关于德州扑克AI中Counterfactual Regret Minimization的介绍

fledsu
微信公众号 深圳病人
+ 关注他

今年初的AI的一个大新闻是 CMU开发的人工智能系统 Libratus在1v1无限注德州扑克上战胜了人类的顶尖选手。但是相比AlphaGo, Libratus带给社会的讨论还是小了很多。我觉得主要的原因是：由于有了AlphaGo等AI领域新进展的铺垫，在德州扑克上的胜利的困难性被缩小了，同时扑克也毕竟是一个小众的游戏；尤其是不久Nature又发了文章使用深度学习来成功诊断皮肤癌，更是抢过不少风头。

不过相比这两年风光无限的深度学习，Libratus采用的却是另外的方法，借用博弈论的原理，基于Counterfactual Regret(CFR) Minimization来进行开发。其基本思路是比较简单粗暴的，但最后可以证明能够得到一个纳什均衡点。这段时间简单看了看他们的一些工作，整理出来一些自己的理解，不一定准备，主要是和大家一起讨论。

CFR是基于非完全信息的博弈的。所谓非完全信息，就是在做决策的时候，有一部分信息是对玩家未知的，比如在德州扑克中，对手手上的牌就是一个未知信息。

首先对于一局游戏（如德州），可以把其转化成是一个树状结构。

每一个盘面的状态(State)（包括玩家手上的牌，当前的下注等游戏相关信息）都是树的一个节点。

在树的每一个节点上，玩家都可以从一系列的动作(Action)中（下注，放弃等）选择一个，然后盘面状态发生转换，从节点A变成节点B。因此每一个当前状态下的可能动作都代表树的一条边。

玩家不断的在不同的盘面状态下做决策，使盘面状态发生改变，直到终局（树的叶子节点Terminate State）。而玩家在当前状态下的决策函数（策略）(strategy)就是当前状态下所有可能动作的一个概率分布。

而每一局游戏，都可以看作是从根节点到叶子节点的一条路径。到叶子节点的时候的最终结果（输或赢了多少钱）就是这局游戏对玩家的效用(utility)。在1v1的游戏中，这是一个零和游戏，即是玩家A和玩家B总的效用为0。

由于这是一个不完全信息的游戏，因此对于没有上帝视角的玩家而言，一些不同的状态(state)给他带了的信息(infomation)是一样的。因此从玩家的角度，所有生成同样的信息的状态可以归为一个集合(infomation set)。

好了，基本的定义都说完了。我们的任务就是要实现一个AI,可以在不同的盘面状态下，制定出不同的策略（概率分布函数），从而在终局的时候实现更好的效用。为此CFR的方法是定义了一个Regrets 值，其含义是，在当前状态下，我选择行为A，而不是行为B，后悔的值是多少。

一个简单的例子：在石头剪刀布中，假设对方出剪刀，我也出了布。这个时候，我们会更后悔没有出 石头， 相比之下，不会那么后悔出 剪刀。因此一个简单的定义Regrets值的方法是根据最后的效用(utility)。比如在石头剪刀布中，对方出剪刀的情况下，我出布的效用是 -1; 出剪刀的效用是 0； 出石头的效用是 1，因此我选择布而不是剪刀的后悔值 R（剪刀->布）=1； 而不选择石头的后悔值R(石头->布） = 2

那么怎么用这个值来指导我们进行决策呢？我们可以使用Regret Matching的方法。其本质的思路是，如果在之前的游戏情况中，我们没有选取某一个行为的后悔值最大，那么在下一次我们就更偏向于选择该行为。

比如在剪刀石头布中，如果我们进行了n次游戏，然后把每次游戏没有选取某个行为的后悔值加起来，然后做一个归一化，就可以得到一个概率分布作为我们的策略。而在每一步的后悔值由对手的行为所决定。

$$\mathrm{R}(Action) = \sum_{i=1}^{n} R_i(Action) \;\; Action \in (剪刀，石头，布)$$

$$P(Action) = \frac{R(Action)}{R(剪刀) + \mathrm{R}(石头) + R(布)} \; Action \in (剪刀，石头，布)$$

