✨

Master Atari, Go, Chess and Shogi by Planning with a Learning Model — hoho（MuZero）

论文试图解决什么问题？

许多的planning算法需要依赖环境的先验知识，譬如游戏的规则、工业的控制流程等。model-free的强化学习方法虽然可以不需要依赖这些知识，但需要精确和重要的预知（require precise and sophisticated lookahead）

这是否是一个新的问题？

不是新问题。已经有类似的研究：Value equivalent models。

这篇文章要验证一个什么科学假设？

能否有一种算法可以不需要依赖环境的先验知识而可以得到很好的学习？

有哪些相关研究？如何归类？谁是这一课题在领域内值得关注的研究员？

model-based方法：

1. 传统的model-based方法关注直接对环境模型建模：学习环境的状态转移概率、奖励函数等
2. 一种不同model-based方法：value equivalence，其主要思想是建立一个抽象的MDP模型，使得在这个MDP抽象模型中planning跟在真实环境的planning一致。譬如

TreeQN，Value iteration networks，Value prediction networks。这个MDP模型可以看作是深度神经网络的隐藏层，将其训练为期望累积回报跟真实环境得到的累积回报一致。

论文中提到的解决方案之关键是什么？

- 模型由三个模块组成：

1. dynamics function

这是一个循环处理过程（RNN之类），假设有K轮。在每个时间步t，有一个假设的输入上一轮k的状态和当前步动作，输出立即奖励和当前步状态：$r^k, s^k = g_\theta\left(s^{k-1}, a^k\right)$

文中还强调s_k跟真实环境的状态无关，只是模型的隐藏状态

2. prediction function

prediction function用于根据s^k计算策略和价值函数（跟AlphaGo Zero一样）：$p^k, v^k = f_\theta\left(s^k\right)$

3. representation function

根据观察输出最初的状态：$s^0 = h_\theta\left(o_1, o_2, ..., o_t\right)$

- 根据模型输出动作时，本文用的是MCTS输出策略π_t，然后动作$a_{t+1} \sim \pi_t$进行采样

- 损失函数：

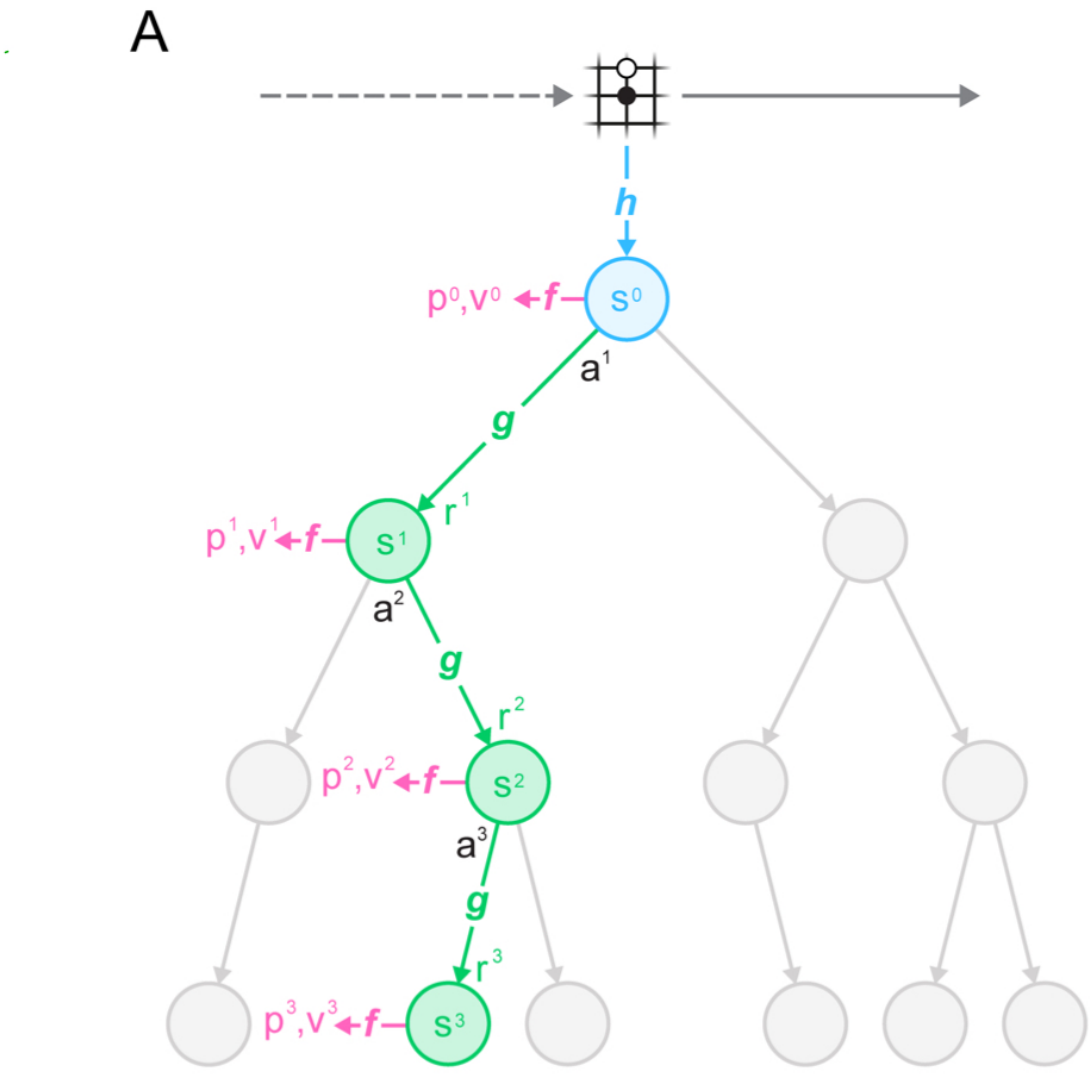
$$l_t(\theta) = \sum_{k=0}^{K} l^r\left(u_{t+k}, r_t^k\right) + l^v\left(z_{t+k}, v_t^k\right) + l^p\left(\pi_{t+k}, p_t^k\right) + c\|\theta\|^2$$

