蒙特卡洛树搜索算法（UCT）：一个程序猿进化的故事

SNYang 2016-10-26 原文

前言：

本文是根据的文章Introduction to Monte Carlo Tree Search by Jeff Bradberry所写。

Jeff Bradberry还提供了一整套的例子，用python写的。

board game server

board game client

Tic Tac Toe board

AI implementation of Tic Tac Toe

阿袁工作的第一天 - 蒙特卡罗树搜索算法 - 游戏的通用接口board 和 player

阿袁看到阿静最近在学习蒙特卡罗树搜索算法。急忙凑上去问：“蒙特卡罗树搜索算法是干什么用的？”

"蒙特卡罗树搜索算法是一种方法（或者说框架），用于解决完美信息博弈。我现在学习一个蒙特卡罗树搜索算法的变种：UCT算法，用于提供一种通用的游戏对弈解决算法。"

注: perfect information games (完美信息)博弈，指的是没有任何信息被隐藏的游戏。

"通用的游戏对弈算法，是对任何游戏都有效，是吗？"

"简单的说，是这样的。重要的一点是，算法并**不用了解**游戏的**领域知识**。"

"领域知识？不是很好理解。难道连游戏规则也不知道，就可以赢吗？"

"游戏的领域知识。举个例子，国际象棋中每个棋子的子力，比如皇后的子力是10，车是5等等。这些就是领域知识。在通用的情况下，马的走法-这样的规则，也算是领域知识。"

"有点糊涂了！AI算法该如何下子呢？"

"用面向对象的逻辑来说，我们可以给游戏定义有一个通用接口(board)，具体的游戏只能实现这个接口，不能提供其它的信息。"

"对于程序猿来说，这就容易理解多了。我们可以先看看这个接口(board)，都应该定义什么样属性和方法。"

"首先，有一个num_players属性，返回游戏的玩家数。"

"嗯，让我想想，游戏开始的时候，需要一个方法start，启动一个游戏。"

"很好，这个方法需要返回一个state对象，用于记录游戏当前的状态。state对象的内容，外部是不可知的。使用board自己可以解释。"

"然后，需要显示棋盘的状态。这样，board就需要提供一个display方法，返回当前的状态或者是棋盘状态。"

"对。应该有个方法返回谁是该下子的玩家:current_player."

"当前玩家是一个AI玩家（也就是对弈算法的使用者），怎么知道如何下子呢？这里需要许多的领域知识吧？"

"一个技巧是让board根据历史的状态列表，返回当前允许的所有下法：legal_actions。"

"再加上一个is_legal(action)，来判断一个下法是否合适。"

"下来应该是根据现在的action，返回下一个游戏状态，next_state。"

"为了判断胜负，需要一个winner方法。"

"如果有了赢家，board需要返回一个winner_message信息。通知玩家谁胜了。"

"看起来不错！我们总结一下board接口的内容。"

```
class Board(object):
    '''
    Define general rules of a game.
    State: State is an object which is only be used inside the board class.
        Normally, a state include game board information (e.g. chessmen positions, action index, current action, current player, etc.)
    Action: an object to describe a move.
    '''

    '''
    num_players: The player numbers of the board.
    '''
    num_players = 2

    def start(self):
        '''
        Start the game
        Return: the initial state
        '''
        return None

    def display(self, state, action, _unicode=True):
        '''
        Dispaly the board
        state: current state
        action: current action
        Return: display information
        '''
        return None

    def parse(self, action):
        '''
        Parse player input text into an action.
```

```
        If the input action is invalid, return None.
        The method is used by a human player to parse human input.
        action: player input action texxt.
        Return: action if input is a valid action, otherwise None.
        '''
        return None

    def next_state(self, state, action):
        '''
        Calculate the next state base on current state and action.
        state: the current state
        action: the current action
        Return: the next state
        '''
        return tuple(state)

    def is_legal(self, history, action):
        '''
        Check if an action is legal.
        The method is used by a human player to validate human input.
        history: an array of history states.
        Return: ture if the action is legal, otherwise return false.
        '''
        return (R, C) == (state[20], state[21])

    def legal_actions(self, history):
        '''
        Calculate legal action from history states.
        The method is mainly used by AI players.
        history: an array of history states.
        Return: an array of legal actions.
        '''
        return actions

    def current_player(self, state):
        '''
        Gets the current player.
        state: the current state.
        Return: the current player number.
```

```
        '''
        return None

    def winner(self, history):
        '''
        Gets the win player.
        history: an array of history states.
        Return: win player number. 0: no winner and no end, players numbers + 1: draw.
        '''
        return 0

    def winner_message(self, winner):
        '''
        Gets game result.
        winner: win player number
        Return: winner message, the game result.
        '''
        return ""
```

