将 AlphaZero 算法应用于五子棋的教程

谢铮*†, 付星宇*†, 虞锦源*†
*似然科技
†中山大学

{xiezh25, fuxy28, yujy25}@mail2.sysu.edu.cn

***Abstract*—这篇论文旨在作为理解和运用 AlphaZero 所用算法的详实的教程，同时我们将介绍把该算法应用于 8×8 五子棋的实践中并训练出拥有与人类相当的对弈实力的 AI 的方法。**

***Index Terms*—AlphaZero; 强化学习; 深度学习; 蒙特卡洛树搜索; 五子棋**

I. 论文简介

谷歌旗下的 DeepMind 公司近期公开的 AlphaZero 的算法 [1][2]，在没有接受任何与特定对弈策略与战术有关的知识的前提下，通过从零开始的自我对弈式强化学习，使 AI 在围棋、国际象棋、将棋等棋类项目中掌握登峰造极的技法，展现了通用人工智能的巨大潜力，成为整个行业的圣杯。

在这篇论文中，我们将总结 AlphaZero 算法的核心点并将其整合为一套渐进的教程供参照和学习。在此基础上，我们进一步将该算法在五子棋项目中复现，同时我们将完整的代码在 Github 上开源[1]。

如果你希望对原算法有更加详尽的理解，我们强烈建议你阅读原文。

II. AlphaZero 算法的构成

A. 关于 *AlphaZero* 各组成部分的仿生学解释

AlphaZero 的决策系统由两部分组成，分别是用于评估策略和行动价值的神经网络和蒙特卡洛搜索树 [6]，两者在仿生学的意义上都有比较直观的解释。

用于评估策略和行动价值的神经网络就好比人的大脑，它可以通过观察当前的局势产生一个直觉上的先验判断。而蒙特卡洛搜索树则模拟人的思考，预判在此局势基础上采取不同行动可能产生的不同结果。

最终采取的真实行动是严格基于蒙特卡洛搜索树生成的模拟结果的，而不是来自于将神经网络作用在当前局面生成的先验判断，这就好比一个真正理性的人不会根据直觉感受就妄下判断，而是依据深入的分析和思考才会做出最终的决策。

在以下的小节中，我们将更具体的阐释以上的每个环节及相应的技术细节。

B. 用以评估策略与行动价值的神经网络

策略-行动价值网络 $f_{\boldsymbol{\theta}}$，以 $\boldsymbol{\theta}$ 为参数，接受一个张量 $\boldsymbol{s}$ 的输入，分析当前的局面并生成一个先验概率分布的向量 $\vec{\boldsymbol{p}}$ 和评估落子方在当前局面上胜率的标量 v。

形式上，我们可以写成如下的等式：

$$(\vec{\boldsymbol{p}}, v) = f_{\boldsymbol{\theta}}(\boldsymbol{s}) \tag{1}$$

具体而言，在我们五子棋的项目中，$\boldsymbol{s} = [\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{L}, \boldsymbol{C}]$ 是一个 $4 \times n \times n$ 的张量，其中 n 指代的是棋盘的长或宽而 4 是通道的数量。$\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 是由带有表明当前落子方和其对手棋子颜色的两组数组成 (其中，X_i 等于 1 意味着棋盘上第 i 个交点被当前落子方的棋子占据；相反，倘若该交点被其对手的棋子占据或尚未落子，则 X_i 等于 0。Y_i 的指向对象可以和 X_i 类比)。$\boldsymbol{L}$ 通道代表的是上一步落子的位置，每个位置的取值为 0 或 1，$L_i = 1$ 当且仅当第 i 个位置为对手上一步落子的位置，否则取值为 0。此外，$\boldsymbol{C}$ 是取值恒为 1 或 0 的颜色通道，分别对应当前落子方为黑方和白方的情形。向量 $\vec{\boldsymbol{p}} = (p_1, p_2, ..., p_{n^2})$ 的每个元素代表了在当前局面的基础上落子在第 i 个位置的先验概率。$v \in [-1, 1]$ 代表了当前局面的落子方的获胜几率，v 越大意味着神经网络认为当前落子方有更高的获胜可能。