这是一种比较简单的方式，因此只用进行一次决策，那么对于多次决策问题，就需要使用Counterfactual Regrets了。

首先我们要要记录的就不光是单个的盘面状态，而是整个状态转变的过程，或者是在我们刚刚说到的树状结构中的从根节点到当前节点的整个链路作为状态历史(history)。我们再定义概率 reach probability 为 游戏在某个策略通过这么一个路径走到当前节点的概率。如果有多个路径到达当前节点（不同的状态历史对应同样的信息集），那么当前信息集出现的概率为所有状态历史的reach probability之和。

那么我们怎么计算在当前的一个节点下，可能获得的收益或者叫counterfactual value呢？这个要分两个方面考虑，一个是在当前玩家希望到达的情况下，到达该节点的概率是多少？ 另一个是，从当前节点走到叶子节点（游戏结束），到达不同的叶子节点的概率是多少？ 然后我们可以根据叶子节点的效用来计算当前节点的counterfactual value 如下：

$$\mathrm{v}_i(s,h) = \sum_{t \in T} P^s(h|action(i)) * P^s(h \to t) * u(t)$$

$$r(h,a) = v_i(s_{I \to a}, h) - v_i(s,h)$$

其中，s 是使用的策略（概率分布），h是当前的历史（根节点到当前节点的路径）， i代表玩家。P代表概率，u代表效用， t是代表可能的结束的情况（叶子节点）。因此这个式子的含义是基于当前路径h, 当前策略s 的counterfactual_value 是 对所有可能的到达的叶子节点的效用加权求和 。这个权重是根据两个方面计算，第二项比较好理解，是从当前路径h到叶子节点t的概率。 第一项是假定当前玩家i在每次选择的时候都选择h对应的行动(Action）从而可以从根节点到达当前路径的概率。 因此很显然，counterfactual regret r(h,a) 就被定义为 玩家在当前节点选择了行动a的效用减去其他行动的效用。

最后，和之前的一样，把同一个信息集的状态所对应的regret加起来，就是该信息集的counterfactual regret r(I,a). 然后用前述的Regret Matching的方法就可以更新当前策略了。

CFR的方法的主要思想是把游戏中所有的盘面状态都考虑到，生成一颗完整的状态树，对树的每一个节点都初始化一个策略，然后根据这个策略来玩游戏，这个Alpha(

走状态树的一条边，然后根据游戏的结果来更新相关节点的策略。当CFR进行了许多次之后，这个状态树的每条路径都被遍历了很多次，每个节点的策略都被更新趋于纳什均衡了，从而得到一个可以玩游戏的AI。

这里有一个简单的游戏Kuhn Poker 的CFR的递归实现。Kuhn Poker的规则很简单，一共有三张牌1,2,3; 玩家A和B各拿一张牌。然后玩家A可以选择 放弃 或者 加注； 然后玩家B可以跟随选择 放弃 或者 加注。 如果两方都选择放弃，则牌大的一方可以加一点，牌小的一方减一点。如果双方都加注，则牌大的一方可以加两点，牌小的一方减两点。如果A加注，B放弃，则A+1, B-1； 如果 A放弃，B加注，A可以选择再加注，则牌大的一方+2; 牌小一方-2; 如果A选择再放弃；则B+1， A-1。

代码是用Python写的。首先是初始化策略。我把history和infomation set都定义成字符串，因此用字典表表示策略和counterfactual regret。

```
def CFR(action_history, reach_probs, player_cards): if isTerminal(action_history): #是
        current_player = len(action_history)%2 return getUtility(action_history, playe

    counterfactual_value_h = 0.0
    counterfactual_value_h_a = [0.0, 0.0]
    current_player = len(action_history)%2 #根据当前action history 计算infomation set
    info_set = str(player_cards[current_player])
    for i in range(len(action_history)):
        info_set += str(action_history[i])


    for i in range(2): #两种可能的行动 pass 0; bet 1;
        new_action_history = action_history
        new_action_history.append(i)
        new_reach_probs = reach_probs
        new_reach_probs[current_player] = new_reach_probs[current_player]*cur_strategy
        counterfactual_value_h_a[i] = CFR(new_action_history,new_reach_probs, player_c
        counterfactual_value_h += cur_strategy[info_set+str(i)]*counterfactual_value_h

   regretSum = 0.0 for i in range(2):
        regret[info_set+str(i)] += reach_probs[1-current_player]*(counterfactual_value
        AccStrategy[info_set+str(i)] += reach_probs[current_player]*cur_strategy[info_
        if regret[info_set+str(i)] > 0:
            regretSum += regret[info_set+str(i)]
    for i in range(2):
        if regretSum > 0:
            cur_strategy[info_set+str(i)] =  max(0, regret[info_set+str(i)])/regretSum
        else:
            cur_strategy[info_set+str(i)] = 0.5
        strategySum[info_set+str(i)]+= cur_strategy[info_set+str(i)]

    return counterfactual_value_h
```