说明：

1. $l^p\left(\pi_{t+k}, p_t^k\right)$跟Alpha Zero一样

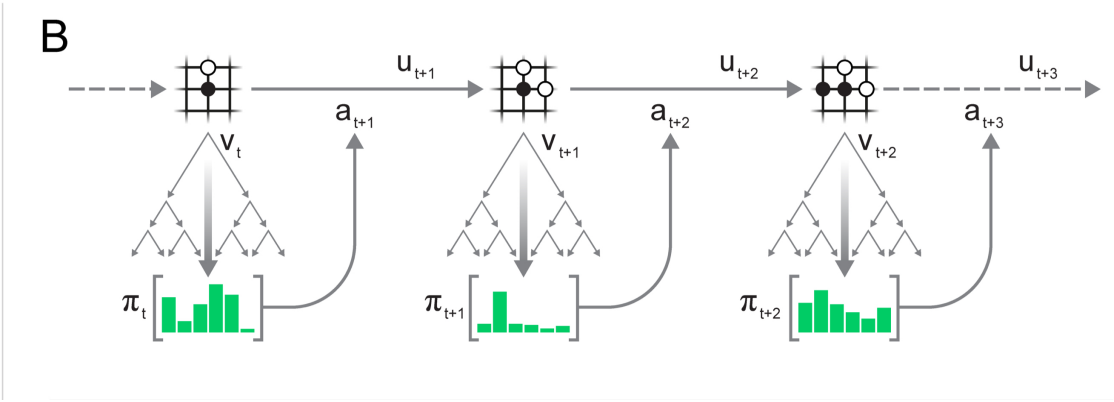
2. 对于$l^v\left(z_{t+k}, v_t^k\right)$，$z_t = u_{t+1} + \gamma u_{t+1} + ... + \gamma^{n-1} u_{t+n} + \gamma^n v_{t+n}$（最后是v而不是u），最后一步的输出根据{负、平、胜}，u_t取值为{-1, 0, +1}。另外，这部分损失建议用交叉熵