我们在这里选用的神经网络是带有残差块 [8] 的深度卷积神经网络 [9]，因为它可以很好的解决由网络过深而造成的退化问题。此外，为了进一步提升神经网络的性能，我们还加入了规范层 [10] 等机制，整体的架构主要由如下的三部分构成：

- 残差塔: 接收原始局面张量的输入并进行高维的特征提取。残差塔的输出将分别进入策略分支和行动价值分支进行进一步的处理。
- 策略分支: 通过 Softmax 激活函数生成先验概率分布的向量 $\vec{\boldsymbol{p}}$。
- 行动价值分支: 通过 tanh 激活函数生成评估获胜几率的标量 v。

我们的网络是在 python 上使用以 TensorFlow[2] 为后端的 Keras 模块[3] 实现的，详细的拓扑结构在如下的表 [I], 表 [II] 和表 [III] 中予以呈现：

C. 蒙特卡洛树搜索 *MCTS*

MCTS $\alpha_{\boldsymbol{\theta}}$，在策略-行动价值网络 $f_{\boldsymbol{\theta}}$ 的指导下，接收一个当前局面的张量 $\boldsymbol{s}$ 的输入并通过大量的模拟输出落子方所有可采取的行动的样本分布向量 $\vec{\boldsymbol{\pi}} = (\pi_1, \pi_2, ..., \pi_{n^2})$。形式上，我们可以写成如下的等式：

$$\vec{\boldsymbol{\pi}} = \alpha_{\boldsymbol{\theta}}(\boldsymbol{s}) \tag{2}$$

和之前由“直觉”生成的先验概率分布向量 $\vec{\boldsymbol{p}}$ 相比，行动向量 $\vec{\boldsymbol{\pi}}$ 包含的信息更有价值因为它来自于蒙特卡洛搜索树

[1] https://github.com/PolyKen/AlphaRenju_Zero

[2] https://www.tensorflow.org/

[3] https://keras.io/

层架构
(1) 包含 32 个 3×3 卷积核且步长为 1 的卷积层
(2) 批标准化
(3) 线性整流函数激活
(4) 包含 32 个 3×3 卷积核且步长为 1 的卷积层
(5) 批标准化
(6) 线性整流函数激活
(7) 包含 32 个 3×3 卷积核且步长为 1 的卷积层
(8) 批标准化
(9) 跨层连接
(10) 线性整流函数激活
(11) 包含 32 个 3×3 卷积核且步长为 1 的卷积层
(12) 批标准化
(13) 线性整流函数激活
(14) 包含 32 个 3×3 卷积核且步长为 1 的卷积层
(15) 批标准化
(16) 跨层连接
(17) 线性整流函数激活

表 I
残差塔

层架构
(1) 包含 2 个 1×1 卷积核且步长为 1 的卷积层
(2) 批标准化
(3) 线性整流函数激活
(4) 展平层
(5) 输出为 n^2 维向量的全连接层
(6) Softmax 函数激活

表 II
策略分支

层架构
(1) 包含 1 个 1×1 卷积核且步长为 1 的卷积层
(2) 批标准化
(3) 线性整流函数激活
(4) 展平层
(5) 输出为 32 维向量的全连接层
(6) 线性整流函数激活
(7) 输出为标量的全连接层
(8) 双曲正切函数激活

表 III
行动价值分支

对于在当前局面基础上采取不同行动可能产生的不同结果进行的综合评估。因此，行动向量 $\vec{\boldsymbol{\pi}}$ 在 AlphaZero 算法中扮演了目标策略的角色，而真实的落子行动却是依照三种类型的策略而决定的：

