《软件产业概论》期末作业

171860607
白晋斌
计算机科学与技术系

关于 Alpha-GuanDan 的研发计划

目录

一、引言

1.编写目的

本研发计划的主要目的是确定一个基于 MCTS（Monte Carlo Tree Search，蒙特卡罗树搜索）、CNN（Convolutional Neural Networks，卷积神经网络）和 RL（reinforcement learning，强化学习）的掼蛋[1]扑克牌博弈程序研发所需要的背景知识、工作内容与研发难点。供研发人员阅读，做到及时协调，按步有序进行项目的研发，减少研发中的不必要损失。

2.项目背景

1956 年“人工智能”首次在达特茅斯会议中被提出后开始酝酿其第一次浪潮，人工智能实验室在全球各地扎根。直到 1973 年，以《莱特希尔报告》宣称”AI 领域的任何一部分都没有能产出人们当初承诺的有主要影响力进步“。各国政府勒令大规模削减人工智能方面的投入，这象征着人工智能正式进入寒冬。而到了八十年代，由于专家系统的崛起人工智能再次迎来一次久旱之后的甘霖期，但随着 1987 年基于通用计算的 Lisp 机器在商业上的失败，人工智能再次滑入了低迷期。到了上世纪九十年代后期，由于计算机计算能力的不断提高，人工智能再次卷土重来。以数据挖掘和机器学习为主要代表的应用非常成功，使人工智能重回人们的视野。

在 1996 年，IBM 开发了基于 RS6000 的超级计算机“深蓝”，并对它输入了 100 年来所有国际象棋特级大师的开局和残局下法，让它通过 alpha-beta 剪枝进行评估与下棋，并挑战世界冠军俄罗斯人卡斯帕罗夫。第一次比赛“深蓝”输了。但一年后（1997 年）升级之后的“深蓝”第二次再次挑战，这次“深蓝” 成功战胜国际象棋世界冠军卡斯帕罗夫[2]。

在 2017 年， AlphaGo Zero 提出了新的估算搜索评价函数的方法[3]，即基于蒙特卡洛树搜索的强化学习；部分地解决了超大状态空间搜索的难点；成功地应用到围棋领域并完胜了人类；证明了强化学习的有效性，是人工智能史上一座里程碑。AlphaGo Zero 虽然不是开创性工作，但它所提出的方法很好地结合了已有的两种重要方法。对于像扑克牌等人工智能其他领域和难点，或许能触类旁通。

3.定义

3.1 蒙特卡罗树搜索

蒙特卡罗树搜索（Monte Carlo Tree Search），是一类树搜索算法的统称，可以较为有效地解决一些探索空间巨大的问题，例如一般的围棋算法都是基于 MCTS 实现的。这类算法要解决的问题是这样的，我们把围棋的每一步所有可能选择都作为树的节点，第零层只有 1 个根节点，第 1 层就有 361 种下子可能和节点，第 2 层有 360 种下子可能和节点，这是一颗非常大的树，我们要在每一层树节点中搜索出赢概率最大的节点，也就是下子方法。那么树搜索的算法有很多，如果树的层数比较浅，我们可以穷举计算每个节点输赢的概率，那么可以使用一种最简单的策略，叫做 minmax 算法。基本思路是这样的，从树的叶子结点开始看，如果是本方回合就选择 max 的，如果是对方回合就是 min 的，实际上这也是假设对方是聪明的也会使用 minmax 算法，

这样在博弈论里面就达到一个纳什均衡点。这是搜索空间比较小的时候的策略，MCTS 要解决的问题是搜索空间足够大，不能计算得到所有子树的价值，这是需要一种较为高效的搜索策略，同时也得兼顾探索和利用，避免陷入局部最优解。MCTS 实现这些特性的方式有多种，例如经典的 UCB（Upper Confidence Bounds）算法，就是在选择子节点的时候优先考虑没有探索过的，如果都探索过就根据得分来选择，得分不仅是由这个子节点最终赢的概率来，而且与这个子节点玩的次数成负相关，也就是说这个子节点如果平均得分高就约有可能选中（因为认为它比其他节点更值得利用），同时如果子节点选中次数较多则下次不太会选中（因为其他节点选择次数少更值得探索），因此 MCTS 根据配置探索和利用不同的权重，可以实现比随机或者其他策略更有启发式的方法。

