德州扑克 AI 报告

姓名：**梁苏叁 李龙成 陈彦帆**
学号：2018K8009918009 2018K8009915034 2018K8009918002
https://github.com/Cothrax/deepfool

1 问题介绍

德州扑克属于不完美信息博弈的一种，一直被人工智能学界作为测试解决不完美信息博弈的算法的基准游戏之一，多种基于 CFR 的方法已经成功解决了 2 人的德州扑克 AI 问题，但 6 人无限注的德扑 AI 依然是一个 open 的问题。在本次大作业中，我们结合的博弈树搜索和深度学习的方法，设计了 6 人无限注的德扑 AI DeepFool，并在 `neuron_poker` 环境对模型进行了测试。

2 模型与算法

DeepFool 采用了蒙特卡洛虚拟遗憾最小化（MCCFR）与神经网络结合的方法，在一轮迭代中，用 MCCFR 与一个策略网络交互，计算遗憾值产生带标注的训练样本，再交给策略网络学习，通过反复迭代收敛到一个较优的策略。在实际测试中，通过将环境信息输入策略网络来得到决策的概率分布。

2.1 MCCFR

算法的主要框架采用了虚拟遗憾最小化（CFR）的方法，寻找一个理论上为纳什均衡的博弈策略。我们将策略定义为一个已知信息 I 到合法行动的概率分布 π 的映射 $\sigma : I \to \pi$ ，已知信息 I 包含了对于当前玩家的所有可见信息，如手牌、公共牌、决策历史等，概率分布 π 的每一项 π_a 代表在已知 I 的情况下采取行动 a 的可能性。

CFR 是一种求解纳什均衡策略的高效算法，它的一次迭代产生一颗博弈树，博弈树上的所有玩家都根据策略 σ 行动，每个节点 u 都会得到一个每个玩家 p 的期望收益 $u_{u,p}$ ，对于终止节点，收益即为游戏结束后赢得的筹码；对于非终止节点，收益为该节点的所有子节点的收益的概率加权 (2)。树上的一个节点已知信息对应了映射 σ 中的一项 I ，在这个节点上一个合法行动 a 的遗憾值 (3) 等于以概率 1 采取行动 a 的期望收益减去 u （严格来讲，还需要乘上从根到该节点上，其他玩家决策概率贡献的乘积作为系数），每次迭代遗憾值累加后再归一化就得到了 I 下的决策概率分布。CFR 被证明可以在 $O(N^2)$ （N 状态空间大小）内收敛到一个纳什均衡的策略 [3] 。

$$v = \textsc{Next-State}(u, a) \tag{1}$$

$$\textsc{Util}_{u,p} = \sum_{a \in \textsc{Action}(u)} \pi_a \cdot \textsc{Util}_{v,p} \tag{2}$$

$$\textsc{Regret}_{u,a} \propto \textsc{Util}_{v,p} - \textsc{Util}_{u,p} \tag{3}$$

但在 6 人的德州扑克中，直接使用朴素的 CFR 会带来两个问题：

1. 博弈树太大，比如在 `neuron_poker` 环境中，环境实际只允许每轮加注 1 次，这意味着最多有 8 轮迭代，环境允许的行动有 6 种，则一次完整的迭代需要访问 $6^{6\times 8}$ 个节点，这依照当代计算机的算力是不现实的。
2. 状态空间太大，即 σ 里的项数太多，需要大量的迭代才能收敛到一个较好的策略。

针对这两个问题，我们的模型对朴素的 CFR 做出了两点改进。

1. 对于第 1 个问题，我们采取的蒙特卡洛采样（MC）的方法，具体来讲，对于每次迭代，固定一个当前玩家 P ，只在 P 决策的节点展开子树，在其余玩家决策的节点直接根据决策概率采样，这样搜索的规模被缩小到了 6×6^6 。可以证明当采样次数足够多时，每个节点采样得到效用的期望等于实际的效用值 [1] 。

2. 对于第 2 个问题，我们用一个策略神经网络来拟合决策映射 σ ，替代朴素 CFR 采用的表格法，CFR 在搜索时通过询问策略网络来遍历博弈树并计算遗憾值，用计算的遗憾值更新得到的节点的决策，将这些新决策作为产生的训练样本交给策略网络学习。

2.2 策略网络

由于德州扑克的状态空间极大，单纯地使用 CFR 进行决策学习需要巨大的存储空间保存每种状态的决策概率分布，因此我们通过使用策略网络学习 MCCFR 在不同输入状态下的决策概率分布，用策略网络表示 CFR 的决策概率分布，实现对信息的压缩。

决策网络整体结构为多层感知机，网络包含三个输入分支，分别为玩家的手牌、当前的公共牌以及所有玩家之前的历史决策，输出为取每种决策（弃牌、跟注、加注等等）的概率，每层感知机包含全连接层和 ReLU 激活层，模型在输出前会通过 SoftMax 层将采取每种决策的概率进行归一化，为了防止网络过拟合在将牌信息和历史信息进行合并时使用了 Dropout 操作，模型的整体结构如图 1 所示。

