基于 AlphaZero 的博弈游戏 AI 研究与实现之实现一个 AlphaZero 框架

SC

摘　要　近年来,“围棋游戏 AI 永远无法战胜人类”的言论被谷歌旗下的 DeepMind 公司研发的 AlphaGo 人工智能彻底击破。AlphaGo 有多个版本，其中 AlphaGo Zero 在仅使用增强学习，完全从零开始，没有任何人类在围棋领域的知识帮助的情况下学会了下围棋，并且击败了人类最强棋手。本篇文章会对运用到 AlplaGo Zero 的增强学习算法进行详细分析，并将此算法推广到多种博弈游戏，实现一个训练博弈游戏 AI 的通用框架。这个增强学习算法需要以一个具体的博弈游戏规则为基础，使用一个具体的策略价值神经网络，每步配合通用的蒙特卡洛树搜索算法走棋，不断进行自我对局，并使用自我对局过程产生的数据来不断训练神经网络，最终这个由神经网络和蒙特卡洛树组合的人工智能能很好的理解并掌握这个游戏。

关键词　增强学习；博弈游戏；人工智能

Abstract　Recent years, the invention of AlphaGo, a superhuman artificial intelligence of Go game, which was developed by Google’s DeepMind Company, broke the belief that Go game AI can never beat humans. AlphaGo Zero is one of AlphaGo’s many versions, it only applies reinforcement learning, starts from tabula rasa, learns how to play Go without any domain knowledge beyond game rules, and has defeated the best human Go player. This article carefully analyses the algorithm used in AlphaGo Zero, generalizes this approach to many similar games, and implements a simple AlphaZero framework. This reinforcement learning algorithm needs to base on a game rule, uses a policy value neural network, which helps Monte Carlo tree search to move in each turn, keeps playing with itself, uses self-play data to train the neural network, and finally this artificial intelligence combined with neural network and Monte Carlo tree can understand and master this game well.

Key words　reinforcement learning; game; artificial intelligence

引言

自从计算机技术的兴起，人工智能一直是人类热议的话题，其中包括棋类游戏人工智能的研究。传统的棋类游戏人工智能多使用监督学习的方式让人工智能来模仿人类的决策。监督学习往往需要大量的数据集来训练。然而在实际情况下，这种数据集往往是很难获取到的，成本很高的，或者不可靠。即使你拿到了这样的数据集，人工智能的上限也会被数据集的质量所限制。相比起来，增强学习更可靠，只需要从自己的经验中获取知识。并且人工智能在增强学习中获取的知识可以超出人类的认知。近几年，人工智能 AlphaGo 在围棋游戏人工智能中取得了突破的进展。最近在论文 *Mastering the game of Go without human knowledge[1]*中，实现了 AlphaGo Zero 版本，这是首次纯靠增强学习实现的围棋游戏人工智能，并击败了 AlphaGo 的所有其它版本。随后在论文 *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm[2]*中对这种纯增强学习算法进行了推广，提出了 AlphaZero，实现了国际象棋和将棋的人工智能，并击败了这两种游戏当前最强人工智能。

传统类似的博弈游戏人工智能一般都是基于规则实现的，这种人工智能实现起来需要大量特定领域的知识，没有通用性，一旦要解决的问题规则发生变化，绝大部分逻辑都要重构，并且人工智能最终的智商还不一定足够高，想要得到一个智商十分高的人工智能需要更加复杂的程序来实现，实现效率低。

本篇论文会详细分析 AlphaZero 的增强学习算法，并实现一个简单的 AlphaZero 框架，此框架可以轻松实现任何类似的博弈游戏人工智能。通过此框架可以极大简化类似的博弈游戏人工智能开发过程，用户只需要实现游戏规则和定义特定的神经网络，使用 AlphaZero 深度学习算法，不需要除了基本游戏规则以外的任何特定领域知识，便可以得到这个游戏几乎任何水平的人工智能，可以方便的直接投入实际使用。

