A list can be requested by running

```
wcaarls@vbox:~/src/grl/bin$ ./grlq
```

and is also available in the appendices of this document.

A typical example of the information flow between the various classes can be seen in Figure 1, which depicts the standard TD control setting.

7.1 Configuration

Each `Configurable` subclass must define its type and a short description using the `TYPEINFO` macro:

```
class OnlineLearningExperiment : public Experiment
{
  public:
    TYPEINFO("experiment/online_learning", "Interactive learning experiment")

  /* ... */
};
```

This textual description of the type is used to facilitate user configuration by limiting the selection of parameter values, as well as enforcing the type hierarchy.

In general, the textual description should follow the C++ class hierarchy, but this is not obligatory.

The basic Configurable interface has three important functions:

7.1.1 request

```
virtual void request(ConfigurationRequest *config);
```

request is called by the configurator to find out which parameters the object requires to be set, and which parameters it exports for other objects to use. To do this, it should extend the given ConfigurationRequest by pushing configuration request parameters (CRPs). A basic CRP has the following signature:

```
CRP(string name, string desc, TYPE value)
```

where TYPE is one of int, double, Vector, or string. For example:

```
config->push_back(CRP("steps", "Number of steps per learning run", steps_));
config->push_back(CRP("output", "Output base filename", output_));
```

The value argument is used both to determine the type of the parameter and the default value suggested by the configurator. request may also be called while the program is running, in which case it is expected to return the current value of all parameters.

To use other Configurable objects as parameters, use

```
CRP(string name, string type, string desc, Configurable *value)
```

The extra type field restricts which Configurable objects may be used to configure this parameter. Only objects whose TYPEINFO starts with the given type are eligible. For example:

```
config->push_back(CRP("policy", "policy/parameterized",

                      "Control policy prototype", policy_));
```

restricts the "policy" parameter to classes derived from ParameterizedPolicy. Note that this extra type hierarchy is related to, but not derived from the actual class hierarchy. Care must therefore be taken in the correct usage of TYPEINFO.

Some parameters are not requested, but rather *provided* by an object. In that case. These have the following signature:

```
CRP(string name, string type, string desc, CRP::Provided)
```

Examples of provided parameters are the number of observation dimensions (provided by Tasks) or the current system state (provided by some Environments).

7.1.2 `configure`

```
virtual void configure(Configuration *config);
```

`configure` is called after all parameters (including other `Configurable` objects) have been initialized. The parameter values may be accessed using mapping syntax (`config["parameter"]`). Note that `Configurable` objects are passed as void pointers and must still be cast to their actual class:

```
steps_ = config["steps"];
output_ = config["output"].str();
policy_ = (ParameterizedPolicy*)config["policy"].ptr();
```

Note the use of `.str()` and `.ptr()` for strings and objects, respectively. Provided parameters should be written to the configuration instead of read, like so:

```
config.set("state", state_);
```

7.1.3 `reconfigure`

```
virtual void reconfigure(const Configuration *config);
```

Some parameters may be defined as reconfigurable by appending `CRP::Online` to the respective `CRP` signature. In the case of a reconfiguration, `reconfigure` will be called with the new values of those parameters in `config`. `reconfigure` may also be used for general messaging, equivalent to RL-GLUE's `message` calls. In that case, it is often helpful to reconfigure all objects in the object hierarchy, which can be done using

```
void Configurable::walk(const Configuration &config);
```

Examples are resetting the hierarchy for a new run (`config["action"] = "reset"`) or saving the current state of all memories (`config["action"] = "save"`). In the latter case, `Configurable::path()` may be used to determine an object's location in the object hierarchy.

7.2 Roles

While using the configurator, the user often has to select previously defined objects as the value of certain parameters. If all such previously defined objects are presented as possibilities, the list would quickly grow very large. To make setting these parameters easier, a class may have various *roles* while providing the same interface. In that case, only previously defined objects with a role that starts with the requested role are valid choices.

An example is a `Representation`, which may represent a state-value function, action-value function, control policy or model. Each has a different number of inputs and outputs, and chosing the wrong representation will result in mismatches. An object requesting a `Representation` may therefore request a certain role. For example:

```
config->push_back(CRP("representation", "representation.value/action",

                      "Q-value representation", representation_));
```

requests any representation that represents action-values. A newly defined `representation` will do, of course, but from the previously defined ones only the ones with the right role are eligible.

The same strategy is used for basic types, for example:

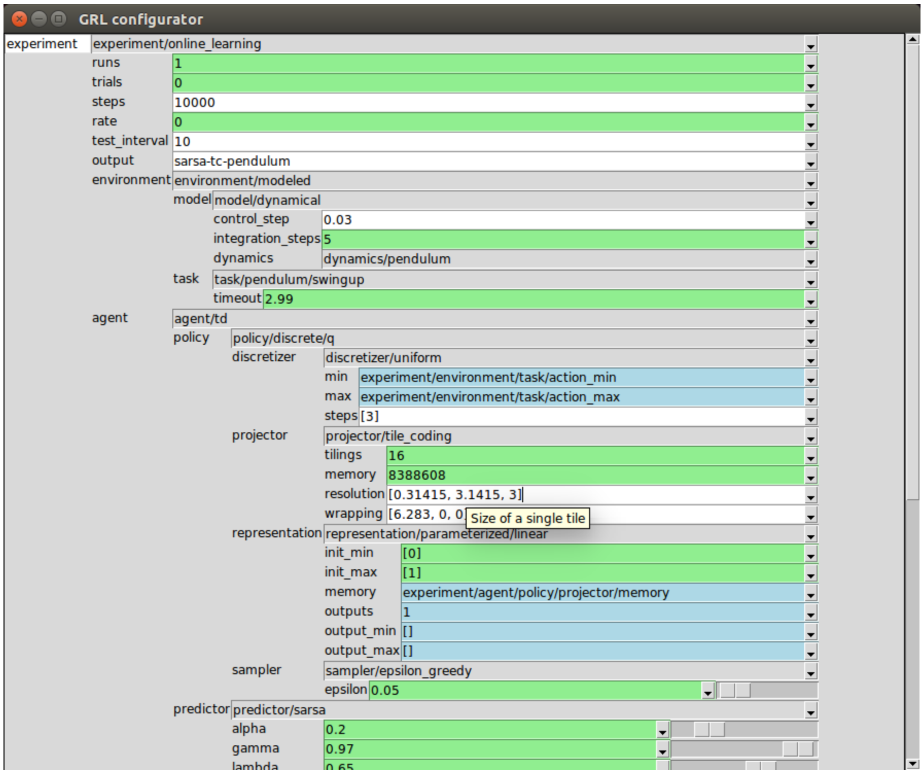
```
config->push_back(CRP("outputs", "int.action_dims",
                      "Number of outputs", outputs_, CRP::System));
```

make sure the only suggested previously defined values for the `"outputs"` parameter are ones with the `"action_dims"` role. As an added convenience, if the parameter is defined as a *system parameter* (`CRP::System`), meaning that the choice is not free but rather defined by the structure of the configuration, and only a single value was previously defined, that value is automatically used.

