MuZero: Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model

梅屹东

2021年3月

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

论文阅读

- **DeepMind 2020发表在Nature论文, AlphaGo四代-MuZero**

Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model

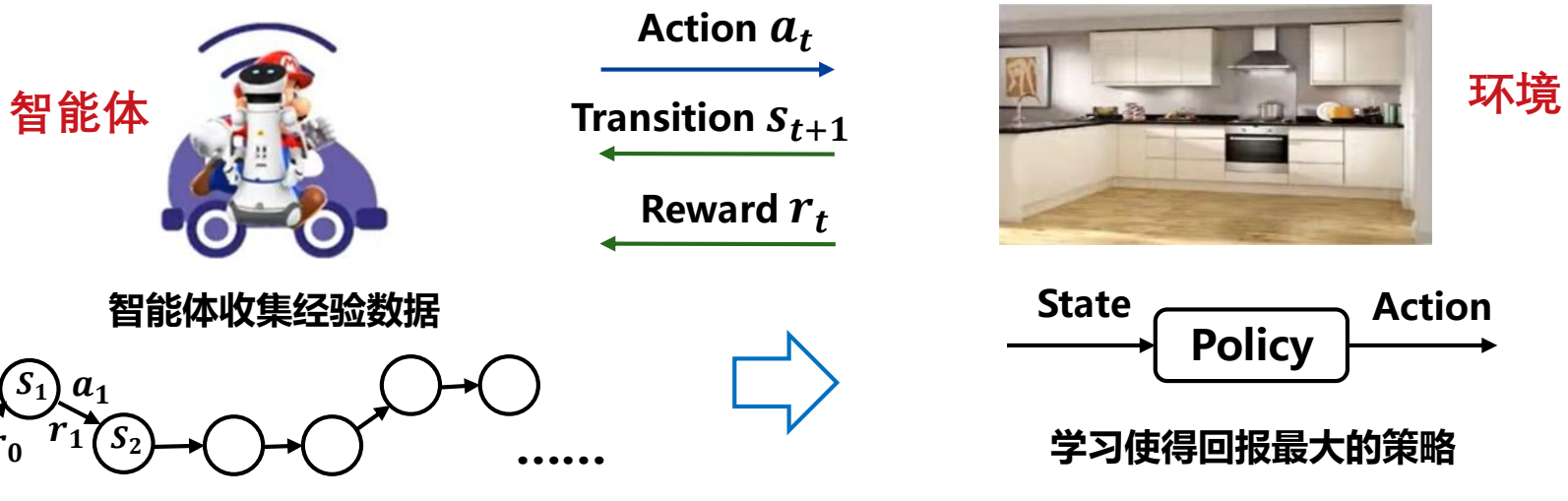
Julian Schrittwieser,[1*] Ioannis Antonoglou,[1,2*] Thomas Hubert,[1*]
Karen Simonyan,[1] Laurent Sifre,[1] Simon Schmitt,[1] Arthur Guez,[1]
Edward Lockhart,[1] Demis Hassabis,[1] Thore Graepel,[1,2] Timothy Lillicrap,[1]
David Silver[1,2*]

[1]DeepMind, 6 Pancras Square, London N1C 4AG.
[2]University College London, Gower Street, London WC1E 6BT.
*These authors contributed equally to this work.