- MuZero plan的过程



- MuZero 与环境交互的过程

每个时间步进行MCTS产生策略π_t。动作a_{t+1}根据π_t生成。环境收到动作a_{t+1}后，生成观测o_{t+1}与立即回报u_{t+1}

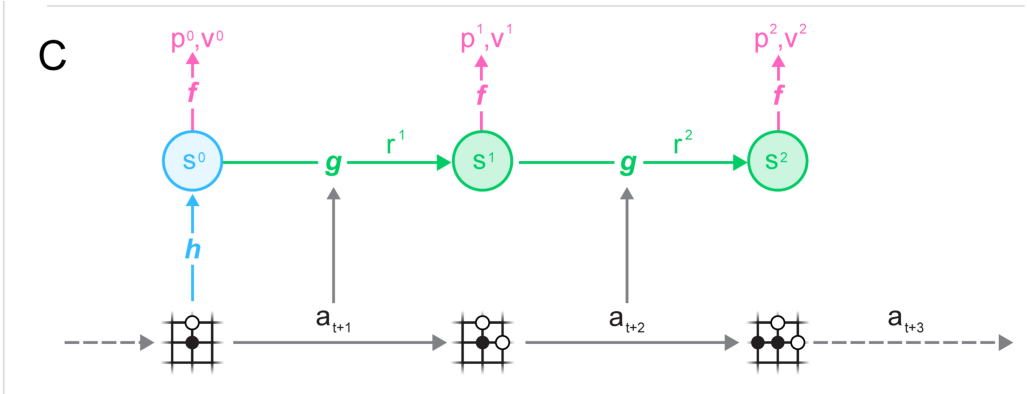


- MuZero模型的训练

1. 从经验回放池采样一条轨迹，一共t个时间步，t个观测$o_1, o_2, ..., o_t$
2. 在这条轨迹上采样K步，模型进行K轮的循环训练，

   for k in 1:K:

$$s^0 = h_\theta(o_1, o_2, ..., o_t)$$
$$r^k, s^k = g_\theta(s^{k-1}, a^k)$$
$$p^k, v^k = f_\theta(s^k)$$



论文中的实验是如何设计的？

将MuZero应用与围棋、国际象棋、日本将棋和Atari游戏

参数设置如下：

K=5

mini-batch size: 2048 for 棋类 / 1024 for Atari游戏

1 million个样本

800场模拟 for 棋类 / 50场模拟 for Atari游戏

representation function架构跟Alpha Zero一样使用卷积跟残差网络，只是残差block数量不同（MuZero为16，AlphaZero为20）

dynamics function跟prediction function跟AlphaZero一样

所有网络有256个隐藏层

用于定量评估的数据集是什么？代码有没有开源？

- 对于Atari有数据集：

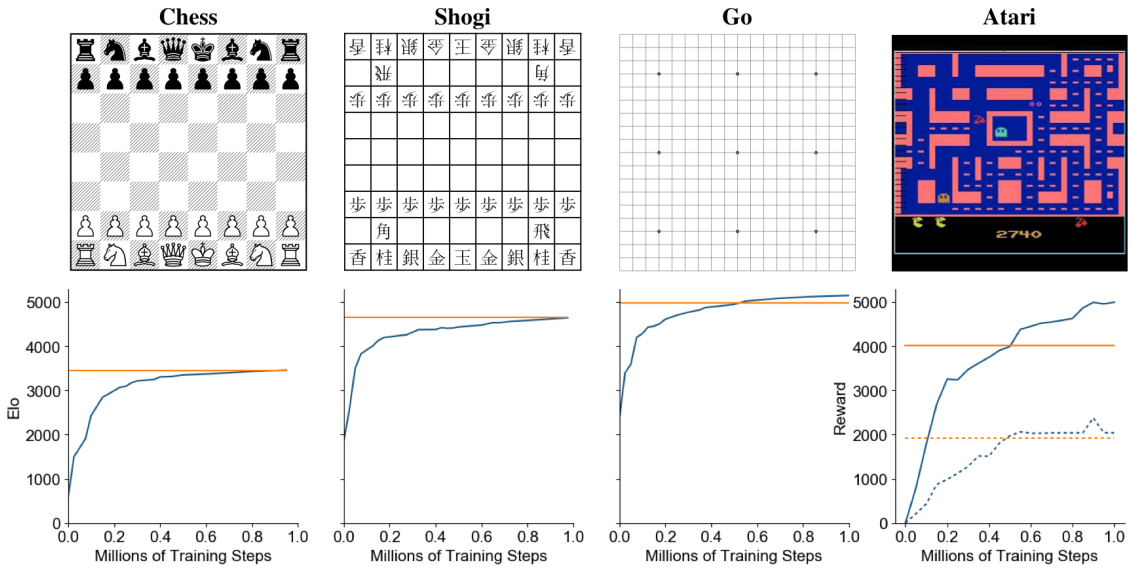
https://paperswithcode.com/dataset/arcade-learning-environment