- 完全随机策略：AI 依照行动分布 $\vec{\boldsymbol{\pi}}$ 进行随机采样从而决定下一步的落子位置，即：

$$a \sim \vec{\boldsymbol{\pi}} \tag{3}$$

- 确定性策略：AI 直接选取当前被探索次数最多的落子位置，即：

$$a \in \arg\max_i \vec{\boldsymbol{\pi}} \tag{4}$$

- 半随机策略：在每一局棋的开始若干步，AI 依照行动分布 $\vec{\boldsymbol{\pi}}$ 进行随机采样决定下一步的落子位置。但随着局势的推进，在给定数目的回合后，AI 选用确定性策略进行落子决策。

关于它们各自的适用情境，我们将在后文详细介绍。

和类似黑箱的神经网络相比，我们对于蒙特卡洛树搜索的运作逻辑有清晰的把握。对于一个搜索树，每一个节点代表一个局面，每条由该节点分叉的边代表了在此局面上可以采取的所有可能的行动。每条边保存了如下的统计量：

- 访问次数，N：指的是访问这条边的总次数。N 越大代表在树搜索的模拟中这条边被选中的次数越多。事实上，给定局面 $\boldsymbol{s}$ 下的行动概率分布 $\vec{\boldsymbol{\pi}}$ 是综合每条边的访问次数 $N(\boldsymbol{s},k)$ 的函数，具体定义如下：

$$\pi_i = N(\boldsymbol{s},i)^{\frac{1}{\tau}} / \sum_{k=1}^{n^2} N(\boldsymbol{s},k)^{\frac{1}{\tau}} \tag{5}$$

  其中 τ 是权衡取舍纵向探索和横向探索两者偏好的温度系数，如果 τ 是一个非常小的量，那么经由指数的运算会放大不同边的探索次数 $N(\boldsymbol{s},k)$ 之间的差异因而减少探索别的尚未被充分探索的边的可能，即有意地选择访问次数较多的边。
- 先验概率，P：指的是神经网络作用在该边的根节点，即当前局面 $\boldsymbol{s}$ 后输出的先验概率。更大的 P 表明神经网络更偏好这一步棋并因此直接导致蒙特卡洛树对其进行纵向的深入探索。当然，必须要明确的一点是，更大的 P 并不能确保这是一步好棋因为它仅来源于一次先验的判断，就好比是人一刻的直觉。
- 行动价值的均值，Q: 指的是每一条边的行动价值总和与该边的总访问次数的商，具体我们按以下的形式定义：

$$Q(\boldsymbol{s},i) = \frac{1}{N(\boldsymbol{s},i)} \sum_{v \in A} \pm v \tag{6}$$

  其中 $A = \{v | (\_, v) = f_{\boldsymbol{\theta}}(\hat{\boldsymbol{s}})$，$\hat{\boldsymbol{s}}$ 取遍当前节点 $\boldsymbol{s}$ 的子树的每一个节点$\}$。这里比较耐人寻味的地方就是加减号“$\pm$”。v 是在 $\hat{\boldsymbol{s}}$ 局面下落子方的获胜几率，而并不仅仅是当前局面 $\boldsymbol{s}$ 下的落子方的胜率。然而这里的 Q 衡量的是当前局面 $\boldsymbol{s}$ 下选择某一行动的获胜几率，因此我们需要相应的调整 v 的符号来更新 Q 的值。
- 总行动价值，W：指的是在某一局面下选择某一行动产生的子树的每一个叶节点的 v 值的加总。W 只是一个用来传递和更新 Q 的中间变量，满足 $Q = W/N$ 的关系。

在真正的采取某一行动前，MCTS 将依据当前局面 $\boldsymbol{s}$ 多次模拟并预判采取不同可能的行动后产生的相应的结果，每一次的模拟由以下三个步骤决定：