The role that needs to be requested may depend on the role of the requesting object itself. In that case, the following signature for `request` should be used:

```
virtual void request(const std::string &role, ConfigurationRequest *config);
```

8 Configurator



```
experiment  experiment/online_learning
    runs  1
    trials  0
    steps  10000
    rate  0
    test_interval  10
    output  sarsa-tc-pendulum
    environment  environment/modeled
        model  model/dynamical
            control_step  0.03
            integration_steps  5
            dynamics  dynamics/pendulum
        task  task/pendulum/swingup
            timeout  2.99
    agent  agent/td
        policy  policy/discrete/q
            discretizer  discretizer/uniform
                min  experiment/environment/task/action_min
                max  experiment/environment/task/action_max
                steps  [3]
            projector  projector/tile_coding
                tilings  16
                memory  8388608
                resolution  [0.31415, 3.1415, 3]
                wrapping  [6.283, 0, 0
            representation  representation/parameterized/linear
                init_min  [0]
                init_max  [1]
                memory  experiment/agent/policy/projector/memory
                outputs  1
                output_min  []
                output_max  []
            sampler  sampler/epsilon_greedy
                epsilon  0.05
        predictor  predictor/sarsa
            alpha  0.2
            gamma  0.97
            lambda  0.65
```

Figure 2: Python configurator user interface

9 Matlab interface

If Matlab is installed (and can be found on the path), a MEX interfaces for the agents and environments is built. If you want to use these, make sure that you're building with a compatible compiler, both by setting the `CC` and `CXX` variables in your call to `cmake` and by correctly configuring `mex`.

9.1 Environments

To initialize an environment, call

```
>> spec = grl_env('cfg/matlab/pendulum_swingup.yaml');
```

Where the argument specifies a configuration file that has a top-level 'environment' tag. `spec` gives some information about the environment, such as number of dimensions, minimum and maximum values, etc. Next, retrieve the first observation of an episode with

```
>> o = grl_env('start');
```

where o is the observation from the environment. All following steps should be called using

```
>> [o, r, t, d] = grl_env('step', a);
```

where a is the action suggested by the agent, r is the reward given by the environment, t signals termination of the episode and txtd is the length of the step. If t is 2, the episode ended in an absorbing state. When all episodes are done, exit cleanly with

```
>> grl_env('fini');
```

9.2 Agents

To initialize the agent, use

```
>> grl_agent('init', 'cfg/matlab/sarsa.yaml');
```

Where the argument specifies a configuration file that has a top-level 'agent' tag. Next, give the first observation of an episode with

```
>> a = grl_agent('start', o);
```

where o is the observation from the environment and a is the action suggested by the agent. All following steps should be called using

```
>> a = grl_agent('step', d, r, o);
```

where r is the reward given by the environment and txtd is the length of the step. To signal the end of an episode (absorbing state), use

```
>> a = grl_agent('end', d, r);
```

To end an episode without an absorbing state, simply start a new one. To exit cleanly after all epsiodes are finished (which also allows you to reinitialize the agent with different options), call

```
>> grl_agent('fini');
```

A Agents

A.1 agent/black_box

Agent that learns from the cumulative reward of complete rollouts

episodes	int	Number of episodes to evaluate policy
optimizer	optimizer	Policy optimizer

A.2 agent/communicator

Communicator agent which connects GRL to a remote agent

communicator	communicator	Comunicator which exchanges messages with an actual/virtual env
observation_dims	int.observation_dims	Number of observation dimensions
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit of action
action_max	vector.action_max	Upper limit of action
test	int.test	Selection of a learning/testing agent role

A.3 agent/delayed_td

Agent that learns from observed state transitions assuming non-integer values of control delay

policy	mapping/policy	Control policy
predictor	predictor	Value function predictor
control_delay	double	Relative control delay: 0 (no delay) - 1 (one timestep delay)

A.4 agent/dyna

Agent that learns from both observed and predicted state transitions

planning_steps	int	Number of planning steps per control step
planning_horizon	int	Planning episode length
threads	int	Threads used for planning (0 = synchronous planning. ¿0 requires re
policy	mapping/policy	Control policy
predictor	predictor	Value function predictor
model	observation_model	Observation model used for planning
model_predictor	predictor/model	Model predictor
model_agent	agent	Agent used for planning episodes

Provided parameters

state	state	Current observed state of planning

A.5 agent/filtering

Agent that filters incoming observations and outgoing actions

observation_idx	vector	Index vector for downstream observation (-1=pad)
action_idx	vector	Index vector for upstream action (-1=pad)
action_dims	int	Number of downstream action dimensions
agent	agent	Downstream agent

A.6 agent/fixed

Fixed-policy agent

policy	mapping/policy	Control policy

A.7 agent/leo/fixed

Leo fixed agent

policy	mapping/policy	Control policy
pub_transition_type	signal/vector	Publisher of the transition type

A.8 agent/leo/sma

State-machine agent for Leo

agent_prepare	agent	Prepare agent
agent_standup	agent	Safe standup agent
agent_starter	agent	Starting agent
agent_main	agent	Main agent
upright_trigger	trigger	Trigger which finishes stand-up phase and triggers preparation agent
fc_trigger	trigger	Trigger which checks for foot contact to ensure that robot is prepared to wal
starter_trigger	trigger	Trigger which initiates a preprogrammed walking at the beginning
sub_ic_signal	signal/vector	Subscriber to the contact signal

A.9 agent/leo/sym_wrapper

Leo agent that symmetrically wraps angles and controls

agent	agent	Target agent with reduced state-action space due to symmetry
sub_ic_signal	signal/vector	Publisher of the initialization and contact signal

A.10 agent/leo/td

Leo agent that learns from observed state transitions

policy	mapping/policy	Control policy
predictor	predictor	Value function predictor
pub_transition_type	signal/vector	Publisher of the transition type

A.11 agent/leo_preprogrammed

Leo preprogrammed agent

rand_gen	random_generator	Random generator for action pertubation
epsilon	double	Exploration rate
output_min	vector.action_min	Lower limit on outputs
output_max	vector.action_max	Upper limit on outputs

A.12 agent/master/exclusive

Master agent that selects one sub-agent to execute

gamma	double	Discount rate
control_step	double.control_step	Characteristic step time on which gamma is defined
predictor	predictor	Optional (model) predictor
agent1	agent/sub	First subagent
agent2	agent/sub	Second subagent

A.13 agent/master/predicated

Master agent in which execution is predicated on preceding agent confidence

gamma	double	Discount rate
control_step	double.control_step	Characteristic step time on which gamma is defined
predictor	predictor	Optional (model) predictor
agent1	agent/sub	First subagent
agent2	agent/sub	Second subagent

A.14 agent/master/random

Master agent that chooses sub-agents randomly

gamma	double	Discount rate
control_step	double.control_step	Characteristic step time on which gamma is defined
predictor	predictor	Optional (model) predictor
agent1	agent/sub	First subagent
agent2	agent/sub	Second subagent

A.15 agent/master/sequential

Master agent that executes sub-agents sequentially

predictor	predictor	Optional (model) predictor
agent1	agent	First subagent, providing the suggested action
agent2	agent	Second subagent, providing the final action
exporter	exporter	Optional exporter for transition log (supports time, state, observation, action, reward,

A.16 agent/master/sequential/additive

Additive master agent that executes sub-agents sequentially and adds their outputs

predictor	predictor	Optional (model) predictor
agent1	agent	First subagent, providing the suggested action
agent2	agent	Second subagent, providing the final action
exporter	exporter	Optional exporter for transition log (supports time, state, observation, act
output_min	vector.action_min	Lower limit on outputs
output_max	vector.action_max	Upper limit on outputs

A.17 agent/solver

Agent that successively solves learned models of the environment

interval	int	Episodes between successive solutions (0=asynchronous)
policy	mapping/policy	Control policy
predictor	predictor	Optional (model) predictor
solver	solver	Model-based solver

A.18 agent/sub/compartmentalized

Sub agent that is valid in a fixed state-space region

min	vector.observation_min	Minimum of compartment bounding box
max	vector.observation_max	Maximum of compartment bounding box
agent	agent	Sub agent

A.19 agent/sub/filtering

Subagent that filters incoming observations and outgoing actions

observation_idx	vector	Index vector for downstream observation (-1=pad)
action_idx	vector	Index vector for upstream action (-1=pad)
action_dims	int	Number of downstream action dimensions
agent	agent/sub	Downstream subagent