MCTS 的算法分为四步，第一步是 Selection，就是在树中找到一个最好的值得探索的节点，一般策略是先选择未被探索的子节点，如果都探索过就选择 UCB 值最大的子节点。第二步是 Expansion，就是在前面选中的子节点中走一步创建一个新的子节点，一般策略是随机进行一个操作并且这个操作不能与前面的子节点重复。第三步是 Simulation，就是在前面新 Expansion 出来的节点开始模拟游戏，直到到达游戏结束状态，这样可以收到到这个 expansion 出来的节点的得分是多少。第四步是 Backpropagation，就是把前面 expansion 出来的节点得分反馈到前面所有父节点中，更新这些节点的 quality value 和 visit times，方便后面计算 UCB 值。

基本思路就是这样的，通过不断的模拟得到大部分节点的 UCB 值，然后下次模拟的时候根据 UCB 值有策略得选择值得利用和值得探索的节点继续模拟，在搜索空间巨大并且计算能力有限的情况下，这种启发式搜索能更集中地、更大概率找到一些更好的节点。前面提到 MCTS 的使用场景需要是搜索空间巨大，因为搜索空间如果在计算能力以内，其实是没必要用 MCTS 的，而真正要应用上还需要有其他的假设。涉及到博弈论的概念，我们要求的条件是 zero-sum、fully information、determinism、sequential、discrete，也就是说这个场景必须是能分出输赢（不能同时赢）、游戏的信息是完全公开的（不像打牌可以隐藏自己的手牌）、确定性的（每一个操作结果没有随机因素）、顺序的（操作都是按顺序执行的）、离散的（没有操作是一种连续值），除了博弈论我们要求场景是一个黑盒环境，不能通过求导或者凸优化方法找到最优解，否则使用 MCTS 也是没有意义。

3.2 卷积神经网络

卷积神经网络（Convolutional Neural Network，CNN）是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元。它包括卷积层(convolutional layer)和池化层(pooling layer)。

卷积层(convolutional layer)：在原始的输入上进行特征的提取。特征提取简言之就是，在原始输入上一个小区域一个小区域进行特征的提取。池化层(pooling layer)就是对特征矩阵进行特征压缩，池化也叫做下采样。选择原来某个区域的 max 或 mean 代替那个区域，整体就浓缩了。

3.3 强化学习

机器学习可以分为三类，分别是 supervised learning（监督学习），unsupervised learning（无监督学习） 和 reinforcement learning（强化学习）。而强化学习与其他机器学习不同之处为：强化学习没有教师信号，也没有 label。只有 reward，其实 reward

就相当于 label。强化学习反馈有延时，不是能立即返回。强化学习输入数据是序列数据。强化学习中 agent 执行的动作会影响之后的数据。

强化学习的关键要素有：environment，reward，action 和 state。强化学习即先从要完成的任务提取一个环境(environment)，从中抽象出状态(state) 、动作(action)、以及执行该动作所接受的瞬时奖赏(reward)。有了这些要素我们就能建立一个强化学习模型。强化学习解决的问题是，针对一个具体问题得到一个最优的 policy，使得在该策略下获得的 reward 最大。所谓的 policy 其实就是一系列 action。也就是 sequential data。

4.参考资料

[1] https://baike.baidu.com/item/%E6%8E%BC%E8%9B%8B/10312030?fr=aladdin
百度百科中关于掼蛋的游戏规则
[2] van den Herik, H. J., Herschberg, I. S., Marsland, T. A., Newborn, M., & Schaeffer, J. (1992). IBM Deep Blue. *ICCA Journal,* 15(4), 245-245.
[3] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Den Driessche, G. V., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature,* 529(7587), 484-489.

二、项目概述

1.工作内容

主要任务是研发一个掼蛋扑克牌博弈程序。以下将通过对扑克牌游戏的建模、输入输出的处理、对手出牌的不确定性、游戏胜负的判定、数据的获取与空间的搜索共六个方面展开分析研发要完成的工作。

1.1 扑克牌游戏的建模

相比较于围棋，扑克牌的建模则略显困难。这里我们有两种建模思路。一种是将手中拥有的扑克牌抽象为 13*4+2=54 维的位示图，即每张扑克牌是否拥有可以用一个 bit 来表示，这样，每个玩家每个时刻手中拥有的牌的种类，可以用 54 个 bit 即不到 7 个字节来表示。另一种是对每种牌进行编码，每种牌用 6 个 bit 即可编码，这样每个人拥有的牌即可编码为 n*6 bit，其中 n 为当前手上拥有的牌的数量。具体采用哪种建模方式，可以分为两个小组分别进行尝试。