- 选择：对于从根局面 $\boldsymbol{s}$ 开始到第一个叶节点为止的所有局面 $\hat{\boldsymbol{s}}$，MCTS 在模拟和迭代后选择行动 j 的依据都是使得下式取到最大值：

$$j \in \arg\min_k \{Q(\hat{\boldsymbol{s}},k) + U(\hat{\boldsymbol{s}},k)\} \tag{7}$$

  其中，$Q(\hat{\boldsymbol{s}},j) + U(\hat{\boldsymbol{s}},j)$ 被称为上限置信区间且 U 满足：

$$U(\hat{\boldsymbol{s}},j) = c_{puct} P(\hat{\boldsymbol{s}},j) \frac{\sqrt{\sum_k N(\hat{\boldsymbol{s}},k)}}{1 + N(\hat{\boldsymbol{s}},j)} \tag{8}$$

  c_{puct} 是一个控制搜索树探索程度的常量。设置上限置信度的目的是平衡行动价值均值 Q，先验概率 P 和

访问次数 N 这三者的关系。从直观上而言，MCTS 倾向于选择访问次数较高，先验概率较高并且能带来尽可能大的行动价值均值的行动。

- 评估与扩展：一旦 MCTS 遇到叶节点，比方说 $\tilde{s}$，同时该节点尚未被神经网络作用并评估过。这时，我们将该局面输入神经网络 $f_{\boldsymbol{\theta}}$ 从而生成该局面的获胜几率 v 和采取各行动的先验概率 $\vec{\boldsymbol{p}}$。然后，我们生成该叶节点 $\tilde{s}$ 的每一条边并将第 i 条边的各统计值初始化为 $N(\tilde{s},i)=0, Q(\tilde{s},i)=0, W(\tilde{s},i)=0, P(\tilde{s},i)=p_i$。
- 回溯：在我们完成评估和扩展后，我们反向的沿着搜索树的“枝干”一直回溯到根节点为止，在这个过程中，我们每经过一条边都相应更新该边的各统计值，具体方法为：$N=N+1, W=W\pm v, Q=W/N$。在这里，加减号”±” 意味着采用方程 [5] 的方法。

图 [1] 描绘了 AlphaZero 算法做决策的整个流程。

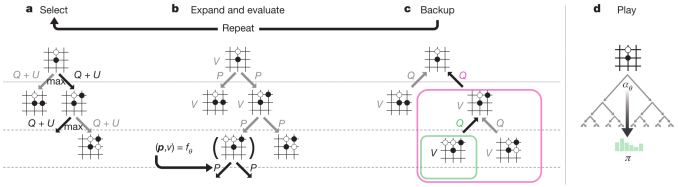

图 1. AlphaZero 算法的决策流程 [1]

在这里，对于死节点的处理需要特别关注，即平局或已经分出胜负的局面，这些局面的特点是没有子节点。假设在 MCTS 的模拟中遇到的死节点，我们要求该搜索树直接进行回溯的操作而并非执行评估和扩展的操作。其中，如果是分出胜负的局面，那么我们将其 v 值以 -1 进行回溯，否则若是平局，则以 0 回溯。

D. 策略-行动价值网络的训练目标

AlphaZero 算法的成功与否取决于由网络生成的先验概率 $\vec{\boldsymbol{p}}$ 和获胜几率 v 能否将 MCTS 很好的引导到一个更小的却包含更多有价值的行动的搜索空间中。先验概率 $\vec{\boldsymbol{p}}$ 和获胜几率 v 将直接影响 MCTS 的决策质量，因此我们需要采用一定的手段来训练网络从而提升它的预测能力。我们预设了如下三个评判标准来衡量一个网络是否具有强大的性能：

- 它可以准确地预测最终的获胜方。
- 它生成的先验概率分布 $\vec{\boldsymbol{p}}$ 应当趋近 MCTS 经过模拟及审慎挑选后的策略分布 $\vec{\boldsymbol{\pi}}$。
- 这个模型具有适应性强的泛化能力。

