

---

# **Cosmos Documentation**

***Release 1.0***

**Marcel van Gerven**

May 13, 2017



## CONTENTS

<b>1 reinforcement_learning package</b>	<b>3</b>
1.1 Submodules . . . . .	3
1.2 reinforcement_learning.agents module . . . . .	3
1.3 reinforcement_learning.models module . . . . .	4
1.4 reinforcement_learning.networks module . . . . .	4
1.5 reinforcement_learning.tasks module . . . . .	5
1.6 reinforcement_learning.unit_test module . . . . .	5
1.7 reinforcement_learning.world module . . . . .	5
1.8 Module contents . . . . .	6
<b>2 Indices and tables</b>	<b>7</b>
<b>Python Module Index</b>	<b>9</b>
<b>Index</b>	<b>11</b>



Contents:



---

CHAPTER  
ONE

---

## REINFORCEMENT LEARNING PACKAGE

### 1.1 Submodules

### 1.2 reinforcement\_learning.agents module

```
class reinforcement_learning.agents.AACAgent (model, optimizer=None, gamma=0.99,  
beta=0.01, cutoff=None)
```

Bases: *reinforcement\_learning.agents.REINFORCEAgent*

Implements Advantage Actor-Critic algorithm

```
train (observation, reward, done)
```

Trains agent on cumulated reward (return)

**Returns** action (Variable)

```
class reinforcement_learning.agents.NESAgent (model, nsteps=100, npop=50, sigma=0.1, al-  
pha=0.001)
```

Bases: object

Implements Natural Evolution Strategies algorithm

<https://blog.openai.com/evolution-strategies/> <https://gist.github.com/karpathy/77fbb6a8dac5395f1b73e7a89300318d>

hard to implement in same framework since the agent needs to run multiple versions of the task

```
train (observation, reward, done)
```

Trains agent on cumulated reward (return)

**Returns** action (Variable)

```
class reinforcement_learning.agents.REINFORCEAgent (model, optimizer=None,  
gamma=0.99, beta=0.01, cutoff=None)
```

Bases: object

Implements REINFORCE algorithm

```
entropy (pi)
```

Computes entropy of policy

**Parameters** **policy** –

```
reset_state ()
```

Resets persistent states

```
score_function (action, policy)
```

Computes score

### Parameters

- **action** (*int*) –
- **policy** –

### Returns score

**test** (*observation, reward, done*)

Tests agent

### Returns action (Variable)

**train** (*observation, reward, done*)

Trains agent on cumulated reward (return)

### Returns action (Variable)

## 1.3 reinforcement\_learning.models module

**class** reinforcement\_learning.models.**ActorCriticModel** (*net, gpu=-1*)

Bases: *reinforcement\_learning.models.Model*

An actor model computes the action and policy from a predictor

**\_\_call\_\_** (*data*)

### Parameters **data** – observation

### Returns action and policy

**class** reinforcement\_learning.models.**ActorModel** (*net, gpu=-1*)

Bases: *reinforcement\_learning.models.Model*

An actor model computes the action and policy from a predictor

**\_\_call\_\_** (*data*)

### Parameters **data** – observation

### Returns action and policy

**class** reinforcement\_learning.models.**Model** (*net, gpu=-1*)

Bases: chainer.link.Chain

Model which wraps a network to generate predictions and compute policies

**has\_state**

Checks if a network has persistent states

### Returns bool

**predict** (*data*)

Returns an action

**reset\_state** ()

## 1.4 reinforcement\_learning.networks module

**class** reinforcement\_learning.networks.**MLP** (*n\_input=None, n\_output=1, n\_hidden=10*)

Bases: chainer.link.Chain

Multilayer perceptron

**has\_state**

Checks if a network has persistent states

**Returns** bool

```
class reinforcement_learning.networks.RNN (n_input=None, n_output=1, n_hidden=10)
```

Bases: chainer.link.Chain

**has\_state**

Checks if a network has persistent states

**Returns** bool

**reset\_state()**

Resets persistent states

## 1.5 reinforcement\_learning.tasks module

```
class reinforcement_learning.tasks.EvidenceTask (n=2, p=0.8)
```

Bases: object

Very simple task which only requires evaluating present evidence and does not require evidence integration. The actor gets a reward when it correctly decides on the ground truth. Ground truth 0/1 determines probabilistically the number of 0s or 1s as observations