A.20 agent/sub/voluntary

Sub agent that has confidence as part of the action

dim	int	Action dimension that indicates confidence
agent	agent	Sub agent

A.21 agent/td

Agent that learns from observed state transitions

policy	mapping/policy	Control policy
predictor	predictor	Value function predictor

B Behaviors

B.1 behavior/leo_squat_sym

Leo squatting behavior with symmetrical switchers of observations

B.2 behavior/leo_walk

Leo walking behavior without symmetrical switchers of observations

B.3 behavior/leo_walk_sym

Leo walking behavior with symmetrical switchers of observations

C Communicators

C.1 communicator/zeromq/pub_sub

Zeromq class to establish a link by sending messages asynchronously (publisher/subscriber)

role	string	Role of the zeromq (Pub/Sub, Request/Reply)
sync	string	Syncronization ip address
pub	string	Publisher address
sub	string	subscriber address

C.2 communicator/zeromq/request_reply

Zeromq class to establish a link by sending messages synchronously (request/reply)

role	string	Role of the zeromq (Pub/Sub, Request/Reply)
sync	string	Syncronization ip address
addr	string	Address

D Converters

D.1 converter/state_action_converter

Configurable which is capable of remapping states and actions

state_in	string	Comma-separated list of state elements in the input vector
state_out	string	Comma-separated list of state elements in the output vector
action_in	string	Comma-separated list of action elements observed in the input vector
action_out	string	Comma-separated list of action elements provided in the output vector

E Discretizers

E.1 discretizer/peaked

Peaked discretizer, with more resolution around center

min	vector	Lower limit
max	vector	Upper limit
steps	vector	Discretization steps per dimension
peaking	vector	Extra resolution factor around center (offset by 1/factor at edges)

E.2 discretizer/policy

Returns the action suggested by a policy

policy	mapping/policy	Policy whose action to return

E.3 discretizer/split

Compound discretizer

identify	int	Identify active discretizer before (-1) or after (1) value
discretizer1	discretizer.	First discretizer
discretizer2	discretizer.	Second discretizer

E.4 discretizer/uniform

Uniform discretizer

min	vector	Lower limit
max	vector	Upper limit
steps	vector	Discretization steps per dimension

F Dynamics

F.1 dynamics/acrobot

Acrobot dynamics

F.2 dynamics/cart_double_pole

Cart-double-pole dynamics from Zhong and Rock

F.3 dynamics/cart_pole

Cart-pole dynamics from Barto et al.

F.4 dynamics/flyer2d

2D flyer dynamics

F.5 dynamics/mountain

Mountain world dynamics

mass	double	Car mass
gravity	double	Gravitational acceleration
friction	double	Coefficient of viscous friction between car and ground
stiffness	double	Spring constant of walls
map	mapping/puddle	Height map

F.6 dynamics/pendulum

Pendulum dynamics based on the DCSC MOPS

F.7 dynamics/rbdl

RBDL rigid body dynamics

file	string	RBDL Lua model file
options	string	Lua string to execute when loading model
points	string	Points
auxiliary	string	Model mass(mm), Center of mass (com), Center of mass velocity (comv), Angular momen

F.8 dynamics/swimmer

Coulom's swimmer dynamics

segments	double.swimmer/segments	Number of swimmer segments

F.9 dynamics/tlm

Two-link manipulator dynamics

G Environments

G.1 environment/communicator

Communicator environment which interects with a real environment by sending and receiving messages

converter	converter	Convert states and actions if needed
communicator	communicator	Comunicator which exchanges messages with an actual/virtual environn
target_obs_dims	int	Observation dimension of a target
target_action_dims	int	Action dimension of a target
benchmark_delays	int	Observation dimension of a target

G.2 environment/leo2

LEO/2 environment

port	string	Device ID of FTDI usb-to-serial converter
bps	int	Bit rate

Provided parameters

state	state	Current state of the robot

G.3 environment/leo_squat

Leo squatting environment

behavior	behavior	Behavior type
xml	string	XML configuration filename
target_env	environment	Interaction environment
observe	string	Comma-separated list of state elements observed by an agent
actuate	string	Comma-separated list of action elements provided by an agent
exporter	exporter	Optional exporter for transition log (supports time, state, observation, a
sub_transition_type	signal/vector	Subscriber to the transition type
pub_ic_signal	signal/vector	Publisher of the initialization and contact signal
measurement_noise	double	Additive measurement noise

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions

G.4 environment/leo_walk

Leo walking environment

behavior	behavior	Behavior type
xml	string	XML configuration filename
target_env	environment	Interaction environment
observe	string	Comma-separated list of state elements observed by an agent
actuate	string	Comma-separated list of action elements provided by an agent
exporter	exporter	Optional exporter for transition log (supports time, state, observation, a
sub_transition_type	signal/vector	Subscriber to the transition type
pub_ic_signal	signal/vector	Publisher of the initialization and contact signal
measurement_noise	double	Additive measurement noise

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions

G.5 environment/modeled

Environment that uses a state transition model internally

model	model	Environment model
task	task	Task to perform in the environment (should match model)
exporter	exporter	Optional exporter for transition log (supports time, state, observation, action, reward, t

Provided parameters

state	signal/vector	Current state of the model

G.6 environment/ode

Open Dynamics Engine simulation environment

xml	string	XML configuration filename
randomize	int	Randomize initial state
visualize	int	Whether to display 3D visualization

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

G.7 environment/pre/noise

Injects noise into an environment

environment	environment	Environment to inject noise into
sensor_noise	vector	Additive sensor noise standard deviation
actuator_noise	vector	Additive actuator noise standard deviation

G.8 environment/pre/shaping

Adds reward shaping to an environment

environment	environment	Environment to inject noise into
shaping_function	mapping	Potential function over states
gamma	double	Discount factor

G.9 environment/sandbox

Non-Markov environment

model	sandbox_model	Environment model
task	task	Task to perform in the environment (should match model)
exporter	exporter	Optional exporter for transition log (supports time, state, observation, action, re

Provided parameters

state	signal/vector	Current state of the model

H Experiments

H.1 experiment/approx_test

Approximator test experiment (supervised learning)

train_samples	int	Number of training samples
test_samples	int	Number of test samples
file	string	Output file (csv format)
input_min	vector	Lower limit for drawing samples
input_max	vector	Upper limit for drawing samples
projector	projector	Projector (should match representation)
representation	representation	Learned representation
mapping	mapping	Function to learn

H.2 experiment/batch_learning

Batch learning experiment using randomly sampled experience

runs	int	Number of separate learning runs to perform
batches	int	Number of batches per learning run
batch_size	int	Number of transitions per batch
rate	int	Test trial control step frequency in Hz
output	string	Output base filename
model	model	Model in which the task is set
task	task	Task to be solved
predictor	predictor	Learner
test_agent	agent	Agent to use in test trials after each batch
observation_min	vector.observation_min	Lower limit for observations
observation_max	vector.observation_max	Upper limit for observations
action_min	vector.action_min	Lower limit for actions
action_max	vector.action_max	Upper limit for actions

Provided parameters

state	signal/vector	Current observed state of the environment

H.3 experiment/multi

Run multiple experiments in parallel

instances	int	Number of experiments to run in parallel
experiment	experiment	Experiment to run

H.4 experiment/online_learning

Interactive learning experiment

runs	int	Number of separate learning runs to perform
trials	int	Number of episodes per learning run
steps	int	Number of steps per learning run
rate	int	Control step frequency in Hz
test_interval	int	Number of episodes in between test trials
output	string	Output base filename
environment	environment	Environment in which the agent acts
agent	agent	Agent
test_agent	agent	Agent to use in test trials
load_file	string	Load policy filename
save_every	string	Save policy to 'output' at the end of event