AlphaZero 增强学习算法

神经网络

使用一个神经网络(p,v)=f(s)。其中输入 s 是棋盘当前状态，一般用一张图像表示；输出 p 是一个矢量，表示每种动作的概率，游戏的动作总数量就是这个矢量的维度；输出 v 是一个在[-1, 1]范围内的值，代表当前棋局的胜率。这个单一网络被称为策略价值网络，主要用于辅助蒙特卡洛树搜索。

在早期的 AlphaGo 版本中，策略网络和价值网络是分开的两个神经网络，在 AlphaGo Zero 版本中首次提出了将两个网络合并为一个网络，并验证了合并后的一个策略价值网络的各方面性能要优于使用分开的策略网络和价值网络，实验验证详见论文[1]。因此，我仅考虑使用合并后的单个策略价值网络的方法。

蒙特卡洛树搜索

使用由神经网络所引导的蒙特卡洛树搜索算法。蒙特卡洛树的作用是在电脑每次考虑执行什么动作时，进行若干次蒙特卡洛树搜索，最后由蒙特卡洛树输出每个动作的概率，动作的概率越大表示执行这

个动作对自己的优势越大。算法详细过程如图 1 所示。

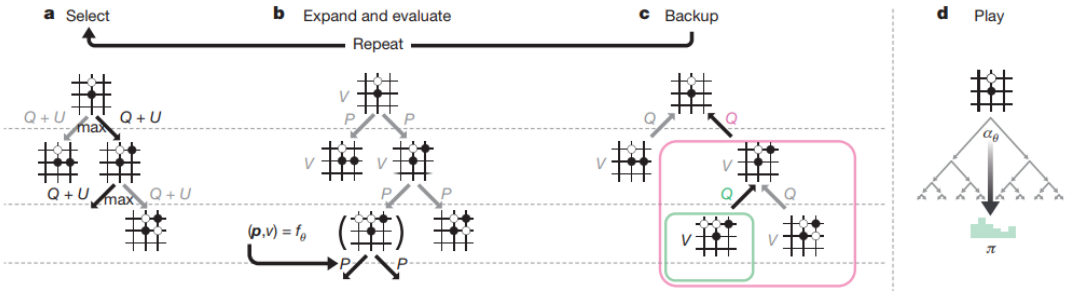


图 1 蒙特卡洛树搜索

每个节点存四个值：访问次数 N，总动作值 W，平均动作值 Q，走此步概率 P。每次蒙特卡洛树搜索算法开始时，整个蒙特卡洛树只有一个根节点，执行步骤 a~c 若干次。

a. 选择。从根节点开始，不断选择 Q+U 值最大的节点，直到找到一个叶子节点。叶子节点是指没有子节点的蒙特卡洛树节点。其中，$U(s,a)=c_{puct}P(s,a)\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$。$c_{puct}$是一个常量，控制探索的广度和深度。

b. 展开叶子节点。叶子节点的棋盘状态 s 带入网络(p,v)=f(s)进行计算，得到每个动作 a 的概率p_a。对于每个有效的动作 a 产生一个新的叶子节点，节点的值初始化为 N=0, W=0, Q=0, P= p_a。有时候被展开的叶子节点没有子节点了，即这个叶子节点是棋局的终态，没有任何有效动作，此时这个叶子节点被展开后仍然是叶子节点，并且不会产生新的叶子节点。

c. 原路返回更新每个节点的值。对于每个走过的节点，更新访问次数 N=N+1，更新总动作值 W=W+v，重新计算平均动作值 Q= W/N。

d. 执行蒙特卡洛树搜索步骤 a~c 若干次后，在根节点选择这一步的动作。根节点的棋盘状态是s_0，计算每个动作的概率矢量π，$\pi(a|s_0)=\frac{N(s_0,a)^{\frac{1}{t}}}{\sum_b N(s_0,b)^{\frac{1}{t}}}$。然后按每个动作 a 的概率$\pi(a|s_0)$来选择动作。其中 t 是一个温度值，控制动作选择的广度，当 t=1 时，选择每个动作的概率与这个动作对应的节点的访问次数成正比，当 t 取一个很小的值的时候（例如 t=0.001），相当于选择被访问次数最多的节点对应的动作。一般在训练过程中，t 可以取 1，增大动作选择广度，然后在实际使用的时候，t 可以取一个很小的值，增强实力发挥的稳定性。自我对局过程中，每步选择一个动作后，选择的动作对应的节点成为蒙特卡洛树新的根节点，根节点及其其它子节点全被舍弃，这棵树在下次考虑执行什么动作时被继续使用。