为了达成这些目标，我们运用带动量 [11] 的学习率退火的随机梯度下降来尽可能最小化如下的损失函数。

$$L(\boldsymbol{\theta})=(z-v)^2-\vec{\boldsymbol{\pi}}^T log\vec{\boldsymbol{p}}+c\|\boldsymbol{\theta}\|^2 \tag{9}$$

其中 c 是控制 L_2 惩罚的参数，z 每一盘棋的真实胜负结果，即：

$$z=\begin{cases}1 & \text{若当前落子方最终取胜}\\ 0 & \text{若对弈双方以平局收尾}\\ -1 & \text{若当前落子方最终输棋}\end{cases} \tag{10}$$

由此可见，损失函数的三个组成部分分别对应了上文提到的三个评判标准的要求。

III. 训练方法

A. 自我对弈式的强化学习

AlphaZero 算法最引人入胜的地方就在于它可以在没有人类经验教学的条件下通过自我对弈和摸索的方法训练 AI 使之精通例如围棋、国际象棋、将棋等博弈游戏。因此，AlphaZero 算法可以称得上是通用 AI 诞生的曙光，同时也是整个行业的圣杯。

为了实现这样的目标，我们通过自我对弈的强化学习训练神经网络，其中，我们始终保持使用共有的搜索树和同一的策略-行动价值网络并在此基础上使其运用半随机的策略进行自我对弈。在每一步落子后，搜索树都会将新的根节点移动到落子后形成的新局面处并舍弃剩余树的分支直至该局分出胜负。我们固定在每进行一定数量棋局后收集一次数据，并在数据池中均匀采样来训练网路。每一个训练样本的结构如下：

$$\{\boldsymbol{s},\vec{\boldsymbol{\pi}},z\} \tag{11}$$

同时，我们需要意识到五子棋是一个对称的游戏，因而我们可以通过对原始局面进行总计七种棋盘旋转、对称变换的方法扩大样本集。在每一次训练之后，我们将受新网络指导的 AI 与当前最强的旧网络指导下的 AI 进行对弈来衡量训练的效果。如果训练后的网络的胜率超过 0.55，那么我们采用该网络并放弃先前的网络。反之，继续对旧网络进行训练，直至更新后的网络击败当前最强的网络。图 [2] 展现了自对弈强化学习的流程。

在这里，我们着重探讨在上述过程中需要特别注意的三个问题：

- 为什么需要评估网络的环节？通过放弃性能较差的网络避免陷入局部最优。
- 为什么需要在自我对弈和网络评估的环节采用半随机的策略？首先，我们在策略选择中加入随机性从而可以探索更多的落子点及相应的结果。其次，我们希望 AI 在一定回合后落子更加谨慎从而避免产生误导性的落子选择和质量较差的数据。

另一个值得注意的有趣发现是自对弈阶段时长的变化。在训练的过程中，每局的耗时先减小继而呈现逐步增加的趋势，这个现象的出现主要是因为随着网络的演进，AI 首先学会了进攻的方法，缩短了每局的战线，继而慢慢学会防守，又将战线拉长。

如果你希望让训练完的 AlphaZero AI 和人类对弈，那么每回合根节点都需要根据人类选手的落子位置不断地更迭，这与自对弈模式下的流程有细微的区别。值得注意的是，在 AI 与人类对弈的模式中，我们让 AI 采用决定性策略使其每一步都能作出搜索范围内的最优选择。

B. 运用少量人类对弈的数据加快训练的步伐

DeepMind 公司公开的论文充分展现了在没有人类经验指导的条件下可以训练出拥有超越人类竞技水平的 AI 的无限可能。然而，即使在理论上具有可行性，但对于算力却是巨大的考验。因此，为了加快训练的步伐，我们使用少量人类的数据通过监督学习的方法初始神经网络从而极大的缩短了训练的时间。在我们的项目中，我们邀请了两位五子棋玩家对弈并记录每一步决策的相应数据：

