强化学习

Reinforcement

Learning

#7 Sarsa 思维决策 (强化学习 Reinforcement Learning 教学)

切换成 优酷 视频

(Chrome无法播放优酷? 网址框输入"chrome://settings/content/", 勾选允许 Flash Player. 实在不行? 请 点击这里)

作者: **Morvan** 编辑: **Morvan**

- 学习资料:
  - 全部代码
  - 什么是 Sarsa 短视频
  - 本节内容的模拟视频效果Youtube, 优酷
  - 学习书籍 Reinforcement learning: An introduction

接着上节内容, 我们来实现 RL_brain 的 SarsaTable 部分, 这也是 RL 的大脑部分, 负责决策和思考.

本节内容包括:

- 代码主结构
- 学习

代码主结构

和之前定义 Qlearning 中的 QLearningTable 一样, 因为使用 tabular 方式的 Sarsa 和 Qlearning 的相似度极高,

```
class SarsaTable:
    # 初始化 (与之前一样)
    def __init__(self, actions, learning_rate=0.01, reward_decay=0.9, e_greedy=0.9):

    # 选行为 (与之前一样)
    def choose_action(self, observation):

    # 学习更新参数 (有改变)
    def learn(self, s, a, r, s_):

    # 检测 state 是否存在 (与之前一样)
    def check_state_exist(self, state):
```

我们甚至可以定义一个 主class RL , 然后将 QLearningTable 和 SarsaTable 作为 主class RL 的衍生, 这个主 RL 可以这样定义. 所以我们将之前的 __init__ , check_state_exist , choose_action , learn 全部都放在这个主结构中, 之后根据不同的算法更改对应的内容就好了. 所以还没弄懂这些功能的朋友们, 请回到之前的教程再看一遍.

```
import numpy as np
import pandas as pd


class RL(object):
    def __init__(self, action_space, learning_rate=0.01, reward_decay=0.9, e_greedy=0.9):
        ... # 和 QLearningTable 中的代码一样

    def check_state_exist(self, state):
        ... # 和 QLearningTable 中的代码一样
```

```
    def learn(self, *args):
        pass # 每种的都有点不同, 所以用 pass
```

如果是这样定义父类的 RL class, 通过继承关系, 那之子类 QLearningTable class 就能简化成这样:

```
class QLearningTable(RL):  # 继承了父类 RL
    def __init__(self, actions, learning_rate=0.01, reward_decay=0.9, e_greedy=0.9):
        super(QLearningTable, self).__init__(actions, learning_rate, reward_decay, e_greedy)  # 表示继承关系

    def learn(self, s, a, r, s_):  # learn 的方法在每种类型中有不一样, 需重新定义
        self.check_state_exist(s_)
        q_predict = self.q_table.ix[s, a]
        if s_ != 'terminal':
            q_target = r + self.gamma * self.q_table.ix[s_, :].max()
        else:
            q_target = r
        self.q_table.ix[s, a] += self.lr * (q_target - q_predict)
```

学习

有了父类的 RL , 我们这次的编写就很简单, 只需要编写 SarsaTable 中 learn 这个功能就完成了. 因为其他功能都和父类是一样的. 这就是我们所有的 SarsaTable 于父类 RL 不同之处的代码. 是不是很简单.

```
class SarsaTable(RL):  # 继承 RL class

    def __init__(self, actions, learning_rate=0.01, reward_decay=0.9, e_greedy=0.9):
        super(SarsaTable, self).__init__(actions, learning_rate, reward_decay, e_greedy)  # 表示继承关系

    def learn(self, s, a, r, s_, a_):
        self.check_state_exist(s_)
        q_predict = self.q_table.ix[s, a]
        if s_ != 'terminal':
            q_target = r + self.gamma * self.q_table.ix[s_, a_]  # q_target 基于选好的 a_ 而不是 Q(s_) 的最大值
        else:
            q_target = r  # 如果 s_ 是终止符
        self.q_table.ix[s, a] += self.lr * (q_target - q_predict)  # 更新 q_table
```

如果想一次性看到全部代码, 请去我的 Github

如果你觉得这篇文章或视频对你的学习很有帮助, 请你也分享它, 让它能再次帮助到更多的需要学习的人.

莫烦没有正式的经济来源, 如果你也想支持 莫烦**Python** 并看到更好的教学内容, 请拉倒屏幕最下方, 赞助他一点点, 作为鼓励他继续开源的动力.

撰写评论

使用社交网站账户登录 或使用来必力便捷评论

邮件

写评论

总评论数 6 按时间正序

2017年6月3日

maze_env.py的Maze类少继承object了。

1 0 0

2017年6月3日

哦，我用的是python2，python3会隐式继承，请忽略我。。。。。

0 0 0

NO NICKNAME 2017年5月15日 · 修改完成

learn函数中，self.q_table.ix[s_, a_]是等于self.q_table.ix[s_, :].max()的，所以learn函数没有变化，唯一变化的是next_action不是下次得出来的

1 0 0

2017年5月15日

@NO NICKNAME 我可否这样理解，如果以Q表查询为基础的话, QLearning和SARSA 效果是一样的。（如果我理解有误的话，能否给一个例子在说明2者的差异，谢谢.)

0 0 0

2017年5月15日

Q_Learning 中，q_target 的取值是s_名下的所有a_的最大值。

q_target = r + self.gamma * self.q_table.ix[s_, :].max()

SARAS 中，尽管q_target = r + self.gamma * self.q_table.ix[s_, a_]
形式不同，但是，a_是事先按照argmax取得动作。不是和Q_Learning 效果等同了吗？

从效果上来看，好像确实有不一样的地方。我的理解哪里出问题了？

1 0 0

莫烦Python 2017年5月16日

SARSA 中的 a_在选取的时候我们还经过了一次 greedy Policy 的过程, 比如90% 选 argmax, 10%随机选, 所以 a_ 在 SARSA 更新的时候就不一定是 Q_max 的 a_

0 0 0

来必力是？ | 询问

支持 让教学变得更优秀

点我 赞助 莫烦

关注我的动向:

Youtube频道 优酷频道 Github 微博

Email: morvanzhou@hotmail.com

© 2016 morvanzhou.github.io. All Rights Reserved