背景知识

• 强化学习

■ 在强化学习算法中，通过智能体与环境交互的过程中收集经验数据来更新策略，从而 **强化** 带来更大更多回报的行为。



背景知识

• 强化学习分类

- 人类学习一项新技能，往往是通过不断试错与计划的交织，那么智能体也应该如此。

强化学习

无模型的强化学习

- 代表的是完全基于试错进行的探索学习，智能体在学习的过程中是一无所知的。

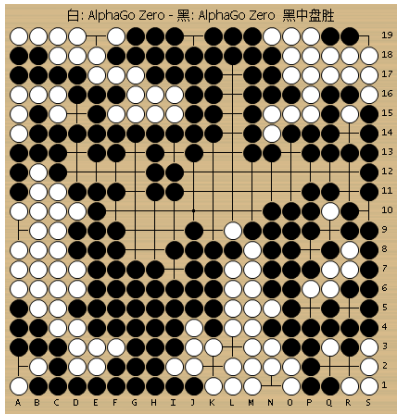
基于模型强化学习

- 已知模型**或环境交互**学习到模型，然后基于模型对未来结果**预测和计划**。
- 更加符合人类在与环境交互学习的过程。

背景知识

• 存在的问题

- 基于模型的强化学习在**视觉信息作为状态输入**的问题中表现不佳，如Atari游戏。
- 无模型的强化学习在**需要精确且复杂的前瞻（lookahead）**的问题中表现不佳，如以围棋为代表的棋类游戏中。



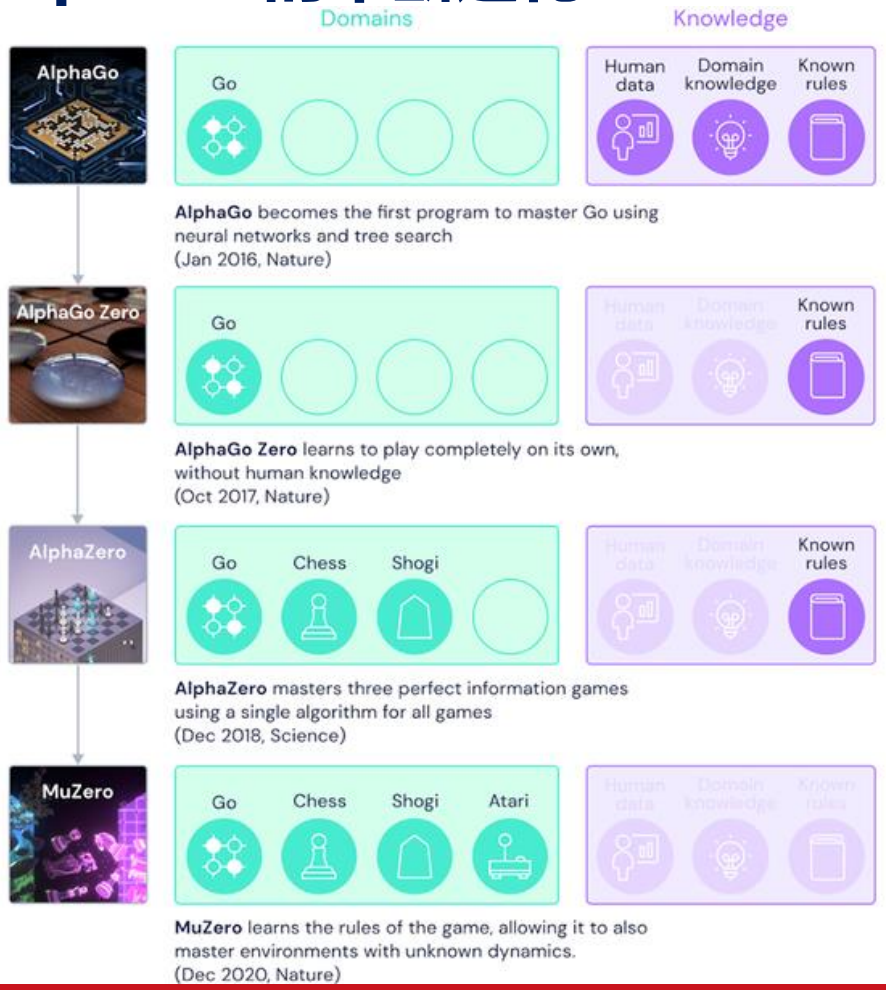
背景知识

- **本文研究目的**
  - 1、如何在不知道状态转移规则的情况下使用基于模型的前瞻搜索（lookahead search）的规划算法（蒙特卡洛树搜索）。
  - 2、设计一个Model-based的算法在视觉信息丰富的环境(如Atari游戏)上表现优于Model-Free算法。

MuZero，它通过**将基于树的搜索（tree-based search）与学习模型（learned model）相结合**，可以在不知道环境基本动态的情况下表现的很好。