增强学习算法

神经网络训练方式描述如下：

1.每步走棋进行多次由神经网络的输出引导的蒙特卡洛树搜索，由蒙特卡洛树输出一个每步 a 的走棋概率矢量 π 。

2.根据最后输赢情况给出每步对应的 z 值。此步对应的玩家在这局赢了，z 为 1，如果输了，z 为-1，如果平局，z 为 0。

3.每收集很多自我对弈产生的数据(s, π , z)后，要神经网络(p, v)=f(s)的(s, p, v)

去拟合(s, π , z)。

4.损失函数定义为（θ是神经网络的参数）：$loss = (z - v)^2 - \pi^T \log p + c\|\theta\|^2$。第一项是 z 和 v 的均方误差，第二项是$\pi$和 p 的交叉熵误差，第三项是 L2 正则项防止过拟合。

AlphaZero 的增强学习算法过程伪代码描述如下：

```
1 初始化对局数据库集
2 初始化神经网络参数
3 n = 0, maxN = 1e9
4 while (n <= maxN):
5     自我对局
6     收集对局数据库
7     训练神经网络
8     n = n + 1
```

增强学习算法主体部分是一个循环，每次循环主要做以下三件事情：

1.自我对局。不断使用最新的神经网络来进行自我对局。

2.维护一个对局数据集库。使用最新的神经网络配合蒙特卡洛树搜索算法持续进行自我对局，收集每步对局数据，放入数据集库中，并不断从数据集库中删掉最老的对局数据，保证对局数据不会太过时。

3.训练一个神经网络。不断的使用数据集库来训练并更新策略价值神经网络。最初的时候，神经网络的参数是随机的，所以一开始对局时会类似于随机的选择动作。神经网络会在自己的对局中不断获得经验，对局势的评估值和每个动作的执行概率会越来越准。

一般处理神经网络有两种方式。第一种方式是运用于 AlphaGo Zero 的：保存两个版本的神经网络，当前神经网络和最强神经网络，两个网络只有参数不同，在增强学习不断自我对局过程中，只由最强神经网络来进行自我对局，收集对局数据，然后只让当前神经网络在收集的对局数据集上进行训练更新神经网络参数，每经过一段时间，用当前神经网络和最强神经网络进行多次对局来评估强度，如果当前神经网络强于最强神经网络，则更新最强神经网络的参数为当前神经网络的参数。第二种方式是运用于 AlphaZero 的：只保留一个当前神经网络，无论是自我对局还是训练网络更新参数都只是在当前神经网络上进行的。方式一更稳定但是会消耗一部分时间在两个网络的评估过程中，方式二速度更快而且在实际实验中我发现当前神经网络一般来说并不会比最强神经网络差很多。所以我目前是使用方式二实现的。

AlphaZero 框架实现

环境

使用 Python 程序语言。Python 是一门简洁的，易读的，可扩展的编程语言，广泛应用于科学研究，同时也是被人工智能领域最广泛使用的语言。比较流行的许多深度学习库都有 Python 接口，我主要使用了谷歌的 TensorFlow[3]。

编码运行过程都是在我的台式机上进行的，环境的参数如表 1 所示：

表 1 实验环境

处理器	Intel Core i5-6500 CPU @ 3.2GHz
内存	16GB RAM
操作系统	Windows 10 专业版 64 位
编译程序	PyCharm 2018.2.4 64 位
编程语言	Python 3.6 64 位