"另外，我们需要定义一个player接口，玩家主要是下子，所以需要一个get_action方法。"

"当一个玩家下完子后，需要通过一个update方法通知所有的玩家，状态要更新了。"

```
class Player(object):
    def update(self, state):
        '''
        Update current state into all states.
        state: the current state.
        '''
        self.states.append(state)

    def display(self, state, action):
        '''
        Display board.
        state: the current state.
        action: the current action.
        Return: display information.
        '''
```

```
16.         return self.board.display(state, action)
17.
18.     def winner_message(self, msg):
19.         '''
20.         Display winner message.
21.         msg: winner infomation
22.         Return: winner message
23.         '''
24.         return self.board.winner_message(msg)
25.
26.     def get_action(self):
27.         '''
28.         Get player next action.
29.         Return: the next action.
30.         '''
31.         return action
```

注：方法: diplay and winner_message用于向游戏的客户端提供board的信息。这样隔离了客户端和board。

阿袁工作的第2天 - 蒙特卡罗树搜索算法 - MonteCarlo Player

阿袁和阿静继续关于蒙特卡罗树搜索算法的讨论。

阿静说道，“在编写一个人工智能游戏对弈的应用中，至少需要两个具体的player，一个是human player，一个是MonteCarlo player。”

"human player向人类玩家提供了一个交互界面。"

“对，MonteCarlo player是一个AI player，也是我们要讨论的重点，MonteCarlo player在实现get_action中，通过board，模拟后面可能下法；并根据模拟的结果，获得一个最优的下法。”

"我们先从一个简单的问题开始：一个游戏下法的组合可能是一个很大的数，我们如何控制这个模拟行为是满足一定时间上的限制的。"

“对于这个问题，解决方法有一些。这里，我们允许一个参数calculation_time来控制时间。每次模拟一条路径，模拟完后，检测一下是否到时。”

“一条路径就是从游戏的当前状态到对局结束的所有步骤。如果这些步骤太长了呢？”

“尽管游戏的下法组合数会很大。但是一个游戏的正常步骤却不会很大哦。我们也可以通过另外一个参数max_actions来控制。”

“明白了。代码大概是这个样子。”

```
class MonteCarlo(object):

    def __init__(self, board, **kwargs):
        # ...

        self.calculation_time = float(kwargs.get('time', 30))
        self.max_actions = int(kwargs.get('max_actions', 1000))

        # ...

    def get_action(self):
        # ...

        # Control period of simulation
        moves = 0
        begin = time.time()
        while time.time() - begin < self.calculation_time:
            self.run_simulation()
            moves += 1

        # ...

    def run_simulation(self):
        # ...

        # Control number of simulation actions
        for t in range(1, self.max_actions + 1):
            # ...

        # ...
```

注：为了易于理解，我简单地重构了源代码，主要是rename了一些变量名。

"今天时间有些紧张，明天我们讨论蒙特卡罗树搜索的步骤"

阿袁工作的第3天 - 蒙特卡罗树搜索 - 蒙特卡罗树搜索的步骤

阿袁昨天晚上，也好好学习了蒙特卡罗树搜索。今天，他开始发言。

"蒙特卡罗树搜索是一个方法，应该是来自于蒙特卡罗方法。这个方法定义了几个步骤，用于找到最优的下法。"

“严格的说，蒙特卡罗树搜索并不是一个算法。”