https://paperswithcode.com/dataset/dqn-replay-dataset

- 有伪代码（https://arxiv.org/src/1911.08265v2/anc/pseudocode.py）

论文中的实验及结果有没有很好地支持需要验证的科学假设？

- 综合性能

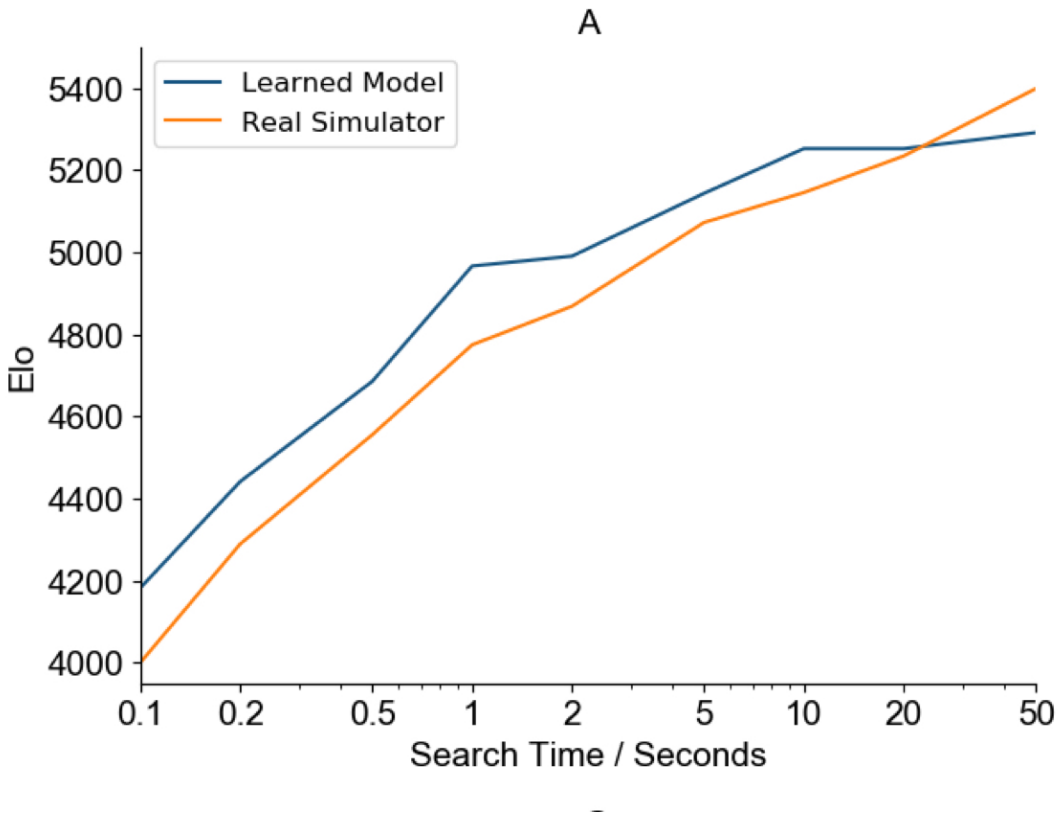


上图橙线为AlphaZero，蓝线为MuZero。

围棋MuZero轻微获胜。

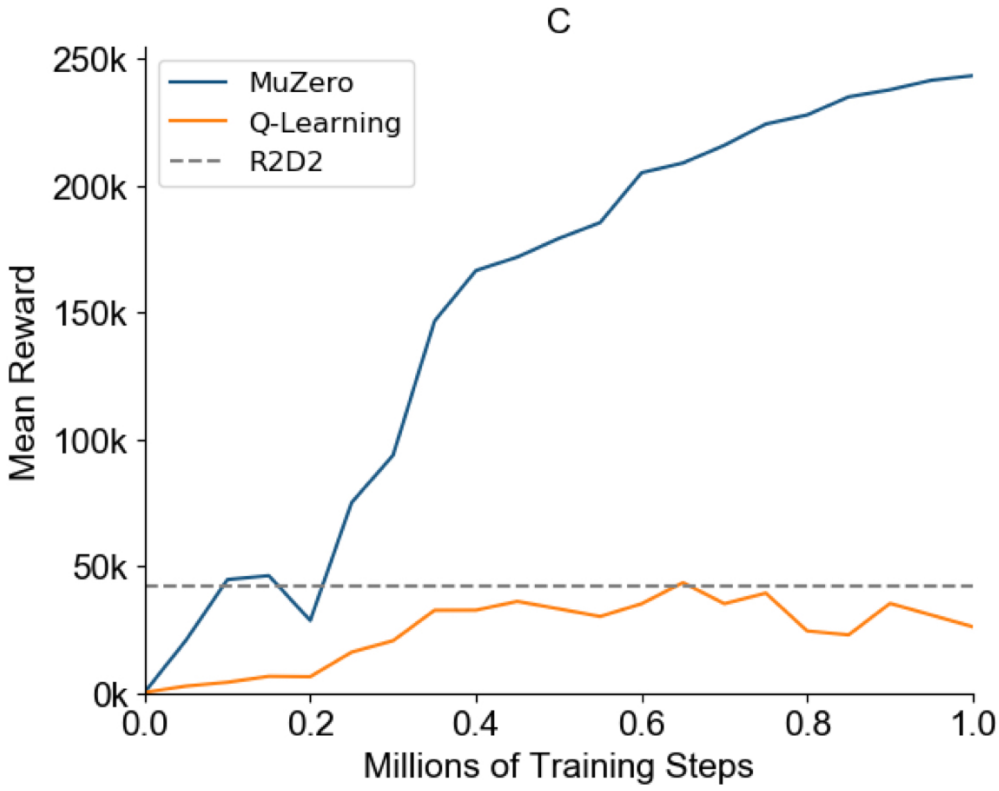
Atari获胜明显

- 关于搜索时效的对比



相同搜索时间下，MuZero更胜一筹

- 与某些model-free算法的对比



更MuZero训练目标改为跟Q-Learning一样（R2D2：参考论文”Recurrent experience replay in distributed reinforcement learning”）

这篇论文到底有什么贡献？

MuZero在高性能planning算法上超越了人类水平（如棋类游戏），也超越了某些传统model-free的擅长领域的性能（如Atari游戏），而且还不需要引入环境的任何先验知识。

下一步呢？有什么工作可以继续深入？

hoho_todo

附：

注意跟AlphaZero的不同点：

- 树中对每条边（代表action）存储的统计量有 $\{N(s,a), Q(s,a), P(s,a), R(s,a), S(s,a)\}$

N：访问次数

Q：平均价值（累积回报）

P：策略概率

R：即时奖励

S：转移的下一个状态

- upper confidence bound计算

$$a^k = \arg\max_a \left[Q(s,a) + P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)} \left(c_1 + \log\left(\frac{\sum_b N(s,b) + c_2 + 1}{c_2} \right) \right) \right]$$

其中超参c1=1.25, c2=19652

