# simple_reinforcement_learning_exploration.R

vaishnavimyadam

2020-06-25

```
## Vaishu Myadam (vmyadam1208@gmail.com)
## June, 2020

## Exploring the reinforcement learning package with grid world and tic tac toe (Q-learning)
## Using official documentation

# Necessary libraries:

library(ReinforcementLearning)

# Tic tac toe:

data("tictactoe")
head(tictactoe, 10)
```

```
##        State Action NextState Reward
## 1  .........     c7 ......X.B      0
## 2  ......X.B     c6 ...B.XX.B      0
## 3  ...B.XX.B     c2 .XBB.XX.B      0
## 4  .XBB.XX.B     c8 .XBBBXXXB      0
## 5  .XBBBXXXB     c1 XXBBBXXXB      0
## 6  .........     c1 X...B....      0
## 7  X...B....     c4 X..XB.B..      0
## 8  X..XB.B..     c3 XBXXB.B..      0
## 9  XBXXB.B..     c8 XBXXBBBX.      0
## 10 XBXXBBBX.     c9 XBXXBBBXX      0
```

```
control <- list(alpha = 0.2, gamma = 0.4, epsilon = 0.1)
model <- ReinforcementLearning(tictactoe, s = "State", a = "Action", r = "Reward", s_new = "NextState",
results <- computePolicy(model)
head(results)
```

```
## .XXBB..XB XXBB.B.X. .XBB..BXX BXX...B.. ..XB..... XBXBXB...
##      "c1"      "c5"      "c5"      "c4"      "c5"      "c9"
```

```
summary(model)
```

```
##
## Model details
## Learning rule:           experienceReplay
## Learning iterations:     1
## Number of states:        1893
## Number of actions:       9
## Total Reward:            5449
```

```
##
## Reward details (per iteration)
## Min:                       5449
## Max:                       5449
## Average:                   5449
## Median:                    5449
## Standard deviation:        NA
```

```
# Grid world:

states <- c("s1", "s2", "s3", "s4")
actions <- c("up", "down", "left", "right")
envie <- gridworldEnvironment
print(envie)
```

```
## function (state, action)
## {
##     next_state <- state
##     if (state == state("s1") && action == "down")
##         next_state <- state("s2")
##     if (state == state("s2") && action == "up")
##         next_state <- state("s1")
##     if (state == state("s2") && action == "right")
##         next_state <- state("s3")
##     if (state == state("s3") && action == "left")
##         next_state <- state("s2")
##     if (state == state("s3") && action == "up")
##         next_state <- state("s4")
##     if (next_state == state("s4") && state != state("s4")) {
##         reward <- 10
##     }
##     else {
##         reward <- -1
##     }
##     out <- list(NextState = next_state, Reward = reward)
##     return(out)
## }
## <bytecode: 0x7fd970f750a8>
## <environment: namespace:ReinforcementLearning>
```

```
data <- sampleExperience(N = 1000, env = envie, states = states, actions = actions)
head(data)
```

```
##   State Action Reward NextState
## 1    s3   down     -1        s3
## 2    s1   down     -1        s2
## 3    s4  right     -1        s4
## 4    s3   left     -1        s2
## 5    s4   down     -1        s4
## 6    s1     up     -1        s1
```

```
control <- list(alpha = 0.1, gamma = 0.1, epsilon = 0.1)
model <- ReinforcementLearning(data, s = "State", a = "Action", r = "Reward", s_new = "NextState", conti
computePolicy(model)
```

```
##      s1      s2      s3      s4
```

```
##  "down" "right"    "up" "right"
```

```
summary(model)
```

```
##
## Model details
## Learning rule:           experienceReplay
## Learning iterations:     1
## Number of states:        4
## Number of actions:       4
## Total Reward:            -450
##
## Reward details (per iteration)
## Min:                     -450
## Max:                     -450
## Average:                 -450
## Median:                  -450
## Standard deviation:      NA
```