“是的。所以蒙特卡罗树搜索有很多变种，我们现在学习的算法是蒙特卡罗树搜索算法的一个变种：**信任度上限树**(Upper Confidence bound applied to Trees(UCT))。这个我们明天研究。”

“好，今天主要了解**蒙特卡罗树搜索方法的步骤**”

"从文章上看一共有四个步骤。"

"是的。分别是选举(selection)，扩展(expansion)，模拟(simulation)，反向传播(Back-Propagation)。"

“我们看看这张图。绿色部分是蒙特卡罗树搜索的四个步骤。”

Monte Carlo Tree Search StepsMonte Carlo Tree Search Stepscluster0Loop: limit simulation period time. One loop one path.cluster0Loop: limit max actions. One loop one action.StartStartreach_time_limiattionReach time limiattion?Start->reach_time_limiattionEndEndloop_meet_max_actionsMeet max actions?reach_time_limiattion->loop_meet_max_actionsnoselect_best_actionSelect the best action and returnreach_time_limiattion->select_best_actionyesback_propagationBack-Propagationback_propagation->reach_time_limiattionloop_meet_max_actions->back_propagationyesget_children_actionsGet children actionsloop_meet_max_actions->get_children_actionsnomeet_selection_criteriaMeet selection criteria?get_children_actions->meet_selection_criteriaselectionSelectionmeet_selection_criteria->selectionyesexpansionExpansionmeet_selection_criteria->expansionnosimulationSimulationselection->simulationexpansion->simulationhas_winnerHas Winner?simulation->has_winnerhas_winner->back_propagationyeshas_winner->loop_meet_max_actionsnoselect_best_action->End

“**选举(selection)**是根据当前获得所有子步骤的统计结果，选择一个最优的子步骤。”

“**扩展(expansion)**在当前获得的统计结果不足以计算出下一个步骤时，随机选择一个子步骤。”

“**模拟(simulation)**模拟游戏，进入下一步。”

“**反向传播(Back-Propagation)**根据游戏结束的结果，计算对应路径上统计记录的值。”

“从上面这张图可以看出，选举的算法很重要，这个算法可以说是来评价每个步骤的价值的。”

“好了。今天，我们了解了蒙特卡罗树搜索的步骤。”

“明天，可以学习Upper Confidence bound applied to Trees(UCT) - 信任度上限树算法。”

阿袁工作的第4天 - 蒙特卡罗树搜索 - Upper Confidence bound applied to Trees(UCT) - 信任度上限树算法

一开始，阿静就开始讲到。

“信任度上限树算法UCT是根据统计学的信任区间公式，来计算一个步骤的价值。这个方法比较简单，只需要每个步骤的访问数和获胜数就可以了。”

“信任区间公式的是什么呢？”

阿静写下信任区间公式。

置信区间(confidence intervals)

$$
\bar{x}_i \pm \sqrt{\frac{z\ln{n}}{n_i}} \\
where: \\
\qquad \bar{x}_i \text{ : the mean of choose i.} \\
\qquad n_i \text{ : the number of plays of choose i.} \\
\qquad n \text{ : the total number of plays.} \\
\qquad z \text{ : 1.96 for 95% confidence level.}
$$

阿静进一步解释道。

“置信区间是一个统计上的计算值，如果z使用1.96，可以使置信区间的置信度达到95%。也就是说：有95%的信心，样本的平均值在置信区间内。”

“UCT算法使用了置信区间的**上限值**做为每个步骤的**价值**。”

“使用**置信区间的上限值**带来的一个好处是：如果当前选择的最优子步骤在多次失败的模拟后，这个值会变小，从而导致另一个同级的子步骤可能会变得更优。”

“另外一个关键点是**选举的条件**，文章中的选举条件是当前所有子步骤都有了统计记录（也就是至少访问了一次，有了访问数。）。”

阿袁工作的第5天 - 蒙特卡罗树搜索 - 图形化模拟 Upper Confidence bound applied to Trees(UCT) - 信任度上限树算法