$$\{\boldsymbol{s},\vec{\boldsymbol{\pi}},z\} \tag{12}$$

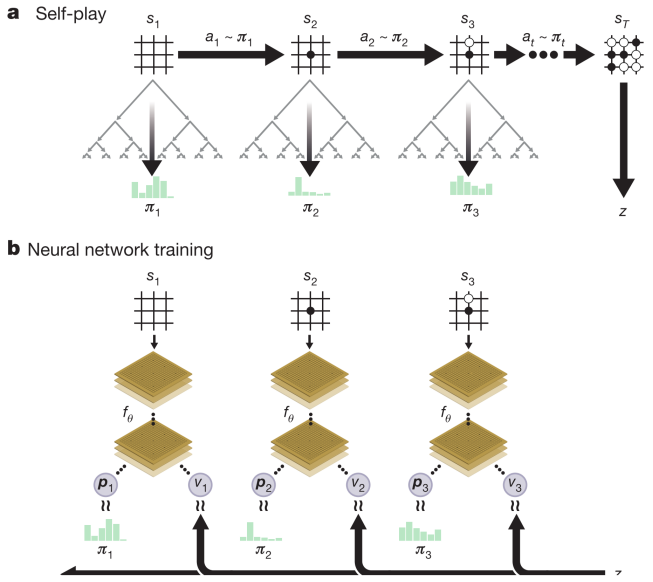

图 2. 通过自我对弈的强化学习 [1] 的流程

其中，如果某一回合内某玩家选择在第 i 个位置落子，那么相应的 $\vec{\pi} = (\underbrace{0, ..., 0, 1}_{i}, 0, ..., 0)$。

C. 异步模拟加速

限制整体训练速度的最大瓶颈是自对弈棋谱的生成过程，这是一个非常耗时的步骤因为每盘棋的每一步都对应着大量的模拟。因此，若要提升整体的速度，我们需要使模拟以并行的方式展开，具体说来，以异步的方式。然而，实现异步的模拟需要着重关注以下两个方面：

- 扩展冲突：倘若两条协程碰巧同时探索到相同的叶节点且同时肩负扩展的任务，那么该叶节点下的子节点数目会被错误统计从而导致模拟整体出错。因此，当一个协程探索到某一个叶节点时，不同于先前直接进行扩展，它会首先遍历搜索待扩展的叶节点的集合来确认当前叶节点是否已被别的协程探索到。如果确实在其中，那么该协程会稍待片刻直至最先发现该叶节点的协程完成扩展任务从而继续完成模拟。反之，该协程会直接执行扩展的任务并回溯更新各边的统计量。
- 虚拟损失：为了使各条协程能尝试尽可能多的路径，在每次选择完成后，我们会短暂地虚拟地增加被选择的边的 N 值并减少相应的 W 值从而人为地降低模拟中选择这条边的概率。DeepMind 公司将这样暂时的增加量和减少量称为虚拟损失。当然，在回溯的阶段每条边的虚拟损失要清零。
- 稀释隐患：倘若协程数目和虚拟损失预设过高，那么在模拟阶段中本该被充分探索的一步好棋的选中概率将被急剧稀释因为其他边的选择价值在好边受虚拟惩罚后相对的增加，甚至出现本末倒置的局面。因此，对于相应超参数的调试非常关键。