其中info_set用字符串来表示，格式是card_value, action1,action2,....；行动 放弃是0， 加注是1，如201 表示当前玩家的牌是 2； 开始选择放弃，对方选择加注。这样的好处是，策略和counterfactual regrets都可以用dictionary来表示，key是info_set, value就是具体的值。当经过多次迭代后，AI的策略是，如果手牌是3，选择bet; 手牌是1，选择pass,如果是2，则会有变化。如果牌的数目增加，比如从3增加到5，则变化会更多一些，当然相应的计算空间和资源要更多，收敛所需要的迭代次数也是要增长很多。

可以看到，该方法的主要问题是，需要的存储空间和计算资源都特别大。因此在实际中，需要缩减搜索空间，比如合并一些Information set, 对一些概率很低的路径进行剪枝，减少路径的记录长度等等，当然这些自然会带来算法效果上的损失。也有不少对CFR的改进，如Fixed Strategy Iteration CFR Minimization，在Forward玩游戏的时候不更新strategy;之后通过backward机制来更新，和batch normalization的机制也有点像。还有CFR+, 就是这个战胜人类的Libratus采用的方法。

我觉得，CFR的进一步发展，应该借鉴alpha go对MCST的改进，使用一个类似value network的东西来帮助CFR快速的更新策略。最近的DeepStack的论文的思路可能就是这样的，不过我还没有细看。

另外一个挑战就是，怎么把1v1的AI扩展到多人对战以及合作的情况，感觉CFR这种基于博弈论的方法可能只能得到一个不坏的结果，但是不一定能够收敛到一个最优的情况。因此AI要征服的下一个游戏是 麻将 和 四国军棋么？~~

参考文章：An Introduction to Counterfactual Regret Minimization Regret in Games, Neller, Todd W; Lanctot, Marc; 2013

欢迎关注：

weixin.qq.com/r/EESdhTL... (二维码自动识别)

编辑于 2018-03-26

德州扑克 人工智能

文章被以下专栏收录

RS&GISer的学习笔记

本专栏虽然看着是像为RS&GISer设立的，但绝不局限于遥感、地信甚至是地理背景...

关注专栏

推荐阅读

德州扑克概率中的一些冷知识（持续整理中）

1、AK vs 对子的最大胜率？AsKs vs 2c2d = 50.1%：49.9%2、两高张 vs 对子的最大胜率？JsTs vs 2c2d = 54.0%：46.0%——JsTs成花顺比例最高，不被对子block；同时22被“淹没”的概率最高。3...

Newbaby

德州扑克常见术语整理

曾有读者给我发消息，说我德州扑克里的英文术语太多，看不明白，查词典也查不到这些单词在德州扑克中的意思。所以我整理了一下德州扑克中常见的英文术语，当然，一些过于常见的（如blinds，...

火烧风

7 条评论

切换为时间排序

写下你的评论...

温柔的暴力 1 年前

这么好的文章啊，谢谢了，作者

2

wpp 1 年前

解析的很明白，对读原论文很有帮助，如果用c++实现cfr，选择什么样的数据结构呢

赞

fledsu (作者) 回复 wpp 1 年前

这个~ ~很难说~ ~

赞

wpp 回复 fledsu (作者) 1 年前

谢谢！我可以再问一下， 在你的cfr代码例子中， 在信息集上的相关计算，这种当前节点所处信息集与信息集中所包含其他历史路径的关系是怎样定义的，因为在计算虚拟遗憾时需要考虑到信息集中包含的所有路径。我是初学者， 会有很多困惑， 方便的情况下希望可以得到一些指点

赞

展开其他 1 条回复

李安 11 个月前

厉害啊

赞

Rain 10 个月前

文章很给力啊，看到的论文全是英文的，看了半天什么也没看懂，看到老哥的文章有一种豁然开朗的感觉！老哥有时间可以写写DeepStack的文章，造福小白！

赞