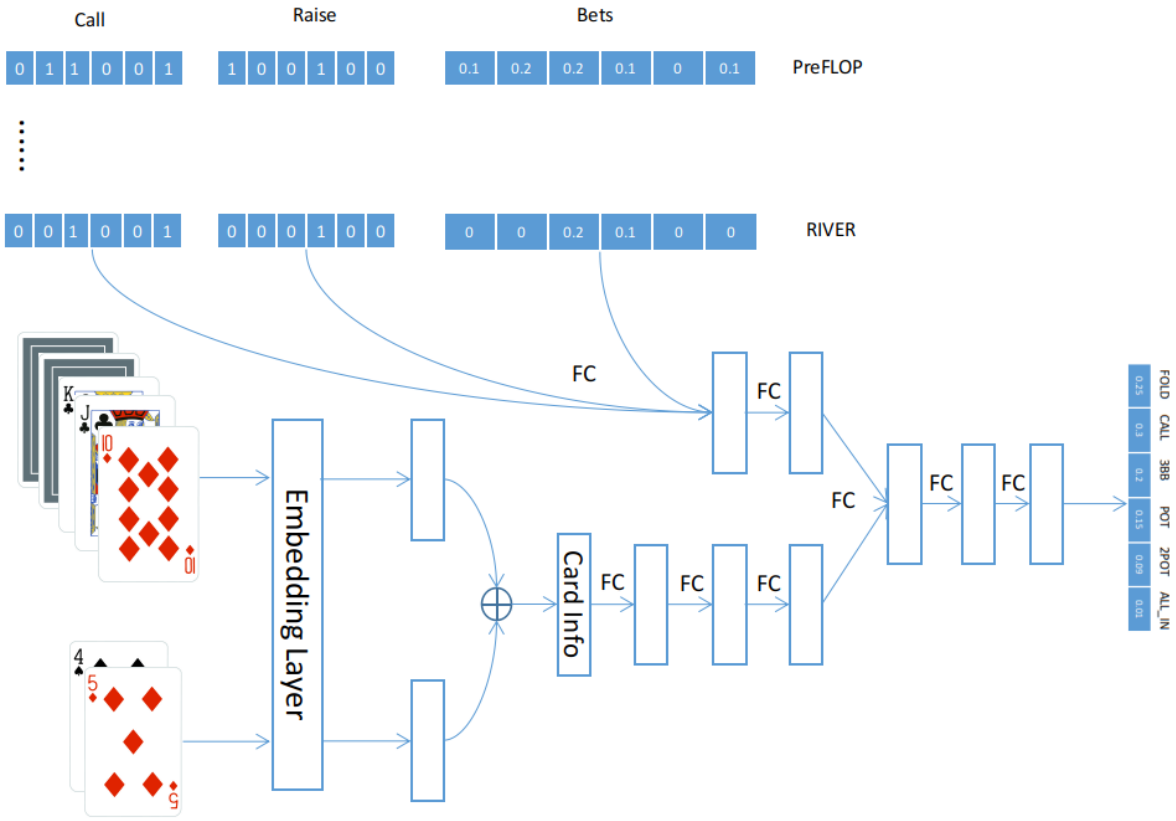

Fig. 1: 策略网络结构图

由于扑克共有 52 张牌，包含 4 种花色，每种花色包含 13 张大小不同的牌，让模型直接判断手牌的大小和花色关系较为困难，因此我们将每张牌根据花色和大小进行编码构造为 `embeddings` 后再输入网络。具体实现中，手牌和公共牌会分别进行编码再叠加在一起作为策略网络的牌信息。

在德州扑克的对局过程中，往往需要通过其他玩家的历史决策信息调整自己的动作决策，因此我们在决策网络中引入了玩家的历史决策信息，希望模型借助已有的历史信息更好地学习动作决策。

由于策略网络需要根据输入信息拟合不同决策的概率分布，因此在约束函数部分，我们选择使用 KL（Kullback-Leibler）散度度量模型输出的概率分布与 MCCFR 提供的概率分布之间的差异，以要求模型尽可能拟合 MCCFR 的概率分布。约束函数如下所示：

$$l(X,Y)=\sum_{i=1}^{n}(y_n\cdot(\log y_n-x_n)) \tag{4}$$

X 为模型输出的概率分布（为包含 n 维的向量），Y 为 MCCFR 给出的概率分布（也为包含 n 维的向量）。

2.3 牌力计算

1. 确定牌力计算：在判断胜负的阶段，需要比较每个玩家 7 张牌的牌力大小。具体实现时，先考虑给定的排好序的 5 张牌的大小。依次检查同花、顺子、四条、葫芦、三条、两对、一对、高牌，以及牌面数字，计算出牌力大小，以一个整数表示。为了提高速度，预先计算好所有 13 选 5 组合的牌型大小（考虑同花和非同花两种情况）。对于给定的 7 张牌，考虑所有 21 种可能的牌型情况，返回其中最大的 5 张牌的牌力。