背景知识

- **AlphaGo的不断进化**

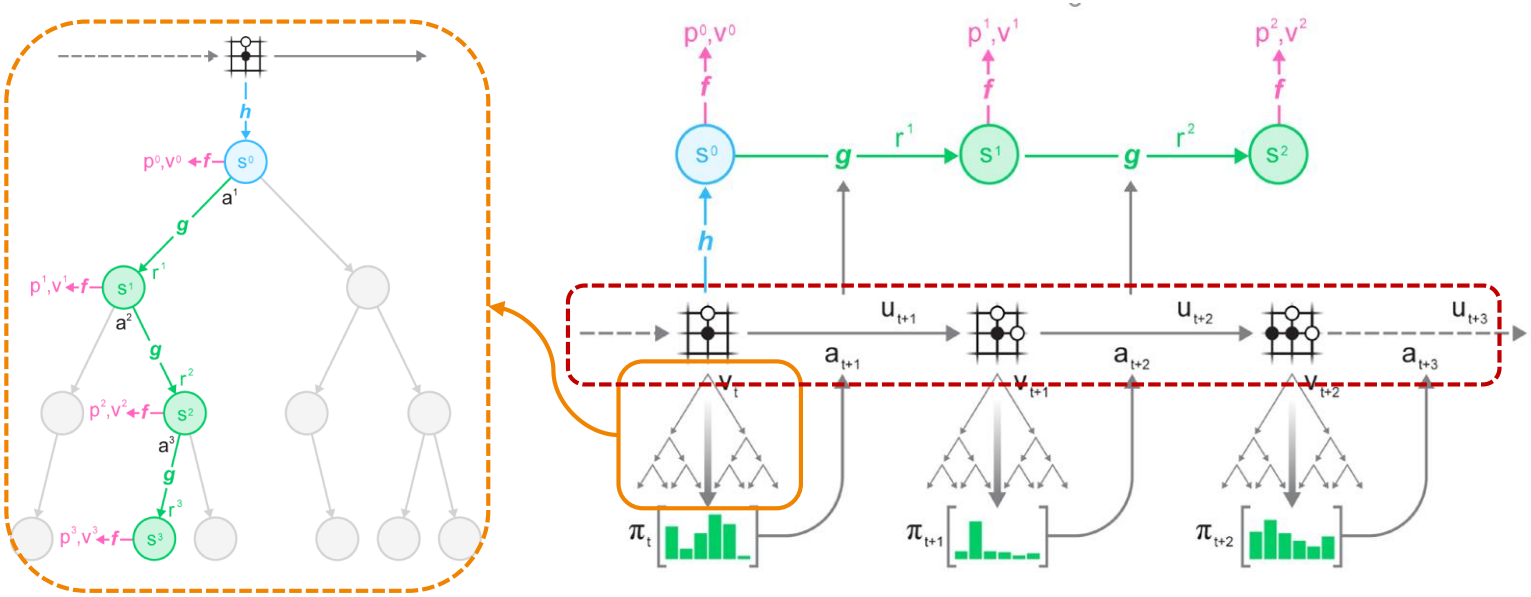


- **AlphaGo：使用神经网络和树搜索解决围棋游戏。**
- **AlphaGo Zero：仅已知围棋规则，通过自我对弈更新策略，效果超出AlphaGo。**
- **AlphaZero：将AlphaGo Zero的算法类推到更多的棋类游戏中。**
- **MuZero：解决在不知道“模型”的情况下使用AlphaZero算法需求。**

