SWRhapsody

Tic Tac Toe

于11月 26, 2018由**SWRhapsody**发布

Introduction

刚开学这一个月真是好忙啊，我都没什么时间来写博客。这几天看了点 Reinforcement Learning 相关的知识写个小总结。因为我自己的专业不是这方面所以为了避免瞎讲，我无法在这篇博客涉及太多的概念。大部分概念可以在 Reference 中看。

Code

Tic Tac Toe!

Reinforcement Learning是机器学习的一部分，这篇博文的目标是用 Reinfocement Learning 实现个玩 Tic Tac Toe 的游戏AI，完整的代码[7]，我大部分的代码都是参照[2]。

我们先写个 Board 来作为一个游戏场地

```
import numpy as np
import pandas as pd

class Game:
    def __init__(self, row, col):
        self.row_size = row
        self.col_size = col
        # 0 for not end
        # 1 for winer is first player
        # 2 for winer is second player
        # 3 for tie
        self.end = 0
        self.board = np.zeros((self.row_size, self.col_size))
        self.size = self.board.size
        self.mapping = {0: " ", 1: "X", -1: "O"}


    def DrawCharForItem(self, item):
        # 0 for ' '
        # 1 for 'X'
        # -1 for 'O'
        return self.mapping[item]
```

```
        board =
             {} | {} | {}
            -----------
             {} | {} | {}
            -----------
             {} | {} | {}
        """.format(*items)
        print(board)

    def CheckWin(self):
        result = []

        for i in range(self.row_size):
            result.append(np.sum(self.board[i, :]))
        for i in range(self.col_size):

            result.append(np.sum(self.board[:, i]))

        result.append(0)
        for i in range(0, self.col_size):
            result[-1] += self.board.item((i, i))
        result.append(0)
        for i in range(0, self.row_size):
            result[-1] += self.board.item((i, self.row_size - 1 - i))

        for i in result:
            if i == 3:
                self.end = 1
                return self.end
            if i == -3:
                self.end = 2
                return self.end
        sum = np.sum(np.abs(self.board))
        if sum == self.size:
            self.end = 3
        return self.end

    def End(self):
        return self.end

    def Step(self, order, action):
        if action[0] < 0 or action[0] > self.row_size-1 or action[1] < 0 or action[1] > self.c
            print("x or y invalid")
            return True
        if self.board[action[0], action[1]] != 0:
            print("this palce has already been taken")
            return True
        self.board[action[0], action[1]] = order
        self.CheckWin()
        self.Print()
        return False
```

接下来写个Agent类作为我们的AI。

```
class Agent
```

现在来介绍下我们这个AI的核心公式（[8]）

SWRhapsody

不过在编码这公式之前我们需要一个数据结构来存储我们的收益。

定义一个 state 来存储收益。这个 state 是一个 row*col 的数组，这个数组存放当前情况下每一格的收益。

```
state，当前情况下走 (0,1) 这个点的收益是 1.30938546e-07
[[ 1.30938546e-07 -4.93034771e-07  1.51138875e-04]
 [-6.41862293e-08 -2.77494609e-08 -5.18934185e-07]
 [-2.61877061e-07 -4.96284826e-05  2.59479593e-07]]
```

我们在定义一个字典来存每一种情况下的 state，为每一种情况生成一个hash，这个hash作为字典的key

```
def ComputeHash(board):
    board_ = board.flatten().tolist()
    return "".join([str(i) for i in board_])

class Agent:
    def __init__(self, name, exploration_rate=0.33, learning_rate=0.5, discount_factor=0.01):
        self.states = {}
        # states stack
        self.state_order = []
        self.learning_rate = learning_rate
        self.exploration_rate = exploration_rate
        self.discount_factor = discount_factor
        self.name = name
```

这样我们的公式就变成了这样

```
class Agent:
    def learn_by_temporal_difference(self, reward, new_state_hash, state_hash):
        prev_state = self.states.get(
            state_hash, np.zeros((BOARD_ROWS, BOARD_COLS)))
        return self.learning_rate*((reward*self.states[new_state_hash] )- prev_state)
```

接着定义一个reward函数，对于什么时候给予奖励有多种选择[9]，在这个AI中我们选择在出现赢家的时候给予奖励。

```
class Agent:
    def set_state(self, board, action):
        hash = ComputeHash(board)
        self.state_order.append((hash, action))

    def OnReward(self, reward):
        hash, action = self.state_order.pop()
        if hash not in self.states:
            self.states[hash] = np.zeros((BOARD_ROWS, BOARD_COLS))
        self.states[hash].itemset(action,reward)

        while self.state_order:
            hash_prev, action_prev = self.state_order.pop()
            reward *= self.discount_factor
```

```
        else:
            self.states[hash_prev] = np.zeros((BOARD_ROWS, BOARD_COLS))
            reward = self.learn_by_temporal_difference(reward, hash, hash_prev).item(actio
            self.states[hash_prev].itemset(action,reward)

        hash = hash_prev
        action = action_prev
```

这里 state_order 是我们存所走过的每一步的一个栈，存的是 (state的hash，这一步的坐标)。这个函数的组要目的是把计算出的收益存储在 self.states 中。