阿袁今天做了一天功课，画了一些图来说明UCT算法的过程。

- 首先，初始状态下，所有的子步骤都没有统计数据。

Monte Carlo Tree Search Steps - Initialize StateMonte Carlo Tree Search Steps - Initialize State No statistics records for all children actions.L0CL1_1L0->L1_1L1_2L0->L1_2L1_3L0->L1_3L1_4L0->L1_4

- 所以，先做**扩展(Expansion)**，随机选择一个子步骤，不停的**模拟(Simulation)**，直到游戏结束。然后**反向传播(Back-Propagation)**，记录扩展步骤的统计数据。

Monte Carlo Tree Search StepsMonte Carlo Tree Search Steps - ExpansionL0CL1_1L0->L1_1L1_20/1L0->L1_2L1_3L0->L1_3L1_4L0->L1_4L1_2_1L1_2->L1_2_1L1_2_1_1L1_2_1->L1_2_1_1L1_2_1_1_1LoseL1_2_1_1->L1_2_1_1_1

- 多次**扩展(Expansion)**之后，达到了**选举(selection)**的条件，开始**选举(selection)**，选出最优的一个子步骤。

Monte Carlo Tree Search Steps - SelectionMonte Carlo Tree Search Steps - Selection After some expansions, all children actions are recorded. Select the one with max win rate.L0CL1_12/5L0->L1_1L1_23/4L0->L1_2L1_30/1L0->L1_3L1_44/6L0->L1_4

- 继续**扩展(Expansion)**，**模拟(Simulation)**，**反向传播(Back-Propagation)**
  下图说明以前最优的子步骤，可能在多次扩展后，发生变化。

Monte Carlo Tree Search Steps - More Expansion, Simulation and Back-PropagationMonte Carlo Tree Search Steps - More Expansion, Simulation and Back-Propagation Would lead the best action is changed to another one.L0CL1_12/5L0->L1_1L1_23/5L0->L1_2L1_30/1L0->L1_3L1_44/6L0->L1_4L1_2_10/1L1_2->L1_2_1L1_2_1_1L1_2_1->L1_2_1_1L1_2_1_1_1LoseL1_2_1_1->L1_2_1_1_1

阿袁的日记

2016年10月X日 星期六

这周和阿静一起学习了蒙特卡罗树搜索的一些知识。基本上了解了蒙特卡罗树搜索的步骤和使用方法。

发现在使用蒙特卡罗树搜索方法中，有许多可以优化的地方。比如：

- 步骤价值计算

  - 是否可以在没有赢的情况下，计算价值？
  - 是否可以计算一个步骤是没有价值的，因而可以及早的砍掉它。

还有许多问题：

- 是否AI程序可以理解规则?比如，理解马走日。
- 是否AI程序可以算出一些领域规则。开局的方法、子力计算等。

参考

- Introduction to Monte Carlo Tree Search by Jeff Bradberry
- Confidence interval

蒙特卡洛树搜索算法（UCT）：一个程序猿进化的故事的更多相关文章

1. 不变(Invariant), 协变(Covarinat), 逆变(Contravariant) : 一个程序猿进化的故事
   阿袁工作的第1天: 不变(Invariant), 协变(Covarinat), 逆变(Contravariant)的初次约 阿袁,早!开始工作吧. 阿袁在笔记上写下今天工作清单: 实现一个scala类 ...