https://www.bilibili.com/video/BV147411i7tM

算法细节

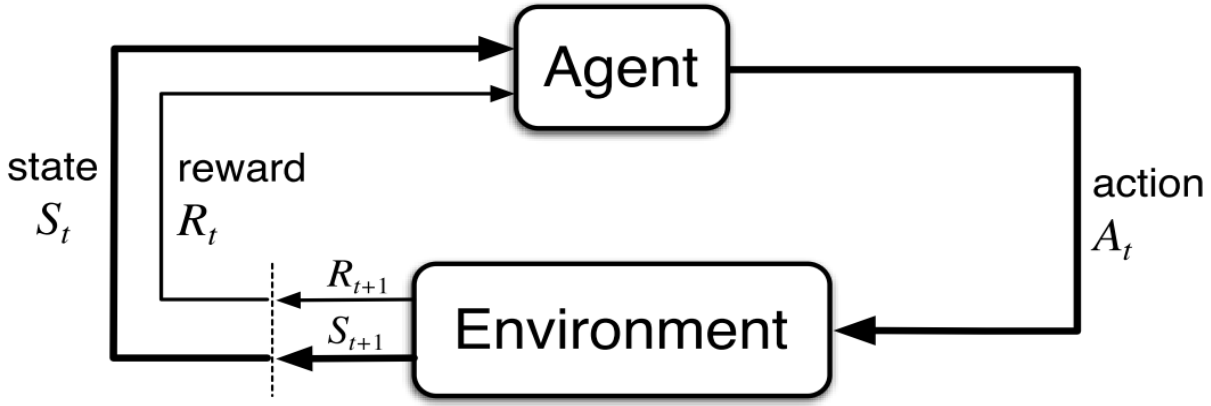
• 算法整体思路



- 利用与真实环境交互获得的数据训练，建立虚拟环境的“模型”。
- 与真实环境进行交互所选择的动作根据MCTS算法决定，在MCTS中搜索树的建立不直接与环境交互，而是使用的是虚拟环境（也就是建立的“模型”）。
- 更新模型与更新策略不断交替，直到找到最优策略。

算法细节

- **马尔科夫决策过程**



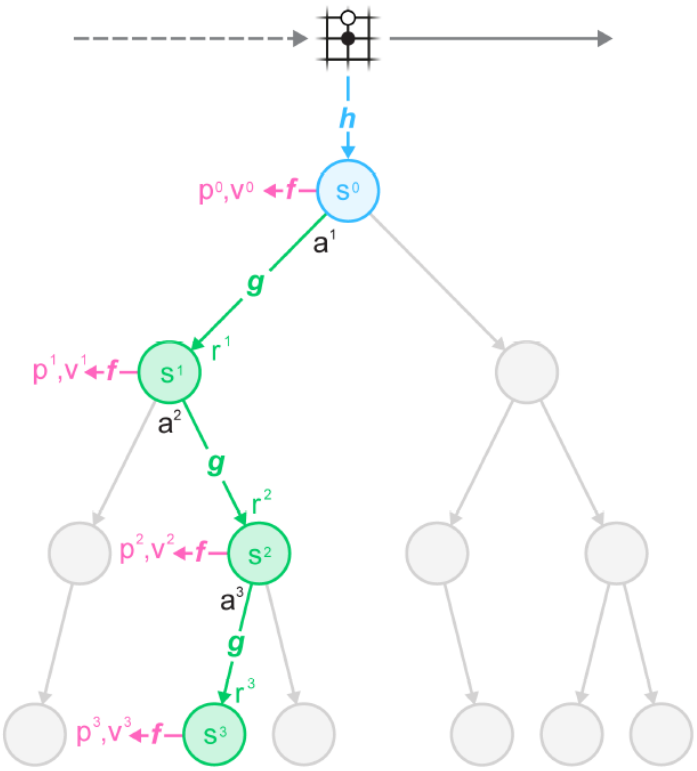
- 马尔科夫决策过程(Markov Decision Process, MDP）是对强化学习研究的过程进行建模的工具。
- 由以下五个元素组成：**状态** S_t、**动作** A_t、**奖励函数** $R(s,a) = R(S_t = s, A_t = a)$、**折扣因子** γ、**状态转移概率** $P(s'|s,a) = (S_{t+1} = s' \mid S_t = s,\ A_t = a)$。
- **状态值函数** $V^{\pi}(s)$：执行策略 π，在状态 s 下累计回报的期望。

算法细节

- **模型学习学什么?**
  - MuZero算法的**主要思想**，是**构造一个抽象的MDP模型，在这个MDP模型上，去预测与规划直接相关的未来数据（策略、价值函数以及奖励），并在此基础上预测数据进行规划。**
    - **“抽象”：**将真实环境中获取的状态，通过一个表征函数（representation function）转换成一个没有直接约束的抽象的状态空间（abstract state space）中的一个隐藏状态，在这个抽象的状态空间中，去学习模型。
    - **MDP模型：**自主学习状态值函数$\mathrm{V}(s)$，奖励函数$R(s,a)$及状态转移$P(s'|s,a)$。