Provided parameters

state	signal/vector	Current observed state of the environment
action	signal/vector	Current action applied to the environment
curve	signal/vector	Learning curve

H.5 experiment/rpc/environment

Environment RPC server

port	int	Listen port
environment	environment	Environment to interface

I Exporters

I.1 exporter/csv

Comma-separated values exporter

file	string	Output base filename
fields	string	Comma-separated list of fields to write
style	string	Header style
variant	string	Variant to export
enabled	int	Enable writing to output file

J Importers

J.1 importer/csv

Comma-separated values importer

file	string	Input base filename
fields	string	Comma-separated list of fields to read

K Mappings

K.1 mapping/displacement

Mapping that returns the state displacement effected by a policy

policy	mapping/policy	Policy for which displacement is calculated
model	observation_model	Observation model on which policy acts

K.2 mapping/multisine

Sum of sines mapping

inputs	int	Number of input dimensions
outputs	int	Number of output dimensions
sines	int	Number of sines

K.3 mapping/policy/action

Policy based on a direct action representation

sigma	vector	Standard deviation of exploration distribution
output_min	vector.action_min	Lower limit on outputs
output_max	vector.action_max	Upper limit on outputs
projector	projector.observation	Projects observations onto representation space
representation	representation.action	Action representation

K.4 mapping/policy/action_probability

Policy based on an action-probability representation

discretizer	discretizer	Action discretizer
projector	projector	Projects observation-action pairs onto representation space
representation	representation	Action-probability representation

K.5 mapping/policy/discrete/random

Policy that chooses discrete random actions

discretizer	discretizer.action	Action discretizer

K.6 mapping/policy/feed_forward

Feed-forward policy

controls	mapping	Maps time to controls

K.7 mapping/policy/mcts

Monte-Carlo Tree Search policy

model	observation_model	Observation model used for planning
discretizer	discretizer.action	Action discretizer
gamma	double	Discount rate
epsilon	double	Exploration rate
horizon	int	Planning horizon
budget	double	Computational budget

K.8 mapping/policy/parameterized/action

Parameterized policy based on a direct action representation

sigma	vector	Standard deviation of exploration distribution
output_min	vector.action_min	Lower limit on outputs
output_max	vector.action_max	Upper limit on outputs
projector	projector.observation	Projects observations onto representation space
representation	representation/parameterized.action	Action representation

K.9 mapping/policy/parameterized/pid

Parameterized policy based on a proportional-integral-derivative controller

setpoint	vector	Setpoint
outputs	int.action_dims	Number of outputs
p	vector	P gains ([out1_in1, ..., out1_inN, ..., outN_in1, ..., outN_inN])
i	vector	I gains
d	vector	D gains (use P gain on velocity instead, if available)
il	vector	Integration limits
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions

K.10 mapping/policy/parameterized/pidt

Parameterized policy based on a proportional-integral-derivative controller for trajectory tracking

trajectory	mapping	Maps time to setpoints
inputs	int.observation_dims	Number of inputs
outputs	int.action_dims	Number of outputs
p	vector	P gains ([out1_in1, ..., out1_inN, ..., outN_in1, ..., outN_inN])
i	vector	I gains
d	vector	D gains (use P gain on velocity instead, if available)
il	vector	Integration limits
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions

K.11 mapping/policy/parameterized/state_feedback

Parameterized policy based on a state feedback controller

operating_state	vector	Operating state around which gains are defined
operating_action	vector	Operating action around which gains are defined
gains	vector	Gains ([in1_out1, ..., in1_outN, ..., inN_out1, ..., inN_outN])
output_min	vector.action_min	Lower action limit
output_max	vector.action_max	Upper action limit

K.12 mapping/policy/post/noise

Postprocesses policy output by injecting noise

sigma	vector	Standard deviation of Gaussian exploration distribution
theta	vector	Ornstein-Uhlenbeck friction term (1=pure Gaussian noise)
policy	mapping/policy	Policy to inject noise into

K.13 mapping/policy/random

Policy that chooses continuous random actions

output_min	vector.action_min	Lower action limit
output_max	vector.action_max	Upper action limit

K.14 mapping/policy/sample_feedback

Policy based on state feedback controller defined over samples

output_min	vector.action_min	Lower action limit
output_max	vector.action_max	Upper action limit

K.15 mapping/policy/uct

Monte-Carlo Tree Search policy using UCB1 action selection

model	observation_model	Observation model used for planning
discretizer	discretizer.action	Action discretizer
gamma	double	Discount rate
epsilon	double	Exploration rate
horizon	int	Planning horizon
budget	double	Computational budget

K.16 mapping/policy/value/q

Q-value based policy

discretizer	discretizer.action	Action discretizer
projector	projector.pair	Projects observation-action pairs onto representation space
representation	representation.value/action	Action-value representation
sampler	sampler	Samples actions from action-values

K.17 mapping/policy/value/q/bounded

Q-value based policy with bounded action deltas

bound	vector	Maximum action delta
discretizer	discretizer.action	Action discretizer
projector	projector.pair	Projects observation-action pairs onto representation space
representation	representation.value/action	Action-value representation
sampler	sampler	Samples actions from action-values

K.18 mapping/policy/value/q/ucb

UCB1 policy

discretizer	discretizer.action	Action discretizer
projector	projector.pair	Projects observation-action pairs onto representation spac
representation	representation.value/action	Q-value representation
visit_representation	representation.value/action	Visit count representation
c_p	double	UCB1 exploration term

K.19 mapping/policy/value/v

State-value based policy

gamma	double	Discount rate
discretizer	discretizer.action	Action discretizer
model	observation_model	Observation model
projector	projector.observation	Projects observations onto representation space
representation	representation.value/state	State-value representation
sampler	sampler	Samples actions from state-values

K.20 mapping/puddle

Random 2D puddles

seed	int	World seed
smoothing	double	Standard deviation of Gaussian filter
steepness	double	Parameter of sigmoid stretching

K.21 mapping/represented

A mapping that internally uses a representation

projector	projector	Projects inputs onto representation space
representation	representation	Representation

K.22 mapping/timeline

Imported timeline mapping

importer	importer.dynamic	Importer with time as the first column

K.23 mapping/value

Mapping that returns the expected value of a value-based policy

policy	mapping/policy/value	Value based policy

L Models

L.1 model/compass_walker

Simplest walker model from Garcia et al.

control_step	double.control_step	Control step time
integration_steps	int	Number of integration steps per control step
slope_angle	double.slope_angle	Inclination of the slope

L.2 model/dynamical

State transition model that integrates equations of motion

control_step	double.control_step	Control step time
integration_steps	int	Number of integration steps per control step
dynamics	dynamics	Equations of motion

L.3 model/pinball

Model of a ball on a plate

control_step	double.control_step	Control step time
integration_steps	int	Number of integration steps per control step
restitution	double	Coefficient of restitution
radius	double	Ball radius
maze	int	Maze ID

L.4 model/puddle

Puddle world model

drag	double	Velocity multiplier for puddles
map	mapping/puddle	Puddle map

L.5 model/windy

Sutton & Barto's windy gridworld model

M Observation_models

M.1 observation_model/approximated

Observation model based on observed transitions

jacobian_step	double	Step size for Jacobian estimation
control_step	double.control_step	Control step time (0 = estimate using SMDP approximator)
differential	vector.differential	State dimensions for which to predict deltas
wrapping	vector.wrapping	Wrapping boundaries
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
stddev_limit	double	Maximum standard deviation of acceptable predictions, as fract
projector	projector.pair	Projector for transition model (—S—+—A— dimensions)
representation	representation.transition	Representation for transition model (—S—+2 dimensions)

