Package 'QLearning'

July 21, 2025

Type Package

Title Reinforcement Learning using the Q Learning Algorithm

Version 0.1.1

Author Liam Bressler

Maintainer Liam Bressler <liam.bressler@yale.edu>

Description Implements Q-Learning, a model-free form of reinforcement learning, described in work by Strehl, Li, Wiewiora, Langford & Littman (2006) <doi:10.1145/1143844.1143955>.

License GNU General Public License

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-21 07:59:42 UTC

Contents

`qlearn` *qlearn*

Description

Input a *game* that has variables *statevars* (which the player can keep track of). The player can perform any of *possibleactions*. The output matrix will give the expected value of each action (column) in each state (row).

Usage

```
qlearn(game, statevars, possibleactions, playername="P1",
  numiter=1000, prevstrategy=NULL, ...)
```

Arguments

game	Name of the game to be played/learned.
statevars	A vector of the states to be monitored inside *game*. These are the conditions under which we the player has to make his decision.
possibleactions	A vector of the names of the possible actions inside *game*. This should be a list of every possible action that can be taken, regardless of state.
playername	The name of the variable that holds the name for the player's action inside *game*. See **Details**.
numiter	Number of iterations of *game*. Defaults to 50.
prevstrategy	Reward matrix returned by a previous *qlearn* function; serves as a starting point. Defaults to a blank reward matrix.
...	Additional arguments to be passed to *game*.

Details

At some point in game, there must be a line of the format

`playername <- 'Choose'`

where playername is substituted with the paramater "playername". This line should be at the point where the user wants to have the player choose an action. Since playername defaults to "P1", it sufficient to put the line:

`P1 <- 'Choose'`

somewhere in the function.

Value

A matrix describing the expected reward values of performing a certain action (columns) in a certain state (rows).

Note

Contact at liam.bressler@yale.edu

Author(s)

Liam Bressler

References

http://labressler.github.io/analytics

Examples

```
cardgame <- function()
{
  playercards <- sample(1:8,4) #distribute the cards, we're player one
  ourcard <- playercards[1] #our card
  playertotals <- rep(-1,4) #including the antes
  playersinpot <- vector()
  for (player in 2:4) #other 3 players go first
  {
    if (playercards[player]>=2)
    {
      playertotals[player] <- (-3)
      playersinpot <- append(playersinpot,player)
    }
  }
  #the next line is where we want to choose our action
  player1 <- 'Choose'
  if (player1=="Call")
  {
    playertotals[1] <- (-3)
    playersinpot <- append(playersinpot,1)
  }
  potsize <- -1*(sum(playertotals)) #the amount in the pot is how much the players put in
  playercards[!(1:4 %in% playersinpot)] <- 0 #get rid of everyone who folded
 winner <- which.max(playercards) #winner is the person with the highest card who didn't fold
  playertotals[winner] <- playertotals[winner]+potsize
  return(playertotals[1]) #return how much we won
}

strat <- qlearn(game="cardgame",statevars="ourcard",possibleactions=c("Call","Fold"),
 playername="player1",numiter=25000) #make sure each function and variable name is a string

strat
```

`qlearningaction` *qlearningaction*

Description

This repository implements Q-Learning, a model-free form of reinforcement learning in R.

Usage

```
qlearningaction(q, currentstate, exploration=.5)
```

Arguments

q	Input state/action matrix.
currentstate	Current state of the game. Does not have to match any of the state for q.
exploration	The probability of choosing a random state, rather than the one with the highest EV. Default 0.5.

Details

For internal use for *qlearn*.

Value

An action to take, taken from the possible actions of q.

Note

Contact at liam.bressler@yale.edu

Author(s)

Liam Bressler

References

http://labressler.github.io/analytics

Examples

```
cardgame <- function()
{
  playercards <- sample(1:8,4) #distribute the cards, we're player one
  ourcard <- playercards[1] #our card
  playertotals <- rep(-1,4) #including the antes
  playersinpot <- vector()
  for (player in 2:4) #other 3 players go first
  {
    if (playercards[player]>=2)
    {
      playertotals[player] <- (-3)
      playersinpot <- append(playersinpot,player)
    }
  }
  #the next line is where we want to choose our action
  player1 <- 'Choose'
  if (player1=="Call")
  {
    playertotals[1] <- (-3)
    playersinpot <- append(playersinpot,1)
  }
  potsize <- -1*(sum(playertotals)) #the amount in the pot is how much the players put in
  playercards[!(1:4 %in% playersinpot)] <- 0 #get rid of everyone who folded
```

```
 winner <- which.max(playercards) #winner is the person with the highest card who didn't fold
  playertotals[winner] <- playertotals[winner]+potsize
  return(playertotals[1]) #return how much we won
}

strat <- qlearn(game="cardgame",statevars="ourcard",possibleactions=c("Call","Fold"),
 playername="player1",numiter=25000) #make sure each function and variable name is a string

qlearningaction(strat,3,exploration=.75)
#Pick an action to perform when we have the 3 card, with high exploration
```

`qlearningupdate` *qlearningupdate*

Description

This repository implements Q-Learning, a model-free form of reinforcement learning in R.

Usage

```
qlearningupdate(q, currentstate, currentaction, currentreward, nextstate=NULL,
  rewardcount=.5, gamma=.25)
```

Arguments

`q`	Input state/action matrix.
`currentstate`	Current state of the game. Does not have to match any of the state for *q*.
`currentaction`	Action to take.
`currentreward`	Reward for *currentaction* in current iteration.
`nextstate`	State that the game is in after taking *currentaction*.
`rewardcount`	Regularization constant for reward.
`gamma`	Learning rate constant for Q-Learning.

Details

For internal use for *qlearn*.

Value

An updated state/action matrix.

Note

Contact at liam.bressler@yale.edu

Author(s)

Liam Bressler

References

http://labressler.github.io/analytics

Examples

```
cardgame <- function()
{
  playercards <- sample(1:8,4) #distribute the cards, we're player one
  ourcard <- playercards[1] #our card
  playertotals <- rep(-1,4) #including the antes
  playersinpot <- vector()
  for (player in 2:4) #other 3 players go first
  {
    if (playercards[player]>=2)
    {
      playertotals[player] <- (-3)
      playersinpot <- append(playersinpot,player)
    }
  }
  #the next line is where we want to choose our action
  player1 <- 'Choose'
  if (player1=="Call")
  {
    playertotals[1] <- (-3)
    playersinpot <- append(playersinpot,1)
  }
  potsize <- -1*(sum(playertotals)) #the amount in the pot is how much the players put in
  playercards[!(1:4 %in% playersinpot)] <- 0 #get rid of everyone who folded
 winner <- which.max(playercards) #winner is the person with the highest card who didn't fold
  playertotals[winner] <- playertotals[winner]+potsize
  return(playertotals[1]) #return how much we won
}

strat <- qlearn(game="cardgame",statevars="ourcard",possibleactions=c("Call","Fold"),
 playername="player1",numiter=25000) #make sure each function and variable name is a string

strat <- qlearningupdate(strat,currentstate=7,currentaction="Call",currentreward=5)
#Update the matrix after an example when we call with the 7 card as our state, winning 5 chips
```

Index