Package 'ReinforcementLearning'

October 12, 2022

Type Package

Title Model-Free Reinforcement Learning

Version 1.0.5

Date 2020-03-02

Maintainer Nicolas Proellochs <nicolas.proellochs@wi.jlug.de>

Description
Performs model-free reinforcement learning in R. This implementation enables the learning of an optimal policy based on sample sequences consisting of states, actions and rewards. In addition, it supplies multiple predefined reinforcement learning algorithms, such as experience replay. Methodological details can be found in Sutton and Barto (1998) <ISBN:0262039249>.

License MIT + file LICENSE

Depends R (>= 3.2.0)

Imports ggplot2, hash (>= 2.0), data.table

Suggests testthat, knitr, rmarkdown

LazyData TRUE

Encoding UTF-8

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Nicolas Proellochs [aut, cre],
Stefan Feuerriegel [aut]

Repository CRAN

Date/Publication 2020-03-02 06:00:15 UTC

R topics documented:

`computePolicy` *Computes the reinforcement learning policy*

Description

Computes reinforcement learning policy from a given state-action table Q. The policy is the decision-making function of the agent and defines the learning agent's behavior at a given time.

Usage

```
computePolicy(x)
```

Arguments

`x`	Variable which encodes the behavior of the agent. This can be either a `matrix`, `data.frame` or an `rl` object.

Value

Returns the learned policy.

See Also

`ReinforcementLearning`

Examples

```
# Create exemplary state-action table (Q) with 2 actions and 3 states
Q <- data.frame("up" = c(-1, 0, 1), "down" = c(-1, 1, 0))

# Show best possible action in each state
computePolicy(Q)
```

epsilonGreedyActionSelection *Performs ε-greedy action selection*

Description

Deprecated. Please use [ReinforcementLearning::selectEpsilonGreedyAction()] instead.

Usage

```
epsilonGreedyActionSelection(Q, state, epsilon)
```

Arguments

Q	State-action table of type hash.
state	The current state.
epsilon	Exploration rate between 0 and 1.

Value

Character value defining the next action.

References

Sutton and Barto (1998). "Reinforcement Learning: An Introduction", MIT Press, Cambridge, MA.

experienceReplay *Performs experience replay*

Description

Deprecated. Please use [ReinforcementLearning::replayExperience()] instead.

Usage

```
experienceReplay(D, Q, control, ...)
```

Arguments

D	A dataframe containing the input data for reinforcement learning. Each row represents a state transition tuple (s,a,r,s_new).
Q	Existing state-action table of type hash.
control	Control parameters defining the behavior of the agent.
...	Additional parameters passed to function.

Value

Returns an object of class hash that contains the learned Q-table.

References

Lin (1992). "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching", Machine Learning (8:3), pp. 293–321.

Watkins (1992). "Q-learning". Machine Learning (8:3), pp. 279–292.

See Also

ReinforcementLearning

---

gridworldEnvironment *Defines an environment for a gridworld example*

Description

Function defines an environment for a 2x2 gridworld example. Here an agent is intended to navigate from an arbitrary starting position to a goal position. The grid is surrounded by a wall, which makes it impossible for the agent to move off the grid. In addition, the agent faces a wall between s1 and s4. If the agent reaches the goal position, it earns a reward of 10. Crossing each square of the grid results in a negative reward of -1.

Usage

```
gridworldEnvironment(state, action)
```

Arguments

state	The current state.
action	Action to be executed.

Value

List containing the next state and the reward.

Examples

```
# Load gridworld environment
gridworld <- gridworldEnvironment

# Define state and action
state <- "s1"
action <- "down"

# Observe next state and reward
gridworld(state, action)
```

lookupActionSelection *Converts a name into an action selection function*

Description

Input is a name for the action selection, output is the corresponding function object.

Usage

```
lookupActionSelection(type)
```

Arguments

type	A string denoting the type of action selection. Allowed values are `epsilon-greedy` or `random`.

Value

Function that implements the specific learning rule.

lookupLearningRule *Loads reinforcement learning algorithm*

Description

Decides upon a learning rule for reinforcement learning. Input is a name for the learning rule, while output is the corresponding function object.