M.2 observation_model/fixed

Observation model based on known state transition model

jacobian_step	double	Step size for Jacobian estimation
model	model	Environment model
task	task	Task to perform in the environment (should match model)

M.3 observation_model/fixed_reward

Observation model based on observed transitions but known task

jacobian_step	double	Step size for Jacobian estimation
control_step	double.control_step	Control step time (0 = estimate using SMDP approximator)
differential	vector.differential	State dimensions for which to predict deltas
wrapping	vector.wrapping	Wrapping boundaries
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
stddev_limit	double	Maximum standard deviation of acceptable predictions, as fract
projector	projector.pair	Projector for transition model (—S—+—A— dimensions)
representation	representation.transition	Representation for transition model (—S—+2 dimensions)
task	task	Task to perform in the environment

N Optimizers

N.1 optimizer/cma

Coverance matrix adaptation black-box optimizer

population	int	Population size
sigma	vector	Initial standard deviation (a single-element vector will be repli
policy	mapping/policy/parameterized	Control policy prototype

N.2 optimizer/rwa

Reward weighted averaging black-box optimizer

mu	int	Parent population size
lambda	int	Offspring population size
sigma	vector	Standard deviation of exploration
policy	mapping/policy/parameterized	Control policy prototype

O Predictors

O.1 predictor/ac/action

Actor-critic predictor for direct action policies

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation,
alpha	double	Critic learning rate
beta	double	Actor learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
update_method	string	Actor update method
step_limit	vector	Actor exploration step limit
critic_projector	projector.observation	Projects observations onto critic representation space
critic_representation	representation.value/state	Value function representation
critic_trace	trace	Trace of critic projections
actor_projector	projector.observation	Projects observations onto actor representation space
actor_representation	representation.action	Action representation
actor_trace	trace	Trace of actor projections

O.2 predictor/ac/probability

Actor-critic predictor for action-probability policies

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observatio
alpha	double	Critic learning rate
beta	double	Actor learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
critic_projector	projector.observation	Projects observations onto critic representation space
critic_representation	representation.value/state	Value function representation
critic_trace	trace	Trace of critic projections
actor_projector	projector.pair	Projects observation-action pairs onto actor representatio
actor_representation	representation.value/action	Action-probability representation
actor_trace	trace	Trace of actor projections
discretizer	discretizer.action	Action discretizer

O.3 predictor/ac/q

Actor-critic predictor for direct action policies with a Q-value based critic

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observatio
alpha	double	Critic learning rate
beta	double	Actor learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
kappa	double	Advantage scaling factor
update_method	string	Actor update method
step_limit	vector	Actor exploration step limit
target	mapping	Target value at next state
critic_projector	projector.pair	Projects observations onto critic representation space
critic_representation	representation.value/action	Value function representation
critic_trace	trace	Trace of critic projections
actor_projector	projector.observation	Projects observations onto actor representation space
actor_representation	representation.action	Action representation
actor_trace	trace	Trace of actor projections

O.4 predictor/ac/qv

Actor-critic predictor for direct action policies with a Q critic storing advantages over a V critic

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observat
alpha	double	Critic Q learning rate
beta_v	double	Critic V learning rate
beta_a	double	Actor learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
update_method	string	Actor update method
step_limit	vector	Actor exploration step limit
critic_q_projector	projector.pair	Projects observations onto critic Q representation spac
critic_q_representation	representation.value/action	Q Value function representation
critic_v_projector	projector.observation	Projects observations onto critic V representation spac
critic_v_representation	representation.value/state	V Value function representation
critic_v_trace	trace	Trace of critic V projections
actor_projector	projector.observation	Projects observations onto actor representation space
actor_representation	representation.action	Action representation
actor_trace	trace	Trace of actor projections

O.5 predictor/advantage

Advantage learning off-policy value function predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, acti
alpha	double	Learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
kappa	double	Advantage scaling factor
discretizer	discretizer.action	Action discretizer
projector	projector.pair	Projects observation-action pairs onto representation space
representation	representation.value/action	A-value representation
trace	trace	Trace of projections

O.6 predictor/dpg

Deterministic policy gradient predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observa
alpha	double	Advantage model learning rate
beta_v	double	Critic learning rate
beta_a	double	Actor learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
projector	projector.state	Projects observations onto representation spaces
critic_representation	representation.value/state	State value function representation
critic_trace	trace	Trace of critic projections
advantage_representation	representation	Local advantage model representation (one output pe
actor_representation	representation.action	Action representation

O.7 predictor/expected_sarsa

Expected SARSA low-variance on-policy value function predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, acti
alpha	double	Learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
projector	projector.pair	Projects observation-action pairs onto representation space
representation	representation.value/action	Q-value representation
policy	mapping/policy/value	Value based target policy
trace	trace	Trace of projections

O.8 predictor/fqi

Fitted Q-iteration predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, a
gamma	double	Discount rate
transitions	int	Maximum number of transitions to store
iterations	int	Number of policy improvement rounds per episode
reset_strategy	string	At which point to reset the representation
macro_batch_size	int	Number of episodes/batches after which prediction is rebuilt
discretizer	discretizer.action	Action discretizer
projector	projector.pair	Projects observations onto critic representation space
representation	representation.value/action	Value function representation

O.9 predictor/full/qi

Deterministic model-based action-value function predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, acti
gamma	double	Discount rate
model	observation_model	Observation model used for planning
discretizer	discretizer.action	Action discretizer
projector	projector.pair	Projects observation-action pairs onto representation space
representation	representation.value/action	Action-value function representation

O.10 predictor/full/vi

Deterministic model-based state-value function predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, actio
gamma	double	Discount rate
model	observation_model	Observation model used for planning
discretizer	discretizer.action	Action discretizer
projector	projector.observation	Projects observations onto representation space
representation	representation.value/state	State-value function representation

O.11 predictor/ggq

Greedy-GQ off-policy value function predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, acti
alpha	double	Learning rate
eta	double	Relative secondary learning rate (actual is alpha*eta)
gamma	double	Discount rate
projector	projector.pair	Projects observation-action pairs onto representation space
representation	representation.value/action	(Q, w) representation
policy	mapping/policy/value	Greedy target policy

O.12 predictor/mbfqi

Minibatch FQI predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, act
gamma	double	Discount rate
transitions	int	Maximum number of transitions to store
minibatch_size	int	Number of transitions to average gradient over.
update_interval	int	Number of minibatches between target updates.
discretizer	discretizer	Action discretizer
projector	projector.pair	Projects observations onto critic representation space
representation	representation.value/action	Value function representation

O.13 predictor/model

Observation model predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, action
differential	vector.differential	State dimensions for which to predict deltas
wrapping	vector.wrapping	Wrapping boundaries
projector	projector.pair	Projector for transition model (—S—+—A— dimensions)
representation	representation.transition	Representation for transition model (—S—+2 dimensions)

O.14 predictor/multi

Updates multiple predictors

predictor1	predictor	First downstream predictor
predictor2	predictor	Second downstream predictor

O.15 predictor/qv

QV on-policy value function predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, ac
alpha	double	State-action value learning rate
beta	double	State value learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
q_projector	projector.pair	Projects observation-action pairs onto representation space
q_representation	representation.value/action	State-action value representation (Q)
v_projector	projector.observation	Projects observations onto representation space
v_representation	representation.value/state	State value representation (V)
trace	trace	Trace of projections

O.16 predictor/sarsa

SARSA on-policy value function predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, acti
alpha	double	Learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
projector	projector.pair	Projects observation-action pairs onto representation space
representation	representation.value/action	Q-value representation
trace	trace	Trace of projections