IV. 实验

我们将 AlphaZero 的算法最终运用于 8×8 五子棋的实践中，并成功训练出拥有与人类水平相当的五子棋 AI。

A. 人类与 AI 对弈

图 [3]、图 [4] 分别展示了一些采取决定性策略和每步 1600 次模拟的 AI 与人类对弈的棋谱。

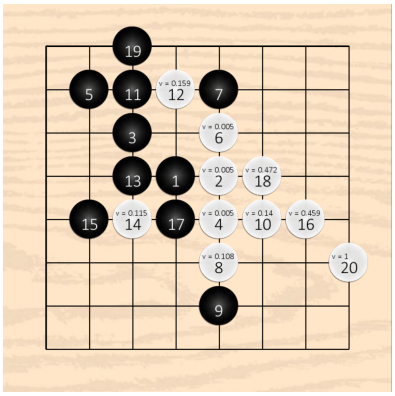

图 3. 第一回合 AI: 白棋 & Human: 黑棋

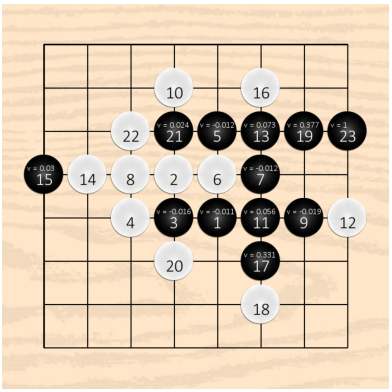

图 4. 第二回合 AI: 黑棋 & Human: 白棋

从中我们可以看出：

- AI 学会了一些进阶的五子棋战术，如“双三”、“双四”和“四三”。
- AI 面对旋转对称、镜像对称的局面采取的战术是稳定的，这归功于扩大样本集的环节。

B. AI 与 AI 对弈

图 [5]、图 [6] 分别展示了一些 AI 采取半随机策略进行自我对弈时生成的棋谱。

C. 损失函数图

图 [7] 展示了我们训练全过程损失函数随时间的变化图。

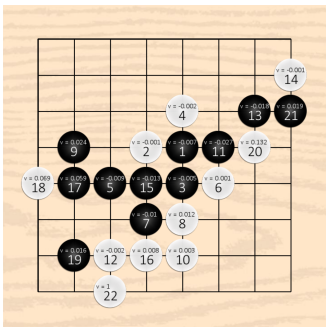

图 5. AI vs AI

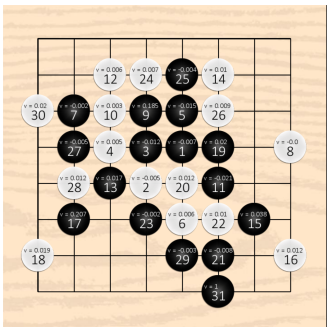

图 6. AI vs AI

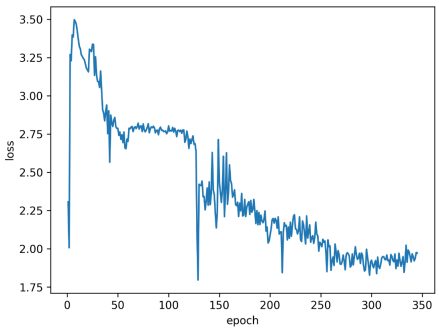

图 7. 损失函数

致谢

我们衷心的感谢来自 MIT 的陈柏安和朝旭投资管理有限公司的刘铭文在项目进行过程中给予的帮助。同时，感谢来自中山大学的梁智鹏和陈昊在训练五子棋 AI 的过程中提供的支持。正是他们的帮助，我们得以顺利完成这个艰巨的任务。

参考文献

[1] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge[J]. Nature, 2017, 550(7676): 354.

[2] Silver D, Hubert T, Schrittwieser J, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm[J]. arXiv preprint arXiv:1712.01815, 2017.

[3] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. nature, 2016, 529(7587): 484.

[4] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529.

[5] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 1998.

[6] Browne C B, Powley E, Whitehouse D , et al. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence and AI in games, 2012, 4(1): 1-43.

[7] Goodfellow I, Bengio Y, Courville A, et al. Deep learning[M]. Cambridge: MIT press, 2016.

[8] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[9] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[10] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. arXiv preprint arXiv:1502.03167, 2015.

[11] Ruder S. An overview of gradient descent optimization algorithms[J]. arXiv preprint arXiv:1609.04747, 2016.

[12] Knuth D E, Moore R W. An analysis of alpha-beta pruning[J]. Artificial intelligence, 1975, 6(4): 293-326.