2. 基于蒙特卡洛采样的期望牌力计算：在游戏进行过程中，需要根据已知的底牌和公共牌信息可以得到当前玩家的牌面胜率。未知牌的组合情况较多，采用蒙特卡洛模拟方法。具体来说，随机采样未知的牌，计算所有玩家的牌力大小，得到当前玩家本轮模拟的胜负情况。重复多次模拟，取平均值，得到当前玩家的期望胜率。

为加快速度，牌力计算用 C++ 实现，得到的结果按需缓存到哈希表中。

3 实验细节

3.1 预训练策略网络

由于 MCCFR 的策略学习是基于多次迭代的增量式学习，因此需要通过大量的迭代才能让模型学到正确的决策，所以在有限的算力下直接使用 MCCFR 进行迭代学习收敛速度太慢。为了加速训练过程，我们使用基于规则的 Agent 在给定牌型下进行决策，生成大量的“信息、决策”样本对，决策网络首先在样本对上进行反复学习从而具备与规则 Agent 相同的决策能力，实现决策网络的预训练过程，再在此网络基础上和 MCCFR 进行交互学习，从而应对更复杂的决策情况。

规则 Agent 在生成“信息、决策”样本对时，将随机初始化手牌和公共牌（公共牌可能部分不可见）信息，通过牌力计算器计算出当前的获胜概率，以获胜概率为中心构造高斯分布，不同的动作依据自身的冒险程度采样高斯分布得到做出该决策的概率。

3.2 基于 CFR 的决策历史学习

在得到一个初步可用的网络后，再采用 MCCFR 来生成决策样本，但因为使用 MCCFR 时每一个节点都需要询问一次神经网络，这样生成样本的效率偏低，所以这一部分分为两个子步骤：

1. 通过直接采样得到大量的样本进行训练：想得到一次迭代的搜索树结构，只需要知道采样的节点选择了哪个行动，在这一个子步骤，MCCFR 在采样时不询问策略网络，而直接随机选择一个节点，这样可以在完成整个搜索树的遍历之后，将所有要询问的样本交给策略网络，可以大大加快迭代速度。

2. 但子步骤 1 的做法违背了 MCCFR 的原理，在理论上是不收敛的，所以在直接采样了一定轮数之后，还是采用标准的 MCCFR 进行搜索树的遍历，伪代码如下：

Algorithm 1: MCCFR 训练决策网络伪代码

```
Input: Current state game, sampling player player, strategy network network
Output: Utility for current state game

Function MCCFR(game, player, network) Begin
    If STATUS(game) = GAME_OVER Then
        Return PAYOFF(game)

    strategy ← GET-STRATEGY(network, game)
    If PLAYER(game) = player Then Begin
        action ← SAMPLE-ACTION(strategy)
```

```
            next ← TAKE-ACTION(game, action)
            Return MCCFR(next, player, network)
        End Else Begin
            cvUtil ← [0, 0, ··· , 0]
            util ← [0, 0, ··· , 0]
            For action ∈ LEGAL-ACTION(game) Begin
                next ← TAKE-ACTION(game, action)
                nextUtil ← MCCFR(next, player, network)
                cvUtil_action ← nextUtil_player
                util ← util + strategy_action · nextUtil
            End
            regret ← cvUtil − util_player
            UPDATE-STRATEGY(network, regret)
            Return Util
        End
    End
```

4 结果与讨论

4.1 neuron_pocker 环境测试结果

4.2 存在问题与讨论

4.2.1 模型的收敛问题

虽然使用神经网络的方法压缩了决策映射的，但因为 MCCFR 的学习决策历史时每次都需要询问策略网络，这导致迭代的速度变得非常慢，我们在训练时采用了并行多个 MCCFR 的方法，即每个子进程遍历一颗博弈树，主进程将生成的 regret 整合成新的样本。尽管如此，由于我们笔记本有限的算力和训练时间，模型没能完全收敛，所以得到的决策网络鲁棒性不好。这是未来可以进一步优化的方向。

4.2.2 neuron_poker 环境的问题

在测试中，我们发现 neuron_pocker 实际存在如下问题：

1. 大作业的要求为设计 6 人无限注德州扑克 AI，但环境中的 is_raising_allowed 函数每轮只允许加注 6 次，也就是说环境中游戏最多进行 8 圈（每轮一圈加注，一圈跟注）
2. 环境中的历史信息没有严格的时序关系，只有在某一轮每个人是否 raise/check，也没有提供 fold 的决策历史和有多少人出局的信息，这给决策历史部分网络的设计带来的很大的困扰。

参考文献

[1] Neil Burch, Marc Lanctot, Duane Szafron, and Richard Gibson. Efficient monte carlo counterfactual regret minimization in games with many player actions. *Advances in Neural Information Processing Systems*, 25:1880–1888, 2012.

[2] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[3] Todd W Neller and Marc Lanctot. An introduction to counterfactual regret minimization. In *Proceedings of Model AI Assignments, The Fourth Symposium on Educational Advances in Artificial Intelligence (EAAI-2013)*, volume 11, 2013.