算法细节

- **MuZero中模型的组成**

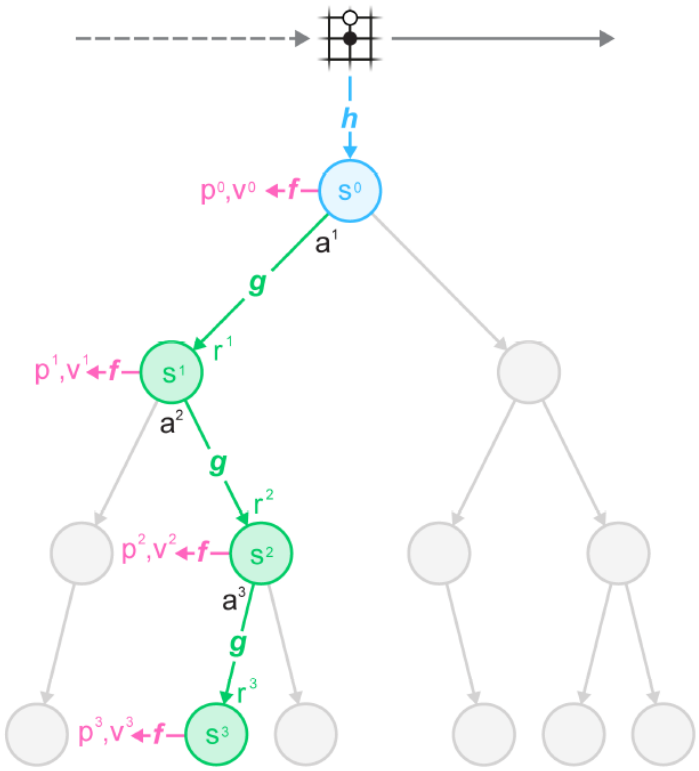


- 本文模型部分符号系统
  - p: 策略， v: 值函数， s: 状态，
    a: 动作， r: 回报
  - h: **表征函数** (representation function)
  - f: **预测函数** (prediction function)
  - g: **动态函数** (dynamics function)

- 把过往的历史观测数据（围棋棋盘或者是Atari的游戏截图）作为输入到**表征函数** h可得到初始隐状态s^0。给定的前一时刻的隐状态（hidden state）s^{k-1}和动作a^k，**动态函数**产生即时回报r^k和下一步隐状态s^k。策略p^k和价值函数v^k是利用**预测函数**f以隐状态s^k为输入计算得到。

算法细节

- **MuZero中模型的组成**



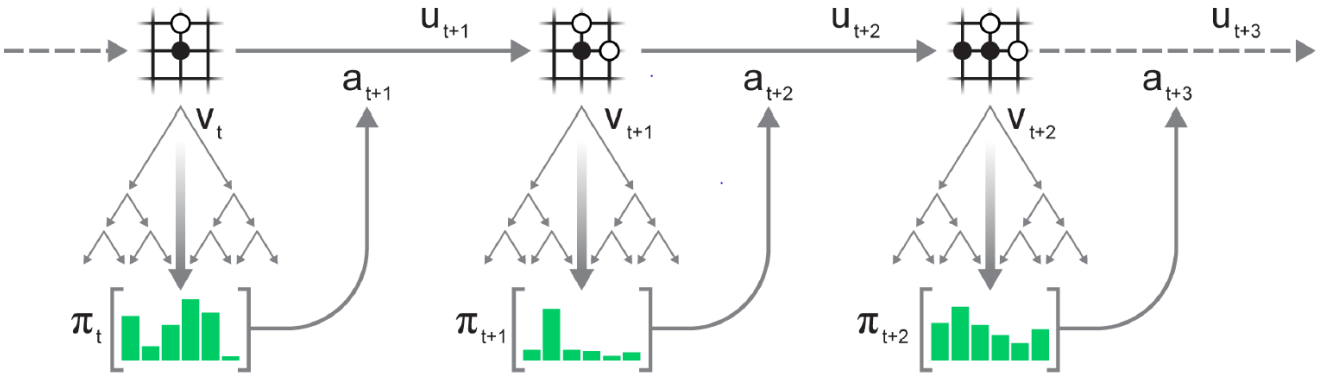
- **Representation：表征编码器h**：$s^0 = h(o^1, \ldots, o^t)$，从历史观测，转换为初始状态。在左边的树型模型中，将连续的 t 帧观测$\{o^1, \ldots, o^t\}$传给表征函数 h 中获得初始隐藏状态s^0。