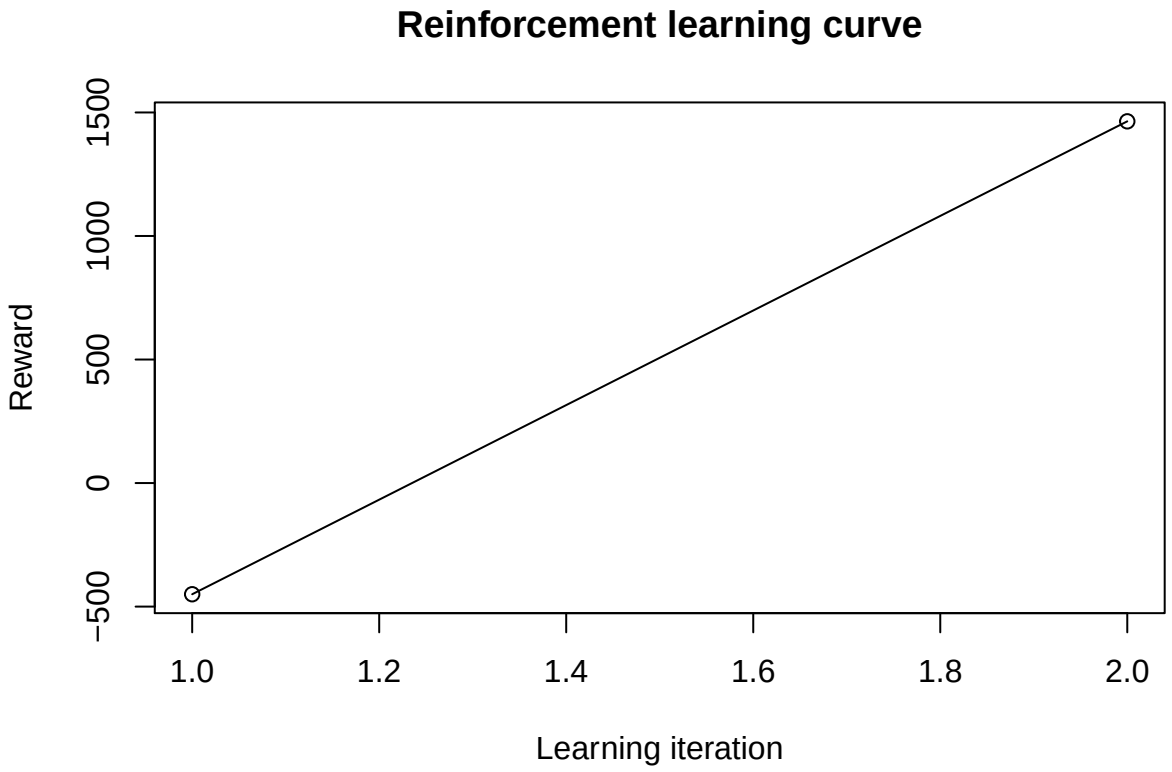
```
# Prediction

data_to_be_predicted <- data.frame(State = c("s1", "s2", "s1"), stringsAsFactors = FALSE)
data_to_be_predicted$OptimalAction <- predict(model, data_to_be_predicted$State)
data_to_be_predicted
```

```
##   State OptimalAction
## 1    s1          down
## 2    s2         right
## 3    s1          down
```

```
# See what happens when you add new data

new_data <- sampleExperience(N = 1000,
                             env = envie,
                             states = states,
                             actions = actions,
                             actionSelection = "epsilon-greedy",
                             model = model,
                             control = control)
modified_model <- ReinforcementLearning(new_data,
                                        s = "State",
                                        a = "Action",
                                        r = "Reward",
                                        s_new = "NextState",
                                        control = control,
                                        model = model)
plot(modified_model)
```


Reinforcement learning curve
Reward
−500
0
500
1000
1500
1.0
1.2
1.4
1.6
1.8
2.0
Learning iteration