编码实现

使用 python3.6，实现了一个简单的 AlphaZero 框架，包括如下内容：

1.抽象的游戏规则：定义了一套游戏规则接口。

2.抽象的模型类：定义了模型的接口，包括神经网络的定义，保存读取，训练方法等。

3.蒙特卡洛树：实现了一个蒙特卡洛树及其搜索算法。

4.训练类：对 AlphaZero 增强学习算法过程的实现，其程序流程如图 2 所示。

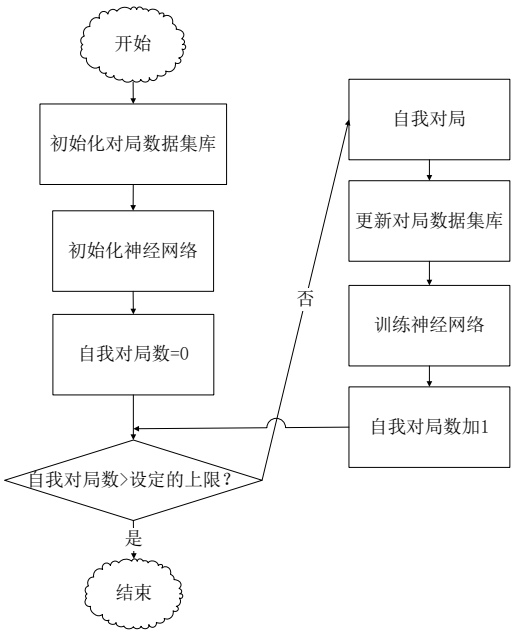


图 2　训练流程

5.工具类：模型的打包，训练过程作图等方法。

此框架的使用：

1.实现一个具体的游戏规则。

2.实现一个具体的模型。

3.将游戏规则和模型传入训练类进行训练，调参。最后得到训练好的模型。

4.使用工具类画图分析训练过程或将模型打包导出为单个文件。

源代码已上传至 github，网址为 https://github.com/SSSxCCC/AlphaZero-In-Unity。

框架评估

实现一个井字棋游戏，测试框架的正确性，易用性以及开发效率。井字棋是一种在 3x3 棋盘上进行的连珠游戏，由于棋盘一般不画边框，格线排成井字故得名。两位玩家轮流在格子里留下标记，任意三个标记形成一条直线即获胜。如果双方玩家都不失误，每步做出最好的选择，则最后只会平局。具体实现过程如下：

1.编写游戏规则。总共有 9 种动作，分别是在棋盘上的 9 个格子标记，两个玩家轮流在棋盘上标记，被标记的位置对应的动作被视为无效动作。游戏开始时。每次任意玩家执行动作后，判定游戏是否结束，如果有某个玩家在水平方向、垂直方向、正斜方向或反斜方向有 3 个标记形成了一条线，则此玩家获胜，如果已经没有未被标记的格子了，则这局判定为平局。

2.定义神经网络。输入是 3 个 3x3 的二元特征平面：第一个平面描述己方标记分布情况，有己方标记的位置是 1，否则是 0；第二个平面描述对方标记分布情况，有对方标记的位置是 1，否则是 0；第三个平面描述己方是先手还是后手，如果是己方是先手，则 3x3 平面全为 1，否则全为 0。输出分成两部分：单个价值值和一个大小为 9 的策略矢量。整体网络结构描述如下：