2. 连载《一个程序猿的生命周期》-《发展篇》 - 3.农民与软件工程师，农业与IT业
   相关文章:随笔<一个程序猿的生命周期>- 逆潮流而动的“叛逆者” 15年前,依稀记得走出大山,进城求学的场景.尽管一路有父亲的陪伴,但是内心仍然畏惧.当父亲转身离去.准备回到 ...
3. 连载《一个程序猿的生命周期》- 44.感谢，我从事了IT相关的工作
   感谢博客园一直以来的支持,写连载都是在这里首发,相比较CSDN和开源中国气氛要好的多. 节前,想以此篇文章结束<一个程序猿的生命周期>的<生存>篇,对过10的年做一个了断,准备 ...
4. 连载《一个程序猿的生命周期》-28、被忽悠来的单身HR（女同志）
   一个程序猿的生命周期 微信平台 口 号:职业交流,职业规划:面对现实,用心去交流.感悟. 公众号:iterlifetime 百木-ITer职业交流奋斗 群:141588103 微 博:h ...
5. 连载《一个程序猿的生命周期》-6、自学C++，二级考过后，为工作的机会打下了基础
   一个程序猿的生命周期 微信平台 口 号:职业交流,职业规划:面对现实,用心去交流.感悟. 公众号:iterlifetime 百木-ITer职业交流奋斗 群:141588103 微 博:h ...
6. 专访雷水果国：离1.5K至18K 一个程序猿5每年的成长之路
   我只是一个小菜鸟,对于自主学习和交流PHP(jquery,linux,lamp,shell,javascript,server)等一系列的知识.小菜鸟创建了一个群.希望光临本博客的人能够进来交流. 寻 ...
7. 连载《一个程序猿的生命周期》-《发展篇》 - 7.是什么阻碍了"程序猿"的发展？
   有两件事想记录一下,具有普遍性和代表性."程序猿"加了引号,是泛指一类人,也并非局限于IT行业. 山东子公司的总经理是公司大股东之一,个子不高.有些秃顶.面容显老,但看 ...
8. 程序猿进化 - 在拉钩子1024对APE节讲座计划
   注意:下面这篇文章来自于我在网上拉勾1024对APE节现场演示程序. 我是蒋宇捷,信天创投的合伙人.之前是百度魔图的联合创始人. 我先做个自我介绍,事实上每次介绍自己事实上是非常痛苦的事情,由于我前不 ...
9. 为程序猿正名，MM们，你们为什么要找一个程序猿男票？【原创】
   前言 免责声明:这篇文章关于什么?六一儿童节马上就要到了,作为一个前端攻城师,自我感觉效率还可以,老早已把任务搞完,页面布局和前端编码高效按时交付,呵呵.趁有时间,写写文章娱乐一下.MM们,请不要拿起 ...

随机推荐

1. ABP督导项目（1）

   创建实体 项目名TQMASP 在领域层创建entities文件夹存放实体类如图 创建Dbcontext public virtual IDbSet<Supervisor> Supervis ...

2. MVC实用架构设计（三）——EF-Code First（4）：数据查询

   前言 首先对大家表示抱歉,这个系列已经将近一个月没有更新了,相信大家等本篇更新都等得快失望了.实在没办法,由于本人水平有限,写篇博客基本上要大半天的时间,最近实在是抽不出这么长段的空闲时间来写.另外也 ...

3. 51nod1085(01背包)

   题目链接: http://www.51nod.com/onlineJudge/questionCode.html#!problemId=1085 题意: 中文题诶~ 思路: 01背包模板题. 用dp[...

4. 工欲善其事-Maven介绍与使用

   Maven是什么? Maven是一个项目管理和综合工具.Maven提供了开发人员构建一个完整的生命周期框架.开发团队可以自动完成项目的基础工具建设,Maven使用标准的目录结构和默认构建生命周期. 在 ...

5. PHP require和include的区别

   require一个文件存在错误的话,那么程序就会中断执行了,并显示致命错误 include一个文件存在错误的话,那么程序不会中端,而是继续执行,并显示一个警告错误. 以下为补充:1. include有 ...

6. ns115 step by step

   一,安装环境: sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev lib ...

7. GsonUtils.java

   package com.vcredit.ddcash.batch.util; import java.util.ArrayList;import java.util.List; import org. ...

8. nginx 匹配.zip .apk 结尾的文件 直接下载

   server { listen 80; server_name ok.xidd.com; index index.html index.htm index.php; root /alidata/www ...

9. JSON和php里的数据序列化

   JSON就是一种数据结构,独立于语言 {"1":"one","2":"two","3":" ...

10. ios中摄像头/相册获取图片，压缩图片，上传服务器方法总结

    相册 iphone的相册包含摄像头胶卷+用户计算机同步的部分照片.用户可以通过UIImagePickerController类提供的交互对话框来从相册中选择图像.但是,注意:相册中的图片机器路径无法直 ...

Home

Powered By WordPress