Usage

```
lookupLearningRule(type)
```

Arguments

type	A string denoting the learning rule. Allowed values are `experienceReplay`.

Value

Function that implements the specific learning rule.

See Also

ReinforcementLearning

policy *Computes the reinforcement learning policy*

Description

Deprecated. Please use [ReinforcementLearning::computePolicy()] instead.

Usage

```
policy(x)
```

Arguments

x	Variable which encodes the behavior of the agent. This can be either a matrix, data.frame or an rl object.

Value

Returns the learned policy.

See Also

ReinforcementLearning

randomActionSelection *Performs random action selection*

Description

Deprecated. Please use [ReinforcementLearning::selectRandomAction()] instead.

Usage

```
randomActionSelection(Q, state, epsilon)
```

Arguments

Q	State-action table of type hash.
state	The current state.
epsilon	Exploration rate between 0 and 1 (not used).

Value

Character value defining the next action.

`ReinforcementLearning` *Performs reinforcement learning*

Description

Performs model-free reinforcement learning. Requires input data in the form of sample sequences consisting of states, actions and rewards. The result of the learning process is a state-action table and an optimal policy that defines the best possible action in each state.

Usage

```
ReinforcementLearning(data, s = "s", a = "a", r = "r",
  s_new = "s_new", learningRule = "experienceReplay", iter = 1,
  control = list(alpha = 0.1, gamma = 0.1, epsilon = 0.1), verbose = F,
  model = NULL, ...)
```

Arguments

`data`	A dataframe containing the input sequences for reinforcement learning. Each row represents a state transition tuple `(s,a,r,s_new)`.
`s`	A string defining the column name of the current state in `data`.
`a`	A string defining the column name of the selected action for the current state in `data`.
`r`	A string defining the column name of the reward in the current state in `data`.
`s_new`	A string defining the column name of the next state in `data`.
`learningRule`	A string defining the selected reinforcement learning agent. The default value and only option in the current package version is `experienceReplay`.
`iter`	(optional) Iterations to be done. iter is an integer greater than 0. By default, `iter` is set to 1.
`control`	(optional) Control parameters defining the behavior of the agent. Default: `alpha = 0.1`; `gamma = 0.1`; `epsilon = 0.1`.
`verbose`	If true, progress report is shown. Default: `false`.
`model`	(optional) Existing model of class `rl`. Default: `NULL`.
`...`	Additional parameters passed to function.

Value

An object of class `rl` with the following components:

`Q` Resulting state-action table.

`Q_hash` Resulting state-action table in `hash` format.

`Actions` Set of actions.

`States` Set of states.

`Policy` Resulting policy defining the best possible action in each state.

`RewardSequence` Rewards collected during each learning episode in `iter`.

`Reward` Total reward collected during the last learning iteration in `iter`.

References

Sutton and Barto (1998). Reinforcement Learning: An Introduction, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA.

Examples

```
# Sampling data (1000 grid sequences)
data <- sampleGridSequence(1000)

# Setting reinforcement learning parameters
control <- list(alpha = 0.1, gamma = 0.1, epsilon = 0.1)

# Performing reinforcement learning
model <- ReinforcementLearning(data, s = "State", a = "Action", r = "Reward",
s_new = "NextState", control = control)

# Printing model
print(model)

# Plotting learning curve
plot(model)
```

`replayExperience` *Performs experience replay*

Description

Performs experience replay. Experience replay allows reinforcement learning agents to remember and reuse experiences from the past. The algorithm requires input data in the form of sample sequences consisting of states, actions and rewards. The result of the learning process is a state-action table Q that allows one to infer the best possible action in each state.

Usage

```
replayExperience(D, Q, control, ...)
```

Arguments

`D`	A `dataframe` containing the input data for reinforcement learning. Each row represents a state transition tuple (`s,a,r,s_new`).
`Q`	Existing state-action table of type `hash`.
`control`	Control parameters defining the behavior of the agent.
`...`	Additional parameters passed to function.

Value

Returns an object of class `hash` that contains the learned Q-table.

References

Lin (1992). "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching", Machine Learning (8:3), pp. 293–321.

Watkins (1992). "Q-learning". Machine Learning (8:3), pp. 279–292.