(1.1) 卷积层，32 个 3x3 过滤器，步长为 1。

(1.2) ReLU 激活函数

从(1.2)连接到策略输出部分：

(2.1) 卷积层，2 个 1x1 过滤器，步长为 1。

(2.2) ReLU 激活函数

(2.3) 全连接层，输出是大小为 9 的矢量。

(2.4) softmax

从(1.2)连接到价值输出部分：

(3.1) 卷积层，1 个 1x1 过滤器，步长为 1。

(3.2) ReLU 激活函数

(3.3) 全连接层，输出是大小为 32 的矢量。

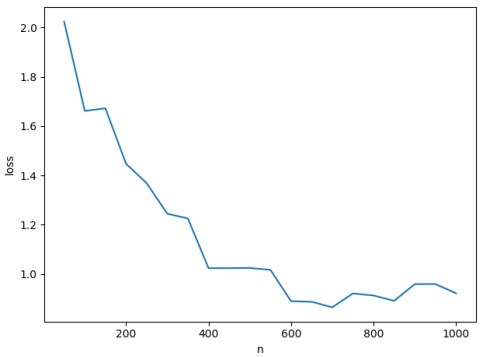
(3,4) ReLU 激活函数

(3.5) 全连接层，输出是大小为 1 的矢量。

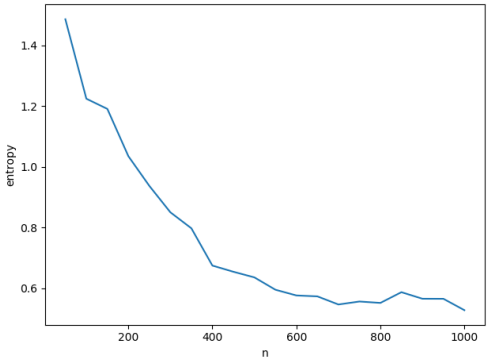
(3.6) tanh 激活函数

3.开始训练。由于井字棋的棋盘满足旋转和翻转等价规则，所以每收集到的 1 个数据集经过 4 次旋转，4 次翻转后扩增为 8 个数据集，极大提升数据集收集效率。

4.分析训练过程。神经网络的训练过程损失函数的值会随着训练的进行而逐渐减小，观察损失函数的值在训练过程中是否逐渐减小可以确保神经网络的训练正常的进行，如果损失函数的值不能随着训练的进行而减小则需要对训练设定的参数值调整后再重新运行。在增强学习过程中，人工智能进行决策的熵值也应该随着训练的进行而降低，即一开始由于人工智能基本上什么都不知道，动作完全不确定，熵值就会很大，但是随着增强学习的进行，人工智能应对局势的决策应该会有一定的把握，熵值会降低，这个熵值在 AlphaZero 算法里面可以是策略网络输出的每个动作的概率矢量的概率熵。训练过程观察损失函数值和熵值（策略矢量的概率熵）的变化，如图 3 所示：



a.损失函数值与训练局数的关系

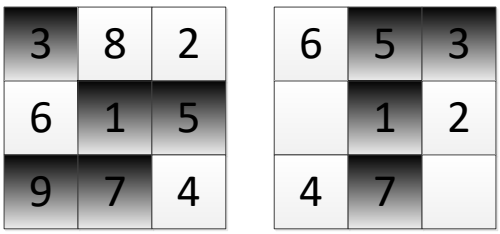


b.熵值与训练局数的关系

图 3 损失函数值和熵值和训练局数的关系

可以看到损失函数值和熵值确实如预期随着训练过程而降低。

5.分析训练结果。人工智能在增强学习过程中和自己对局 1000 盘，耗时 56 分钟。最终我与 AI 对局 17 盘。其中 9 局我先手，第一个标记分别留在 9 个不同的位置，结果全是平局； 8 局我后手，人工智能先手每次一定会标记在棋盘正中间，4 局我第一步标记在边上最终我输，4 局我第一步标记在角上最终平均。几种典型的对局棋谱如图 4 所示。所以可以看出，经过训练的人工智能已经完全会玩井字棋游戏了。



a.玩家先手，最终平局 b.玩家后手走边，玩家输

图 4 与人工智能典型的对局举例

参 考 文 献

[1] Mastering the game of Go without human knowledge. Nature volume 550, pages 354–359 (19 October 2017)

[2] Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. ArXiv (5 Dec 2017)

[3] TensorFlow官网：https://tensorflow.google.cn/。

附 录

此文章实现的项目的源代码在github上的网址：

https://github.com/SSSxCCC/AlphaZero-In-Unity