O.17 predictor/td

TD value function predictor

importer	importer.static	Optional importer for pre-training
exporter	exporter	Optional exporter for transition log (supports observation, actio
alpha	double	Learning rate
gamma	double	Discount rate
lambda	double	Trace decay rate
projector	projector.observation	Projects observations onto representation space
representation	representation.value/state	State value representation
trace	trace	Trace of projections

P Projectors

P.1 projector/fourier

Fourier basis function projector

input_min	vector	Lower input dimension limit (for scaling)
input_max	vector	Upper input dimension limit (for scaling)
order	int	Order of approximation (bases per dimension)
parity	string	Whether to use odd or even bases

Provided parameters

memory	int.memory	Feature vector size

P.2 projector/grid/index

Discretizes continuous input to a linear grid index

discretizer	discretizer	Discretizer

Provided parameters

memory	int.memory	Grid size

P.3 projector/grid/position

Discretizes continuous input to a grid center position

discretizer	discretizer	Discretizer

Provided parameters

memory	int.memory	Grid size

P.4 projector/identity

Simply returns the input vector

P.5 projector/monomial

Monomial basis function projector

operating_input	vector	Origin
degree	int	Maximum degree of monomials

Provided parameters

memory	int.memory	Feature vector size

P.6 projector/multi

Combines multiple projections

dim	int	Indicator dimension (-1=union)
projector1	projector.	First downstream projector
projector2	projector.	Second downstream projector
memories	vector.memory	Memory of downstream projectors

Provided parameters

memory	int.memory	Feature vector size

P.7 projector/pre/normalizing

Preprocesses projection onto a normalized [0, 1] vector

input_min	vector	Lower input dimension limit (for scaling)
input_max	vector	Upper input dimension limit (for scaling)
projector	projector.	Downstream projector

P.8 projector/pre/peaked

Preprocesses projection for more resolution around center

peaking	vector	Extra resolution factor around center (offset by 1/factor at edges)
input_min	vector	Lower input dimension limit (for scaling)
input_max	vector	Upper input dimension limit (for scaling)
projector	projector.	Downstream projector

P.9 projector/pre/scaling

Preprocesses projection onto a scaled vector

scaling	vector	Scaling vector
projector	projector.	Downstream projector

P.10 projector/rbf

Projection on a grid of triangular radial basis functions

input_min	vector	Lower input dimension limit
input_max	vector	Upper input dimension limit
steps	vector	Basis functions per dimension

Provided parameters

memory	int.memory	RBF size

P.11 projector/sample/ann

Projects onto samples found through approximate nearest-neighbor search

samples	int	Maximum number of samples to store
neighbors	int	Number of neighbors to return
locality	double	Locality of weighing function
interval	int	Samples to accumulate before rebuilding kd-tree
incremental	int	Search samples that haven't been indexed yet
bucket_size	int	?
error_bound	double	?
inputs	int	Number of input dimensions

P.12 projector/sample/ertree

Projects onto samples found through the Extra-trees algorithm by Geurts et al.

samples	int	Maximum number of samples to store
trees	int	Number of trees in the forest
splits	int	Number of candidate splits
leaf_size	int	Maximum number of samples in a leaf
inputs	int	Number of input dimensions
outputs	int	Number of output dimensions

P.13 projector/split

Splits a feature vector into distinct sets

index	vector	Binary vector that specifies which dimensions to use as index
discretizer	discretizer	Determines the distinct set based on the index dimensions
projector	projector	Projects the non-index dimensions onto a feature vector
projector_memory	int.memory	Memory of downstream projector

Provided parameters

memory	int.memory	Resulting feature vector size

P.14 projector/tile_coding

Hashed tile coding projector

tilings	int	Number of tilings
memory	int.memory	Hash table size
safe	int	Collision detection (0=off, 1=claim on write, 2=claim always)
resolution	vector	Size of a single tile
wrapping	vector.wrapping	Wrapping boundaries (must be multiple of resolution)

Q Random_generators

Q.1 random_generator/normal

Normal Random generator

mu	double	Mean
sigma	double	Standart deviation

Q.2 random_generator/ornstein_uhlenbeck

Ornstein-Uhlenbeck Random generator

center	double	Attraction point
theta	double	Theta
sigma	double	Sigma

Q.3 random_generator/uniform

Uniform Random generator

lower	double	Lower bound of an interval
upper	double	Upper bound of an interval

Q.4 random_generator/uniform_integer

Uniform Integer Random generator

ma	int	Upper bound of an interval [0, ma)

R Representations

R.1 representation/additive

Linear combination of two representations

representation1	representation.	First representation
representation2	representation.	Second representation
learning	int	Which representation to learn (0=both)

R.2 representation/communicator

Interface to an out-of-process representation

inputs	int	Number of input dimensions
outputs	int	Number of output dimensions
communicator	communicator	Communicator which exchanges messages with the out-of-process representa

R.3 representation/dictionary

Stores examples as key-value pairs in a dictionary

R.4 representation/iterative

Representation that iteratively trains a sub-representation

epochs	int	Learning epochs
cumulative	int	Add to training set instead of replacing it
representation	representation.	Downstream representation

R.5 representation/llr

Performs locally linear regression through samples

ridge	double	Ridge regression (Tikhonov) factor
order	int	Order of regression model
input_nominals	vector	Vector indicating which input dimensions are nominal
output_nominals	vector	Vector indicating which output dimensions are nominal
outputs	int	Number of output dimensions
output_min	vector	Lower output limit
output_max	vector	Upper output limit
projector	projector/sample	Projector used to generate input for this representation

R.6 representation/parameterized/ann

Parameterized artificial neural network representation

inputs	int	Number of input dimensions
outputs	int	Number of output dimensions
hiddens	vector	Number of hidden nodes per layer
eta	double	Learning rate (0=RPROP, ¡0=RMSPROP)

R.7 representation/parameterized/linear

Linear-in-parameters representation

init_min	vector	Lower initial value limit
init_max	vector	Upper initial value limit
memory	int.memory	Feature vector size
outputs	int	Number of outputs
output_min	vector	Lower output limit
output_max	vector	Upper output limit

S Samplers

S.1 sampler/ac_ornstein_ohlenbeck

Action-correlated maximum search with an Ornstein-Uhlenbeck random chance of non-maximums

rand_max	int	In case of multiple maximumum values select a random index among t
discretizer	discretizer.action	Action discretizer
theta	vector	Theta parameter of Ornstein-Uhlenbeck
sigma	vector	Sigma parameter of Ornstein-Uhlenbeck
center	vector	Centering parameter of Ornstein-Uhlenbeck
pub_sub_ou_state	signal/vector	Publisher and subscriber to the value of noise (or action in the ACOU
epsilon	double	Exploration rate

S.2 sampler/epsilon_greedy

Maximum search with a uniform random chance of non-maximums

rand_max	int	In case of multiple maximumum values select a random index among them
epsilon	vector	Exploration rate (can be defined per action)

S.3 sampler/epsilon_ornstein_ohlenbeck

Exploitations are done by greedy action selection without constraints, as in e-greedy. Explorations are done with time-correlated noise, as it is in OU.

rand_max	int	In case of multiple maximumum values select a random index among t
discretizer	discretizer.action	Action discretizer
theta	vector	Theta parameter of Ornstein-Uhlenbeck
sigma	vector	Sigma parameter of Ornstein-Uhlenbeck
center	vector	Centering parameter of Ornstein-Uhlenbeck
pub_sub_ou_state	signal/vector	Publisher and subscriber to the value of noise (or action in the ACOU
epsilon	double	Exploration rate

S.4 sampler/epsilon_pada

exploitations are done by greedy action selection without constraints, as in e-greedy. Explorations are done with constrained set of actions, as it is in pada.