- **Dynamics：动态生成器g**：$r^k, s^k = g(s^{k-1}, a^k)$，表示系统中的动态变化。给定前一个隐藏状态（previous hidden state）s^{k-1} 和一个候选操作 a^k，系统动态生成函数g就会产生一个即时奖励（immediate reward）r^k 和一个新的隐藏状态 s^k。

- **Prediction：预测器f**：$p^k, v^k = f(s^k)$。策略p^k以及价值函数v^k是通过预测函数 f 从隐藏状态 s^k 中计算出来的。

算法细节

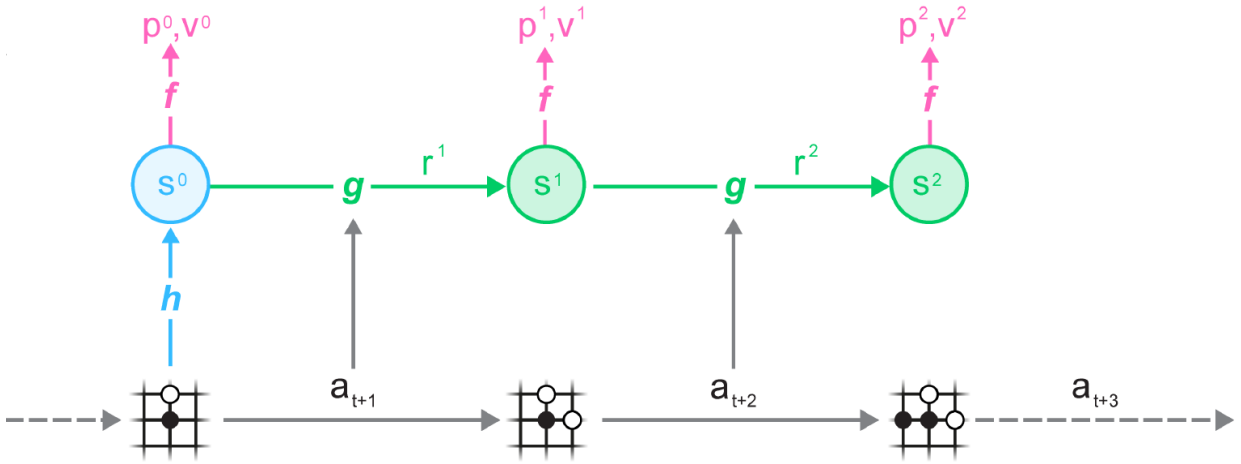
- **MuZero如何与环境进行交互并决策**



- 1、对于左图中所描述的态势而言，**使用蒙特卡洛树搜索对其进行建模**，得到一个**策略网络π_t**，并针对策略网络进行采样，**选出可执行动作a_{t+1}。**
- 2、在执行动作之后a_{t+1}，得到奖励u_{t+1}，得到下一时刻的观测（中间的图），同样的使用MCTS进行建模，得到策略网络π_{t+1}，并选出可执行动作a_{t+2}。
- 3、环境接受了行动，产生了一个新的观察o_{t+2}和回报u_{t+2}，得到右图。
- 4、就这样不断往下推演，**在episode结束时，轨迹数据被存储到回放缓冲区中。**