- 在树的expansion阶段

遇到叶节点时，进行expand操作。

此时用dynamic function预测回报与下一个状态：

$r^l, s^l = g_\theta(s^{l-1}, a^l)$，则变的相关统计量赋值为$R(s^{l-1}, a^l) = r^l, S(s^{l-1}, a^l) = s^l$，

而p,v输出跟Alpha Zero一样，$p^l, v^l = f_\theta(s^l)$

但此时只会生成一个新节点（对应状态为s^l），和其下的多个边（代表action），并初始化每条action边统计量$\{N(s^l, a) = 0, Q(s^l, a) = 0, P(s^l, a) = p^l\}$

- 在树的backup阶段

更新每条边的规则为：

$$Q(s^{k-1}, a^k) := \frac{N(s^{k-1}, a^k) \cdot Q(s^{k-1}, a^k) + G^k}{N(s^{k-1}, a^k) + 1}$$
$$N(s^{k-1}, a^k) := N(s^{k-1}, a^k) + 1$$

其中：

$$G^k = \sum_{\tau=0}^{l-1-k} \gamma^\tau r_{k+1+\tau} + \gamma^{l-k} v^l$$

k=l…0，G为k-l步的累积回报

- 策略的生成

$$p_\alpha = \frac{N(\alpha)^{1/T}}{\sum_b N(b)^{1/T}}$$

T为带衰减的温度系数，如训练1000k轮，前500K轮，T=1；当到500K~750K轮，T=0.5；剩下的T=0.25

- 关于动作的编码

因为g_θ需要输入动作，所以要对动作编码。

参考：

对于围棋，action编码为一个棋盘平面（称为plane），无下子动作的方格为0，下子的方格为0（pass用全为0的棋盘表示）

对于国际象棋：用8个plane编码。其中，1个plane表示下子前的位置，1个表示下子后的位置，1个表示下子后的位置是否合法（全为1或全为0），其余5个plane分别表示下的是哪个子（queen,knight,bishop,rock,none）