See Also

ReinforcementLearning

`sampleExperience` *Sample state transitions from an environment function*

Description

Function generates sample experience in the form of state transition tuples.

Usage

```
sampleExperience(N, env, states, actions, actionSelection = "random",
  control = list(alpha = 0.1, gamma = 0.1, epsilon = 0.1),
  model = NULL, ...)
```

Arguments

`N`	Number of samples.
`env`	An environment function.
`states`	A character vector defining the enviroment states.
`actions`	A character vector defining the available actions.
`actionSelection`	(optional) Defines the action selection mode of the reinforcement learning agent. Default: `random`.
`control`	(optional) Control parameters defining the behavior of the agent. Default: `alpha = 0.1`; `gamma = 0.1`; `epsilon = 0.1`.
`model`	(optional) Existing model of class `rl`. Default: `NULL`.
`...`	Additional parameters passed to function.

Value

An `dataframe` containing the experienced state transition tuples `s,a,r,s_new`. The individual columns are as follows:

`State` The current state.

`Action` The selected action for the current state.

`Reward` The reward in the current state.

`NextState` The next state.

See Also

ReinforcementLearning

gridworldEnvironment

Examples

```
# Define environment
env <- gridworldEnvironment

# Define states and actions
states <- c("s1", "s2", "s3", "s4")
actions <- c("up", "down", "left", "right")

# Sample 1000 training examples
data <- sampleExperience(N = 1000, env = env, states = states, actions = actions)
```

---

sampleGridSequence *Sample grid sequence*

Description

Function uses an environment function to generate sample experience in the form of state transition tuples.

Usage

```
sampleGridSequence(N, actionSelection = "random", control = list(alpha
  = 0.1, gamma = 0.1, epsilon = 0.1), model = NULL, ...)
```

Arguments

N	Number of samples.
actionSelection	(optional) Defines the action selection mode of the reinforcement learning agent. Default: random.
control	(optional) Control parameters defining the behavior of the agent. Default: alpha = 0.1; gamma = 0.1; epsilon = 0.1.
model	(optional) Existing model of class rl. Default: NULL.
...	Additional parameters passed to function.

Value

An dataframe containing the experienced state transition tuples s,a,r,s_new. The individual columns are as follows:

State The current state.

`Action` The selected action for the current state.

`Reward` The reward in the current state.

`NextState` The next state.

See Also

`gridworldEnvironment`

`ReinforcementLearning`

`selectEpsilonGreedyAction`
Performs ε-greedy action selection

Description

Implements ε-greedy action selection. In this strategy, the agent explores the environment by selecting an action at random with probability ε. Alternatively, the agent exploits its current knowledge by choosing the optimal action with probability $1 - \varepsilon$.

Usage

```
selectEpsilonGreedyAction(Q, state, epsilon)
```

Arguments

`Q`	State-action table of type `hash`.
`state`	The current state.
`epsilon`	Exploration rate between 0 and 1.

Value

Character value defining the next action.

References

Sutton and Barto (1998). "Reinforcement Learning: An Introduction", MIT Press, Cambridge, MA.

selectRandomAction *Performs random action selection*

Description

Performs random action selection. In this strategy, the agent always selects an action at random.

Usage

```
selectRandomAction(Q, state, epsilon)
```

Arguments

Q	State-action table of type hash.
state	The current state.
epsilon	Exploration rate between 0 and 1 (not used).

Value

Character value defining the next action.

state *Creates a state representation for arbitrary objects*

Description

Converts object of any class to a reinforcement learning state of type character.

Usage

```
state(x, ...)
```

Arguments

x	An object of any class.
...	Additional parameters passed to function.

Value

Character value defining the state representation of the given object.

`tictactoe` *Game states of 100,000 randomly sampled Tic-Tac-Toe games.*

Description

A dataset containing 406,541 game states of Tic-Tac-Toe. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game. All states are observed from the perspective of player X, who is also assumed to have played first.

Usage

```
tictactoe
```

Format

A data frame with 406,541 rows and 4 variables:

State The current game state, i.e. the state of the 3x3 grid.

Action The move of player X in the current game state.

NextState The next observed state after action selection of players X and B.

Reward Indicates terminal and non-terminal game states. Reward is +1 for 'win', 0 for 'draw', and -1 for 'loss'.

Index