算法细节

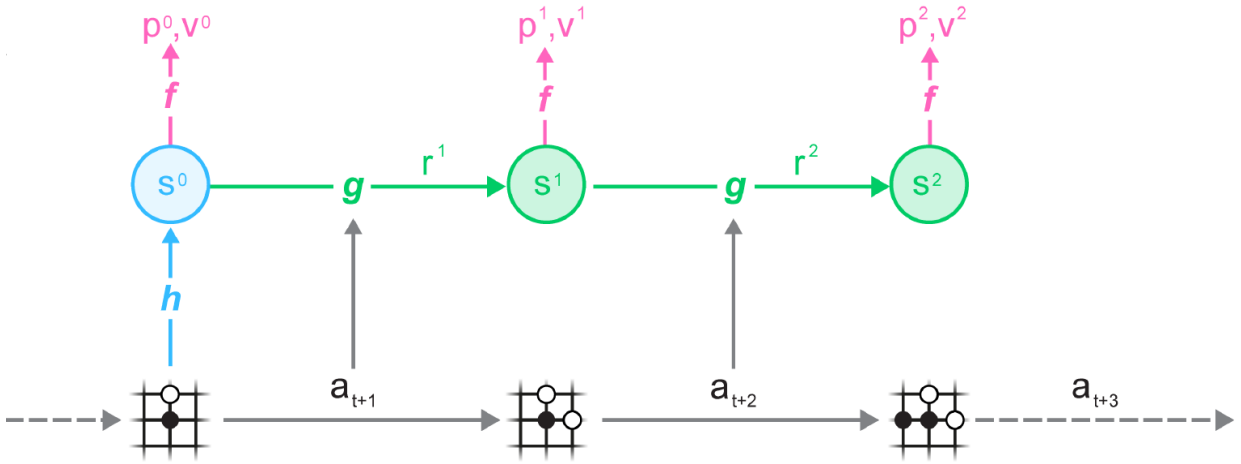
- **MuZero如何训练模型**



- **对回放缓存区中的轨迹数据进行采样，选取一个序列，然后根据该轨迹运行MuZero模型。**
- 在初始步中，**编码器** h 接受来自所选轨迹的过去观测值$\{o^1, \ldots, o^t\}$。
- **随后，模型展开K步循环。**
- 在第k个步骤中，系统**动态生成函数** g 接收上一步的隐状态 s_{t-1} 和真实的动作 a_{t+1}。

算法细节

- **MuZero如何训练模型**



- **表征编码器h**、**动态生成器g** 以及**预测器 f** 的参数，进行**端到端的联合训练**，被用来预测下面三个量：
  - **策略网络：** $p^k \approx \pi_{t+k}$
  - **价值网络：** $v^k \approx z_{t+k}$ ，其中， z_{t+k}是采样回报（sample return），如棋类游戏的最终胜负或Atari中的n步回报。
  - **即时奖励：** $r^k \approx u_{t+k}$

算法细节

• 训练目标

- **表征编码器h**、**动态生成器g**和**预测器f**的参数，进行**端到端的联合训练**，输出分别对应于真实环境中的策略 π、采样回报/状态价值 z 以及回报 u。
- **联合训练总体loss**，在末尾加入L2正则项：

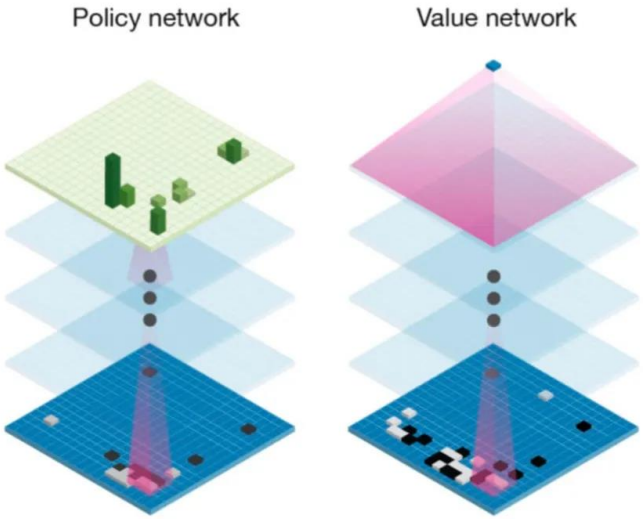
$$l_t(\theta) = \sum_{k=0}^{K} l^r(u_{t+k}, r_t^k) + l^v(z_{t+k}, v_t^k) + l^p(\pi_{t+k}, \mathbf{p}_t^k) + c||\theta||^2$$

