

SimpleMDPLibrary

Generated by Doxygen 1.8.11

Contents

1	Simple MMDP Library - A simple Markov Decision Process Library.	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	MDP Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Member Data Documentation	7
4.1.2.1	a	7
4.1.2.2	gamma	7
4.1.2.3	P	8
4.1.2.4	pi	8
4.1.2.5	r	8
4.1.2.6	s	8
4.1.2.7	t	8
4.1.2.8	v	8

5	File Documentation	9
5.1	mdp.c File Reference	9
5.2	mdp.h File Reference	9
5.2.1	Detailed Description	10
5.2.2	Function Documentation	10
5.2.2.1	allocate_MDP(int s, int a, float gamma)	10
5.2.2.2	best_expected_action(MDP *mdp, int s)	10
5.2.2.3	best_expected_u(MDP *mdp, int s)	11
5.2.2.4	compute_optimal_policy(MDP *mdp)	11
5.2.2.5	error(MDP *mdp, float *v)	11
5.2.2.6	expected_u(MDP *mdp, int s, int a)	11
5.2.2.7	print_MDP(MDP *mdp)	11
5.2.2.8	random_MDP(MDP *mdp)	12
5.2.2.9	release_MDP(MDP *mdp)	12
5.2.2.10	v_iteration_compute(MDP *mdp, float epsilon)	12
5.3	random_example.c File Reference	12
5.3.1	Detailed Description	12
5.3.2	Function Documentation	13
5.3.2.1	main(int argc, char **argv)	13
5.4	ryn_chapter_17.c File Reference	13
5.4.1	Detailed Description	13
5.4.2	Function Documentation	14
5.4.2.1	main(int argc, char **argv)	14
	Index	15

Chapter 1

Simple MMDP Library - A simple Markov Decision Process Library.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

MDP	A Markov Decision Process	7
---------------------	-------------------------------------	-------------------

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

mdp.c	Simple MDP library	9
mdp.h	Simple MDP library	9
random_example.c	Example with a random instance of a 3 state MDP	12
ryn_chapter_17.c	Simple MDP library	13

Chapter 4

Class Documentation

4.1 MDP Struct Reference

A Markov Decision Process.

```
#include <mdp.h>
```

Public Attributes

- int [s](#)
- int [a](#)
- float [gamma](#)
- float *** [P](#)
- unsigned char * [t](#)
- float * [r](#)
- float * [v](#)
- int * [pi](#)

4.1.1 Detailed Description

A Markov Decision Process.

Structure containing all parameters, infinite horizon expected rewards and optimal policy.

4.1.2 Member Data Documentation

4.1.2.1 int MDP::a

number of actions

4.1.2.2 float MDP::gamma

discount factor

4.1.2.3 float* MDP::P**

transition probabilities, $P(s'|s,a)$ (access: `P[s'] [s][a]`)

4.1.2.4 int* MDP::pi

the optimal policy

4.1.2.5 float* MDP::r

the reward $r(s)$

4.1.2.6 int MDP::s

number of states

4.1.2.7 unsigned char* MDP::t

terminal states

4.1.2.8 float* MDP::v

the infinite horizon expected rewards

The documentation for this struct was generated from the following file:

- [mdp.h](#)