接下来我们定义 exploit 和 explore 函数

```
class Agent:
    def exploit(self, board):
        state_value = self.states[ComputeHash(board)]
        x, y = np.where(state_value == state_value.max())
        best_choices = [(a, b) for a, b in zip(x, y)]
        return best_choices[np.random.choice(len(best_choices))]

    def explore(self, board):
        x, y = np.where(board == 0)
        vacant = [(a, b) for a, b in zip(x, y)]
        return vacant[np.random.choice(len(vacant))]
```

最后定义个用来选择走那一步的函数，何时选择探索具体可以参考[5]中的2.2

```
class Agent:
    def SelectMove(self, board):
        action = None
        exploration = np.random.random() < self.exploration_rate
        if exploration or hash not in self.states:
            print("%s exploit" % self.name)
            action = self.explore(board)
        else:
            print("%s exploit" % self.name)
            action = self.exploit(board)
        # update state
        self.set_state(board, action)
        return action
```

训练下这个AI

```
def Train(round, bot1, bot2):
    win_trace = pd.DataFrame(data=np.zeros(
        (round, 2)), columns=["bot1", "bot2"])
    for i in range(round):
        print("-"*100)
        print("Round:{}".format(i+1))
        game = Game(BOARD_ROWS, BOARD_COLS)
        turn = 1
        while game.End() == 0:
            if turn == 1:
                action = bot1.SelectMove(game.board)
                game.Step(1, action)
                turn = 2
            else:
                action = bot2.SelectMove(game.board)
```

SWRhapsody

```
            bot2.OnReward(-1)
            win_trace.set_value(i, 'bot1', 1)
        elif game.End() == 2:
            bot1.OnReward(-1)
            bot2.OnReward(1)
            win_trace.set_value(i, 'bot2', 1)
    return win_trace

if __name__ == "__main__":
    bot1 = Agent("bot1")
    bot2 = Agent("bot2")
    Train(5000, bot1, bot2)
```

最后跟它玩下

```
Bot first!
bot1 exploit

               |   | X
            -----------
               |   |
            -----------
               |   |

Your turn(enter x,y):1,1

               |   | X
            -----------
               | O |
            -----------
               |   |

bot1 exploit

               |   | X
            -----------
               | O |
            -----------
               |   | X

Your turn(enter x,y):1,2

               |   | X
            -----------
               | O | O
            -----------
               |   | X

bot1 exploit

               |   | X
            -----------
             X | O | O
            -----------
               |   | X

Your turn(enter x,y):0,1

               | O | X
            -----------
```

```

             | O | X
          -----------
           X | O | O
          -----------
           X |   | X

Your turn(enter x,y):2,1

             | O | X
          -----------
           X | O | O
          -----------
           X | O | X

You win
```

尝试了比较多的次数，只有一次训练出来的智商过关(结果忘保存结果)。感觉训练的次数不够并且需要把规则教给AI，不然训练的速度太慢。

Reference

[1] https://en.wikipedia.org/wiki/Reinforcement_learning

[2] https://github.com/AmreshVenugopal/tic_tac_toe/blob/master/Tic%20tac%20toe.ipynb

[3] https://github.com/ShangtongZhang/reinforcement-learning-an-introduction/blob/master/chapter01/tic_tac_toe.py

[4] https://en.wikipedia.org/wiki/Temporal_difference_learning

[5] http://tokic.com/www/tokicm/publikationen/papers/AdaptiveEpsilonGreedyExploration.pdf

[6] https://detailed.af/a-game-of-tic-tac-toe/

[7] https://github.com/Iceware/blog_code/blob/master/2018/tic_tac_toe.py

[8] http://incompleteideas.net/book/the-book.html

[9] https://medium.freecodecamp.org/an-introduction-to-reinforcement-learning-4339519de419

分类: UNCATEGORIZED

SWRhapsody

0 条评论

发表评论

名称 *

电子邮件 *

网站

在想些什么?

发表评论

近期文章

携程Apollo YAML 反序列化

CVE-2020-5410

CodeQL部分源码简读

服务器与网关的不一致

SWRhapsody

近期评论

文章归档

- 2020年8月
- 2020年6月
- 2020年5月
- 2020年3月
- 2020年1月
- 2019年12月
- 2019年11月
- 2019年8月
- 2019年7月
- 2019年5月
- 2019年4月
- 2019年1月
- 2018年11月
- 2018年10月
- 2018年9月
- 2018年4月
- 2018年3月
- 2018年2月
- 2018年1月

分类目录

SWRhapsody

cryptography

Exercise

Exploit

HackTheBox

Penetration Test

Uncategorized

相关文章

UNCATEGORIZED

WordPress Social Warfare XSS

Introduction Plugin name: Social Wa 阅读更多…

UNCATEGORIZED

自动化程序修复

Automatic patch generation Introduc 阅读更多…

SWRhapsody

Criminology

Introduction 网络安全也要学习犯罪学了吗，每次写这类文章都 阅读更多…

ABOUT

Hestia |由ThemeIsle开发