1.2 输入输出的处理

在围棋中，作为数据的<输入，输出>对被定义得很清楚，输入就是棋盘状态而输出就是当前状态下的选择。但在扑克牌问题上，函数的定义没那么清楚。在这里我们根据第一小节的建模，把输入记为<当前轮最厉害的牌，各玩家已出的牌，自己的牌，各玩家未出的牌（可根据总牌量-各玩家已出的牌-自己的牌求得）>，输出即自己要出的牌（如果不出则要出的牌为空）。

1.3 对手拥牌和出牌的不确定性

在围棋中，每个行动都有确定的结果，落子必定会成功。然而在扑克牌问题中，行动的后果是不确定的，并不保证一定成功。例如自己出牌压制了上家，牌又有可能被下家压制。此外，只知道所有玩家未出的牌的总集合，不清楚每个玩家未出的牌的集合（虽然可以根据每个玩家的出牌逻辑进行分析推断）。由于对其他玩家所拥

有扑克牌的未知性，仅根据自身所拥有的牌以及当前轮回的出牌情况，程序需要首先推断对手可能的出牌情况并进行剪枝。这一部分难点在于列出对手可能的出牌。

1.4 游戏胜负的判定

在围棋中，游戏规则简单且通用。但在扑克牌游戏中，通过阅读百度百科，我们发现掼蛋游戏的游戏规则较为复杂，如果正确的判定自己是否能取胜当前牌面，也需要额外一部分精力去研发。

1.5 大规模高质量数据的获取

在围棋中，奖惩机制相当清楚。所以很容易获取强化学习需要的数据。然而在扑克牌游戏中，通过对扑克牌全流程的转化，使之成为一个合适的输入就是一个很复杂的问题，这需要一部分精力单独去研发。

1.6 庞大空间的搜索

关于这个问题，扑克牌的搜索空间远小于围棋，因此我们初步可以尝试 alpha-beta 剪枝暴力枚举，之后再考虑结合蒙特卡洛搜索与强化学习。

2.产品

实现一个基于 MCTS（Monte Carlo Tree Search，蒙特卡罗树搜索）、CNN（Convolutional Neural Networks，卷积神经网络）和 RL（reinforcement learning，强化学习）的掼蛋扑克牌博弈程序。

3.运行环境

跨平台，包括 Windows、macOS、Linux、android、iOS。

4.验收标准

扑克牌出牌的效率、功耗和 AI 能力。

三、实施计划

1.技术路线

1.0 1v1 掼蛋

先从简单的情况开始考虑。因为 1v1 掼蛋可以从己方拥牌和已出牌中推测出对方拥牌，所以 1v1 掼蛋是完全信息游戏。首先从实现 1v1 掼蛋开始。

1.1 游戏建模

可维护 3 个长度为 54 的位示图，分别表示己方拥有的牌，已出的牌，对方拥有的牌（对方拥有的牌可通过总牌数-己方拥有的牌-已出的牌得到）。

掼蛋扑克牌的牌型共有单牌、对牌、三张、三带两、顺子、三连对、钢板、火等 8 种。我们可以通过本人拥有的长度为 54 的位示图，分别构造这八种可出的牌，同样可以用 8*k*54（k 代表某牌型可以创造的出牌种类）长度的位示图做标记。每个出牌轮回都会更新自己的出牌表。

对 8 种出牌方式，分别定义所对应的比较方式，大或者小。可通过位运算定义。并定义不同牌种之间的大小比较关系。

当某方拥牌的位示图==0 时，游戏结束。

1.2 alpha-beta 剪枝与蒙特卡罗树搜索

首先从最基本的技术开始[1]。对于己方出牌，分别计算对方可以进行的攻击（或者是“不要”），再分别枚举己方可以出的牌，构造蒙特卡罗树，对每个节点的权值按照自己出牌数量*权重-对手出牌数量*权重来赋值，最终进行 alpha-beta 剪枝。这里有个细节是如何定义权重？有两种解决方案，一种是直接按牌的大小来进行加权，越大的牌权重越小（好牌留在最后）；另一种是按照牌的连贯性来进行加权，即出掉这几张牌后，会有多少种可能的出牌方式失效（失效出牌方式越大，权重越小），毕竟分散的牌质量不如整牌高。

1.3 强化学习与卷积神经网络

首先是大规模高质量数据的获取。这里我们通过一些棋牌类软件可以导出一些掼蛋数据。然后数据处理为 3*54*R（R 为回合数）的位示图。以此作为训练与检测数据集。在自我博弈的过程中，扑克牌系统可以独立地找出扑克牌游戏的数学最优策略。具体的训练过程可以结合深度学习方法如 CNN（Convolutional Neural Network，卷积神经网络）、DNN（Deep neural network，深度神经网络）、DBN（Deep Belief Network，深度置信网络）等。在训练数据数量不足，或者训练数据不足以保障训练质量时，强化学习可以帮上忙。在其支撑下，掼蛋扑克牌博弈程序在执行任务时可以从自己的错误中吸取教训，进而强化训练效果。这个循环会一直持续，直至掼蛋扑克牌博弈程序的表现达到要求为止。