rand_max	int	In case of multiple maximumum values select a random index amon
epsilon	vector	Exploration rate (can be defined per action)
discretizer	discretizer.action	Action discretizer
delta	vector	Delta of PADA
pub_sub_pada_state	signal/vector	Publisher and subscriber to the value of action of the PADA familiy

S.5 sampler/greedy

Maximum search

rand_max	int	In case of multiple maximumum values select a random index among them

S.6 sampler/leo/action

Wrapper for an action sampler for Leo (can modify memory of samplers with memory at contact events)

sampler	sampler	Samples actions from action-values
sub_ic_signal	signal/vector	Subscrider to the initialization and contact signal
pub_sub_sampler_state	signal/vector	Publisher and subscriber of the sampler state with memory such as p

S.7 sampler/ornstein_ohlenbeck

Maximum search with an Ornstein-Uhlenbeck random chance of non-maximums

rand_max	int	In case of multiple maximumum values select a random index among t
discretizer	discretizer.action	Action discretizer
theta	vector	Theta parameter of Ornstein-Uhlenbeck
sigma	vector	Sigma parameter of Ornstein-Uhlenbeck
center	vector	Centering parameter of Ornstein-Uhlenbeck
pub_sub_ou_state	signal/vector	Publisher and subscriber to the value of noise (or action in the ACOU

S.8 sampler/pada

Maximum search with a PADA random chance of non-maximums

rand_max	int	In case of multiple maximumum values select a random index amon
epsilon	vector	Exploration rate (can be defined per action)
discretizer	discretizer.action	Action discretizer
delta	vector	Delta of PADA
pub_sub_pada_state	signal/vector	Publisher and subscriber to the value of action of the PADA familiy

S.9 sampler/pada_ornstein_ohlenbeck

Explorations and exploitations are same as OU, but action is selected from a constrained set, as in PADA.

rand_max	int	In case of multiple maximumum values select a random index among t
discretizer	discretizer.action	Action discretizer
theta	vector	Theta parameter of Ornstein-Uhlenbeck
sigma	vector	Sigma parameter of Ornstein-Uhlenbeck
center	vector	Centering parameter of Ornstein-Uhlenbeck
pub_sub_ou_state	signal/vector	Publisher and subscriber to the value of noise (or action in the ACOU
pada	sampler	Pada sampler
pub_new_action	signal/vector	Publisher of the signal with noise

S.10 sampler/softmax

Softmax (Gibbs/Boltzmann) sampler

tau	double	Temperature of Boltzmann distribution

T Sandbox_models

T.1 sandbox_model/compass_walker

Simplest walker model from Garcia et al. with a sequential evaluation

control_step	double.control_step	Control step time
integration_steps	int	Number of integration steps per control step
slope_angle	double.slope_angle	Inclination of the slope
exporter	exporter	Optional exporter for transition log (supports time, state, observatio
use_avg_velocity	int	Velocity type

T.2 sandbox_model/leo_squatting

State transition model that integrates equations of motion and augments state vector with additional elements

control_step	double.control_step	Control step time
integration_steps	int	Number of integration steps per control step
dynamics	dynamics/rbdl	Equations of motion
target_dof	int.target_dof	Number of degrees of freedom of the target model
animation	string	Save current state or full animation
target_env	environment	Interaction environment
lower_height	double.lower_height	Lower bound of root height to switch direction
upper_height	double.upper_height	Upper bound of root height to switch direction

U Signals

U.1 signal/matrix

Matrix-based signal (trajectory, etc.)

U.2 signal/vector

Vector-based signal (state, observation, etc.)

V Solvers

V.1 solver/agent

Solver that uses a simulated agent

steps	int	Number of planning steps before solution is returned
horizon	int	Planning episode length
start	vector	Starting state for planning
model	observation_model	Observation model used for planning
agent	agent	Agent used for planning episodes

Provided parameters

state	state	Current observed state of planning

V.2 solver/ilqg

Iterative Linear Quadratic Gaussian trajectory optimizer

horizon	int	Horizon
iterations	int	Maximum number of iterations
stddev	vector	Standard deviation of initial random action sequence
regularization	string	Regularization method
model	observation_model	Observation model
policy	mapping/policy/sample_feedback	Sample feedback policy to adjust

Provided parameters

trajectory	signal/matrix	Predicted trajectory

V.3 solver/lqr

Linear Quadratic Regulator solver

operating_state	vector	Operating state around which to linearize
operating_action	vector	Operating action around which to lineariz
model	observation_model	Observation model
policy	mapping/policy/parameterized/state_feedback	State feedback policy to adjust

V.4 solver/vi

Value iteration solver

sweeps	int	Number of planning sweeps before solution is returned
parallel	int	Perform backups in parallel (requires reentrant representation)
discretizer	discretizer.observation	State space discretizer
predictor	predictor/full	Predictor to iterate

W Tasks

W.1 task/acrobot/balancing

Acrobot balancing task

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.2 task/acrobot/regulator

Acrobot regulator task

start	vector	Starting state
goal	vector	Goal state
stddev	vector	Starting state standard deviation
q	vector	Q (state cost) matrix diagonal
r	vector	R (action cost) matrix diagonal
function	string	Cost function style
smoothing	double	Cost function smoothing parameter

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.3 task/cart_double_pole/balancing

Cart-double-pole balancing task

timeout	double	Episode timeout

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.4 task/cart_double_pole/regulator

Cart-double-pole regulator task

start	vector	Starting state
goal	vector	Goal state
stddev	vector	Starting state standard deviation
q	vector	Q (state cost) matrix diagonal
r	vector	R (action cost) matrix diagonal
function	string	Cost function style
smoothing	double	Cost function smoothing parameter
timeout	double	Episode timeout

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.5 task/cart_double_pole/swingup

Cart-double-pole swing-up task

timeout	double	Episode timeout

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.6 task/cart_pole/balancing

Cart-pole balancing task

timeout	double	Episode timeout

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.7 task/cart_pole/regulator

Cart-pole regulator task

start	vector	Starting state
goal	vector	Goal state
stddev	vector	Starting state standard deviation
q	vector	Q (state cost) matrix diagonal
r	vector	R (action cost) matrix diagonal
function	string	Cost function style
smoothing	double	Cost function smoothing parameter
timeout	double	Episode timeout

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.8 task/cart_pole/swingup

Cart-pole swing-up task

timeout	double	Episode timeout
randomization	int	Start state randomization
shaping	int	Whether to use reward shaping
gamma	double	Discount rate for reward shaping
end_stop_penalty	int	Terminate episode with penalty when end stop is reached

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.9 task/compass_walker/vref

Compass walker tracking velocity task

timeout	double	Learning episode timeout
initial_state_variation	double	Variation of initial state
slope_angle	double.slope_angle	Inclination of the slope
negative_reward	double	Negative reward
observe	vector	State elements observed by an agent
steps	int	number of steps after which task is terminated
reference_velocity	double	Reference velocity
per_step_reward	int	If set, give reward per every step

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.10 task/compass_walker/vrefu

Compass walker tracking velocity task with controls minimization

timeout	double	Learning episode timeout
initial_state_variation	double	Variation of initial state
slope_angle	double.slope_angle	Inclination of the slope
negative_reward	double	Negative reward
observe	vector	State elements observed by an agent
steps	int	number of steps after which task is terminated
reference_velocity	double	Reference velocity
per_step_reward	int	If set, give reward per every step

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.11 task/compass_walker/walk

Compass walker walking task

timeout	double	Learning episode timeout
initial_state_variation	double	Variation of initial state
slope_angle	double.slope_angle	Inclination of the slope
negative_reward	double	Negative reward
observe	vector	State elements observed by an agent
steps	int	number of steps after which task is terminated