**reset()**

Resets state and generates new observations

**Returns** observations, reward, done

**step(action)**

This task always produces a new state and observation after each decision

**Parameters** **action** – agent(s) action

**Returns**

## 1.6 reinforcement\_learning.unit\_test module

```
class reinforcement_learning.unit_test.UnitTest (methodName='runTest')
```

Bases: unittest.case.TestCase

**test\_aac\_stateful()**

Test Advantage Actor-Critic on stateful network

**test\_reinforce\_stateful()**

Test REINFORCE on stateful network

**test\_reinforce\_stateless()**

Test REINFORCE on stateless network

## 1.7 reinforcement\_learning.world module

```
class reinforcement_learning.world.World (agents, out='result')
```

Bases: object

Wrapper object which takes care of training and testing on some data iterator for one or more agents

**test** (*task*, *n\_steps*)

**Parameters**

- **task** – task to run agent(s) on
- **n\_steps** (*int*) – number of steps to train on

**Returns** test loss and reward

**train** (*task*, *n\_steps*, *snapshot=0*)

**Parameters**

- **task** – task to run agent(s) on
- **n\_steps** (*int*) – number of steps to train on
- **snapshot** (*int*) – whether or not to save model after each epochs modulo snapshot

**Returns** rewards

## 1.8 Module contents

---

**CHAPTER  
TWO**

---

**INDICES AND TABLES**

- genindex
- modindex
- search



## PYTHON MODULE INDEX

r

reinforcement\_learning, 6  
reinforcement\_learning.agents, 3  
reinforcement\_learning.models, 4  
reinforcement\_learning.networks, 4  
reinforcement\_learning.tasks, 5  
reinforcement\_learning.unit\_test, 5  
reinforcement\_learning.world, 5



## Symbols

<code>__call__()</code> (reinforcement_learning.models.ActorCriticModel method), 4	reinforcement_learning.models (module), 4
<code>__call__()</code> (reinforcement_learning.models.ActorModel method), 4	reinforcement_learning.networks (module), 4
	reinforcement_learning.tasks (module), 5
	reinforcement_learning.unit_test (module), 5
	reinforcement_learning.world (module), 5
	<code>reset()</code> (reinforcement_learning.tasks.EvidenceTask method), 5
	<code>reset_state()</code> (reinforcement_learning.agents.REINFORCEAgent method), 3
	<code>reset_state()</code> (reinforcement_learning.models.Model method), 4
	<code>reset_state()</code> (reinforcement_learning.networks.RNN method), 5
	RNN (class in reinforcement_learning.networks), 5
	<b>S</b>
	<code>score_function()</code> (reinforcement_learning.agents.REINFORCEAgent method), 3
	<code>step()</code> (reinforcement_learning.tasks.EvidenceTask method), 5
	<b>T</b>
	<code>test()</code> (reinforcement_learning.agents.REINFORCEAgent method), 4
	<code>test()</code> (reinforcement_learning.world.World method), 6
	<code>test_aac_stateful()</code> (reinforcement_learning.unit_test.UnitTest method), 5
	<code>test_reinforce_stateful()</code> (reinforcement_learning.unit_test.UnitTest method), 5
	<code>test_reinforce_stateless()</code> (reinforcement_learning.unit_test.UnitTest method), 5
	<code>train()</code> (reinforcement_learning.agents.AACAgent method), 3
	<code>train()</code> (reinforcement_learning.agents.NESAgent method), 3
	<code>train()</code> (reinforcement_learning.agents.REINFORCEAgent method), 4

train() (reinforcement\_learning.world.World method), [6](#)

## U

UnitTest (class in reinforcement\_learning.unit\_test), [5](#)

## W

World (class in reinforcement\_learning.world), [5](#)