强化学习如何应用于掼蛋扑克牌博弈程序呢？在任何决策点上，玩家知道他的所拥有的所有牌以及对手拥有的所有牌，这就是状态。然后他可以采取行动：出某种组合的牌，不出。然后得到奖励——赢得本回合胜利/赢得本局胜利。通过模拟组合出牌来找到游戏的策略。掼蛋扑克牌博弈程序将使用该信息来学习 Q 函数 Q(S,A)。Q 的参数为状态 S 和动作 A，输出值为在该状态下采取该动作时得到的最终奖励值。一旦我们有 Q，策略选择就很容易了：我们可以评估每个策略，看哪一个更好。所以，这里的主要任务是估计 Q，使用 Q’来指代这个估计。初始化时，掼蛋扑克牌博弈程序随机猜测一些 Q’。然后，模拟一些出牌，两名玩家根据 Q’ 做出决定。每个出牌回合结束之后，掼蛋扑克牌博弈程序将调整估计值 Q’，以反映玩家在特定状态下采取特定动作后获得的实际值。最终，掼蛋扑克牌博弈程序应该得到一个很好的 Q’估计，这就是确定玩家策略所需的所有内容。

这里需要注意一点——要确保在所有状态采取所有动作，每个状态-动作组合至少尝试一次，这样才能很好地估计出最终每个可能的值。所以，我们会让掼蛋扑克牌博弈程序在一小段时间ε内随机地采取行动，使用他们（当前估计的）最佳策略。首先，掼蛋扑克牌博弈程序应积极探索选择的可能性，频繁地随机选择。随着时间的推移，掼蛋扑克牌博弈程序将更多地利用自己之前所获得的知识。也就是说，ε将随着时间的推移而缩小。

简单说，首先随机发牌。然后令掼蛋扑克牌博弈程序选择一个动作并得到反馈。最后使用观测到的（状态，动作，结果）元组更新模型直至达到自己的预期。

1.4 输出

当掼蛋扑克牌博弈程序通过计算求得最优解或者在当前时间限制下得到局部最优解，便将输出的长度为 54 的位示图转化为具体的扑克牌内容并输出。

[1] 本人曾用 alpha-beta 剪枝实现过一个黑白棋 AI，可以稳赢普通同学。

1.5 NvN 掼蛋

如果掼蛋游戏的玩家数大于 2 时，掼蛋扑克牌博弈程序便无法通过自己手上的扑克牌和已输出的扑克牌来确定每个用户拥有什么扑克牌。这时候就要考虑让掼蛋扑克牌博弈程序从不完全信息游戏的自我博弈中深度强化学习。

具体可以先假设某用户拥有某张牌，维护该用户的假设位示图某一位是 1。然后如果觉得该用户应该出这张牌却没有，则将位置为 0，并传给其他人。或者是如果连续若干个回合该用户都没有使用这张牌，则将位置为 0，并传给其他人。具体的训练过程可以结合前述 1v1 训练逻辑与对其他玩家所拥有牌的猜测与剪枝。

2.关键问题

多人对战时，其他玩家手中的牌是不确定的，这就要考虑到让掼蛋扑克牌博弈程序从不完全信息游戏的自我博弈中深度强化学习的能力。

3.时间安排

1 个月时间实现掼蛋的建模与基本函数。

1 个月时间实现 1v1 掼蛋扑克牌博弈程序。

3-6 个月时间实现 NvN 掼蛋扑克牌博弈程序并对扑克牌出牌的效率、功耗和 AI 能力进行优化。

四、技术可行性分析

掼蛋扑克牌博弈程序所需要的三项核心技术：MCTS（Monte Carlo Tree Search，蒙特卡罗树搜索）、CNN（Convolutional Neural Networks，卷积神经网络）和 RL（reinforcement learning，强化学习）均已成熟，且已被运用在国际象棋与围棋游戏中，所以 1v1 掼蛋游戏程序应该是容易的。对于 NvN，则要考虑掼蛋扑克牌博弈程序从不完全信息游戏的自我博弈中深度强化学习的能力，此处亦可用蒙特卡洛搜索树对对方可能拥有的牌进行推理、剪枝。因此本研发项目是可行的。