Chapter 5

File Documentation

5.1 mdp.c File Reference

Simple [MDP](#) library.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <float.h>
#include <math.h>
#include "mdp.h"
Include dependency graph for mdp.c:
```

5.2 mdp.h File Reference

Simple [MDP](#) library.

This graph shows which files directly or indirectly include this file:

Classes

- struct [MDP](#)
A Markov Decision Process.

Functions

- [MDP](#) * [allocate_MDP](#) (int s, int a, float gamma)
Allocates memory for an [MDP](#).
- void [release_MDP](#) ([MDP](#) *mdp)
Releases memory for an [MDP](#).
- void [print_MDP](#) ([MDP](#) *mdp)
Prints the [MDP](#) parameters.
- void [random_MDP](#) ([MDP](#) *mdp)
Set random probabilities for the state transitions.

- void `v_iteration_compute` (`MDP *mdp`, float epsilon)
Compute $v^(s)$ using value iteration.*
- float `best_expected_u` (`MDP *mdp`, int s)
Computes the best expected utility value for the state s .
- int `best_expected_action` (`MDP *mdp`, int s)
Computes the action for the best expected utility value for the state s .
- float `expected_u` (`MDP *mdp`, int s, int a)
Computes the expected utility value for the state s when taking action a .
- float `error` (`MDP *mdp`, float *v)
Computes the maximum error between the values of the `MDP` and the given vector v .
- void `compute_optimal_policy` (`MDP *mdp`)
Computes the optimal policy for the given `MDP`.

5.2.1 Detailed Description

Simple `MDP` library.

Author

Stalin Muñoz Gutiérrez

Date

27 may 2018 Simple Markov Decision Process library header file

5.2.2 Function Documentation

5.2.2.1 `MDP* allocate_MDP (int s, int a, float gamma)`

Allocates memory for an `MDP`.

Parameters

<code>s</code>	the number of states
<code>a</code>	the number of actions
<code>gamma</code>	the discount factor for future rewards

5.2.2.2 `int best_expected_action (MDP * mdp, int s)`

Computes the action for the best expected utility value for the state s .

Parameters

<code>mdp</code>	the <code>MDP</code>
<code>s</code>	the state for the computation

5.2.2.3 float best_expected_u (MDP * mdp, int s)

Computes the best expected utility value for the state s.

Parameters

<i>mdp</i>	the MDP
<i>s</i>	the state for the computation

5.2.2.4 void compute_optimal_policy (MDP * mdp)

Computes the optimal policy for the given [MDP](#).

Parameters

<i>mdp</i>	the MDP
------------	-------------------------

5.2.2.5 float error (MDP * mdp, float * v)

Computes the maximum error between the values of the [MDP](#) and the given vector v.

Parameters

<i>mdp</i>	the MDP
<i>v</i>	the vector of values for the computation

5.2.2.6 float expected_u (MDP * mdp, int s, int a)

Computes the expected utility value for the state s when taking action a.

Parameters

<i>mdp</i>	the MDP
<i>s</i>	the state for the computation
<i>a</i>	the action to take

5.2.2.7 void print_MDP (MDP * mdp)

Prints the [MDP](#) parameters.

Parameters

<i>mdp</i>	the MDP to print
------------	----------------------------------

5.2.2.8 void random_MDP (MDP * mdp)

Set random probabilities for the state transitions.

Parameters

<i>mdp</i>	the MDP
------------	-------------------------

5.2.2.9 void release_MDP (MDP * mdp)

Releases memory for an [MDP](#).

Parameters

<i>the</i>	MDP to release
------------	--------------------------------

5.2.2.10 void v_iteration_compute (MDP * mdp, float epsilon)

Compute $v^*(s)$ using value iteration.

Parameters

<i>mdp</i>	the MDP
<i>epsilon</i>	the maximum allowed error for the computation of $v(s)$

5.3 random_example.c File Reference

Example with a random instance of a 3 state [MDP](#).

```
#include <stdio.h>
#include "mdp.h"
```

Include dependency graph for random_example.c:

Functions

- void **random_example** ()
- int **main** (int argc, char **argv)
Simple invocation of a random instance of an [MDP](#).

5.3.1 Detailed Description

Example with a random instance of a 3 state [MDP](#).

Author

Stalin Muñoz Gutiérrez

Date

27 may 2018

5.3.2 Function Documentation**5.3.2.1 int main (int *argc*, char ** *argv*)**

Simple invocation of a random instance of an [MDP](#).

Parameters

<i>argc</i>	number of command line params
<i>argv</i>	no params for this example

5.4 ryn_chapter_17.c File Reference

Simple [MDP](#) library.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <float.h>
#include <math.h>
#include "mdp.h"
Include dependency graph for ryn_chapter_17.c:
```

Functions

- void **ryn_chapter_17** ()
- int [main](#) (int *argc*, char ***argv*)

Runs the Russell & Norvig chapter 17 example Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Pearson Education Limited, 2016.

5.4.1 Detailed Description

Simple [MDP](#) library.

Author

Stalin Muñoz Gutiérrez

Date

28 may 2018 Simple Markov Decision Process library implementation.

5.4.2 Function Documentation

5.4.2.1 `int main (int argc, char ** argv)`

Runs the Russell & Norvig chapter 17 example Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Pearson Education Limited, 2016.

Parameters

<i>argc</i>	number of command line params
<i>argv</i>	no params for this example

Index

a

MDP, [7](#)

allocate_MDP
mdp.h, [10](#)

best_expected_action
mdp.h, [10](#)

best_expected_u
mdp.h, [10](#)

compute_optimal_policy
mdp.h, [11](#)

error
mdp.h, [11](#)
expected_u
mdp.h, [11](#)

gamma
MDP, [7](#)

MDP, [7](#)
a, [7](#)
gamma, [7](#)
P, [7](#)
pi, [8](#)
r, [8](#)
s, [8](#)
t, [8](#)
v, [8](#)

main
random_example.c, [13](#)
ryn_chapter_17.c, [14](#)

mdp.c, [9](#)

mdp.h, [9](#)
allocate_MDP, [10](#)
best_expected_action, [10](#)
best_expected_u, [10](#)
compute_optimal_policy, [11](#)
error, [11](#)
expected_u, [11](#)
print_MDP, [11](#)
random_MDP, [12](#)
release_MDP, [12](#)
v_iteration_compute, [12](#)

P

MDP, [7](#)

pi
MDP, [8](#)

print_MDP

mdp.h, [11](#)

r

MDP, [8](#)

random_MDP
mdp.h, [12](#)
random_example.c, [12](#)
main, [13](#)

release_MDP
mdp.h, [12](#)

ryn_chapter_17.c, [13](#)
main, [14](#)

s

MDP, [8](#)

t

MDP, [8](#)

v

MDP, [8](#)

v_iteration_compute
mdp.h, [12](#)