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.12 task/flyer2d/regulator

2D flyer regulator task

start	vector	Starting state
goal	vector	Goal state
stddev	vector	Starting state standard deviation
q	vector	Q (state cost) matrix diagonal
r	vector	R (action cost) matrix diagonal
function	string	Cost function style
smoothing	double	Cost function smoothing parameter
action_range	double	Range of allowed actions
timeout	double	Episode timeout

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.13 task/leo_squatting

Task specification for Leo squatting with a fixed arm

timeout	double.timeout	Task timeout
rand_init	int.rand_init	Initialization from a random pose

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.14 task/lua

User-provided task specification in LUA

file	string	Lua task file
options	string	Lua string to execute when loading task

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.15 task/mountain/regulator

Mountain world regulator task

start	vector	Starting state
goal	vector	Goal state
stddev	vector	Starting state standard deviation
q	vector	Q (state cost) matrix diagonal
r	vector	R (action cost) matrix diagonal
function	string	Cost function style
smoothing	double	Cost function smoothing parameter

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.16 task/pendulum/regulator

Pendulum regulator task

start	vector	Starting state
goal	vector	Goal state
stddev	vector	Starting state standard deviation
q	vector	Q (state cost) matrix diagonal
r	vector	R (action cost) matrix diagonal
function	string	Cost function style
smoothing	double	Cost function smoothing parameter

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.17 task/pendulum/swingup

Pendulum swing-up task

timeout	double	Episode timeout
randomization	double	Level of start state randomization

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.18 task/pinball/movement

Pinball movement task

tolerance	double	Goal tolerance

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.19 task/pinball/regulator

Pinball regulator task

start	vector	Starting state
goal	vector	Goal state
stddev	vector	Starting state standard deviation
q	vector	Q (state cost) matrix diagonal
r	vector	R (action cost) matrix diagonal
function	string	Cost function style
smoothing	double	Cost function smoothing parameter

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.20 task/puddle/regulator

Puddle world regulator task

start	vector	Starting state
goal	vector	Goal state
stddev	vector	Starting state standard deviation
q	vector	Q (state cost) matrix diagonal
r	vector	R (action cost) matrix diagonal
function	string	Cost function style
smoothing	double	Cost function smoothing parameter
penalty	double	Penalty multiplier for puddles
map	mapping/puddle	Puddle map

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.21 task/swimmer/reaching

Swimmer reaching task

timeout	double	Episode timeout
randomization	double	Level of start state randomization
segments	double.swimmer/segments	Number of swimmer segments

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.22 task/tlm/balancing

Two-link manipulator balancing task

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

W.23 task/windy/movement

Windy gridworld movement task

Provided parameters

observation_dims	int.observation_dims	Number of observation dimensions
observation_min	vector.observation_min	Lower limit on observations
observation_max	vector.observation_max	Upper limit on observations
action_dims	int.action_dims	Number of action dimensions
action_min	vector.action_min	Lower limit on actions
action_max	vector.action_max	Upper limit on actions
reward_min	double.reward_min	Lower limit on immediate reward
reward_max	double.reward_max	Upper limit on immediate reward

X Traces

X.1 trace/enumerated/accumulating

Accumulating eligibility trace using a queue of projections

X.2 trace/enumerated/replacing

Replacing eligibility trace using a queue of projections

Y Trigges

Y.1 trigger

Event trigger

min	vector.observation_min	Minimum of compartment bounding box
max	vector.observation_max	Maximum of compartment bounding box
delay	double	Settlement delay for which conditions are continuously fullfilled

Z Visualizations

Z.1 visualization/acrobot

Acrobot visualization

state	signal/vector	Acrobot state to visualize

Z.2 visualization/cart_double_pole

Cart-double-pole visualization

state	signal/vector	Cart-double-pole state to visualize

Z.3 visualization/cart_pole

Cart-pole visualization

state	signal/vector	Cart-pole state to visualize

Z.4 visualization/compass_walker

Compass walker visualization

state	signal/vector	Compass walker state to visualize

Z.5 visualization/field/mapping

Visualizes a mapping over a field of states

field_dims	vector	Dimensions to visualize
input_min	vector	Lower input dimension limit
input_max	vector	Upper input dimension limit
points	int	Number of points to evaluate
savepoints	int	Number of points to evaluate when saving to file ('s')
state	signal/vector	Optional current state to overlay
projection	string	Method of projecting values onto 2d space
mapping	mapping	Mapping
output_dim	int	Output dimension to visualize

Z.6 visualization/field/policy/value

Visualizes the value of a policy over a field of states

field_dims	vector	Dimensions to visualize
input_min	vector	Lower input dimension limit
input_max	vector	Upper input dimension limit
points	int	Number of points to evaluate
savepoints	int	Number of points to evaluate when saving to file ('s')
state	signal/vector	Optional current state to overlay
projection	string	Method of projecting values onto 2d space
policy	mapping/policy/value	Value based control policy

Z.7 visualization/field/value

Visualizes an approximation over a field of states

field_dims	vector	Dimensions to visualize
input_min	vector	Lower input dimension limit
input_max	vector	Upper input dimension limit
points	int	Number of points to evaluate
savepoints	int	Number of points to evaluate when saving to file ('s')
state	signal/vector	Optional current state to overlay
projection	string	Method of projecting values onto 2d space
output_dim	int	Output dimension to visualize
projector	projector	Projects inputs onto representation space
representation	representation	Value representation

Z.8 visualization/flyer2d

2D flyer visualization

state	signal/vector	2D flyer state to visualize

Z.9 visualization/pendulum

Pendulum visualization

state	signal/vector	Pendulum state to visualize

Z.10 visualization/pinball

Pinball visualization

state	signal/vector	Pinball state to visualize

Z.11 visualization/sample

Visualizes a sample-based approximation

field_dims	vector	Dimensions to visualize
field_min	vector	Lower visualization dimension limit
field_max	vector	Upper visualization dimension limit
output_dim	int	Output dimension to visualize
points	int	Texture size
projector	projector/sample	Sample projector whose store to visualize

Z.12 visualization/sample/random

Visualizes an approximation over randomly sampled states

field_dims	vector	Dimensions to visualize
input_min	vector	Lower input dimension limit
input_max	vector	Upper input dimension limit
output_dim	int	Output dimension to visualize
points	int	Texture size
projector	projector	Projects inputs onto representation space
representation	representation	Value representation

Z.13 visualization/slice

Visualizes a slice from a mapping

field_dims	vector	Dimensions to visualize
input_min	vector	Lower input dimension limit
input_max	vector	Upper input dimension limit
operating_point	vector	Fixed values for non-visualized dimensions
output_dim	int	Output dimension to visualize
points	int	Number of points to evaluate
state	signal/vector	Optional current state to overlay
action	signal/vector	Optional current action to overlay
mapping	mapping	Mapping to visualize

Z.14 visualization/state

Plots state values

input_dims	vector	Input dimensions to visualize
input_min	vector	Lower input dimension limit
input_max	vector	Upper input dimension limit
memory	int	Number of data points to draw
state	signal/vector	State to visualize

Z.15 visualization/swimmer

Swimmer visualization

state	signal/vector	Swimmer state to visualize

Z.16 visualization/tlm

Two-link manipulator visualization

state	signal/vector	Two-link manipulator state to visualize

Z.17 visualization/trajectory

Plots trajectories

input_dims	vector	Input dimensions to visualize
input_min	vector	Lower input dimension limit
input_max	vector	Upper input dimension limit
trajectory	signal/matrix	Trajectory to visualize

Z.18 visualization/windy

Windy gridworld visualization

state	signal/vector	Windy gridworld state to visualize

AA Visualizers

AA.1 visualizer/glut

Visualizer based on the GLUT library