- 第一个目标是最小化预测策略 p_t^k 和MCTS得到的搜索策略 π_{t+k} 之间的误差。
- 第二个目标是最小化预测价值 v_t^k 和MCTS得到采样回报 z_{t+k} 的之间的误差。
- 第三个目标是最小化预测奖励 r_t^k和观察到的奖励 u_{t+k} 之间的误差。

算法细节

• 策略网络和价值网络

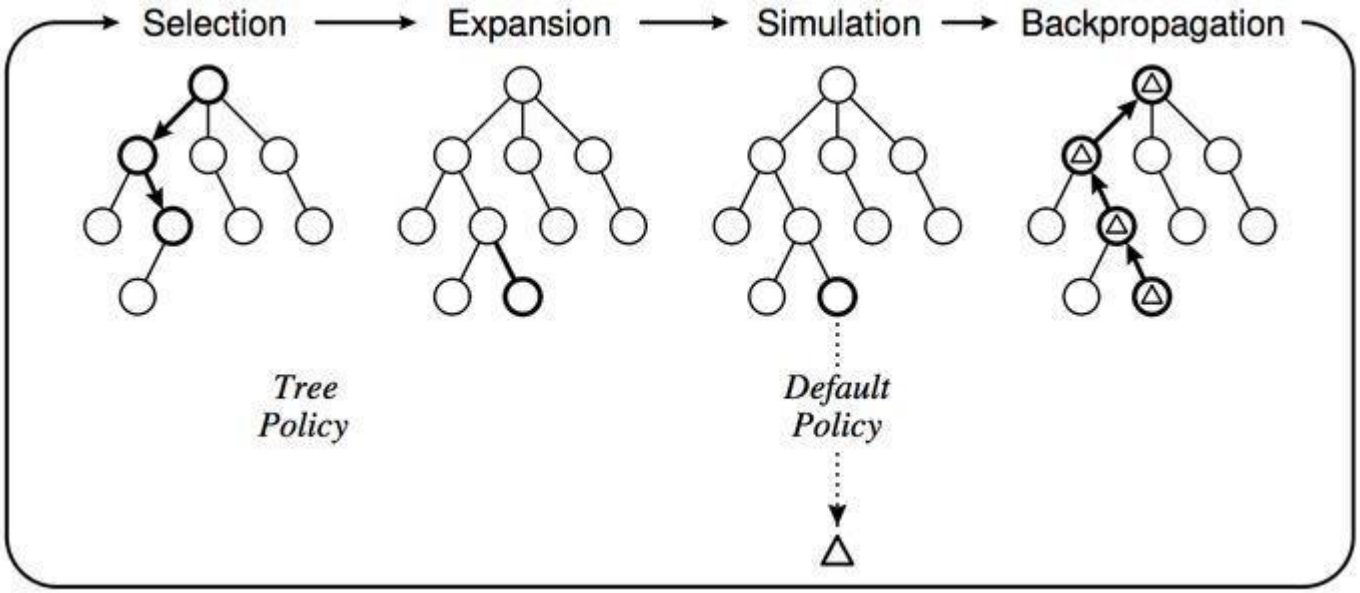
- **如何使用神经网络？** MuZero算法使用了AlphaZero中提到的策略网络和值网络。



- **策略** $p(s,a)$ 表示在状态 s 时所有可能的动作 a 分布，据此可以估计最优的动作。比如，在下棋的时候，玩家会想我当前可能会如何走，对手如何走，我怎么走是最优的。
- **价值**$v(s)$是对当前状态 s 下获胜的可能性的评估，即通过对所有的未来可能性进行加权平均，确定当前局势的获胜概率。
- 根据**策略网络，能够预测每一步的动作**；依赖**价值网络，可以选择价值最高的动作**。将这两个估计结合起来可以得到更好的结果。

算法细节

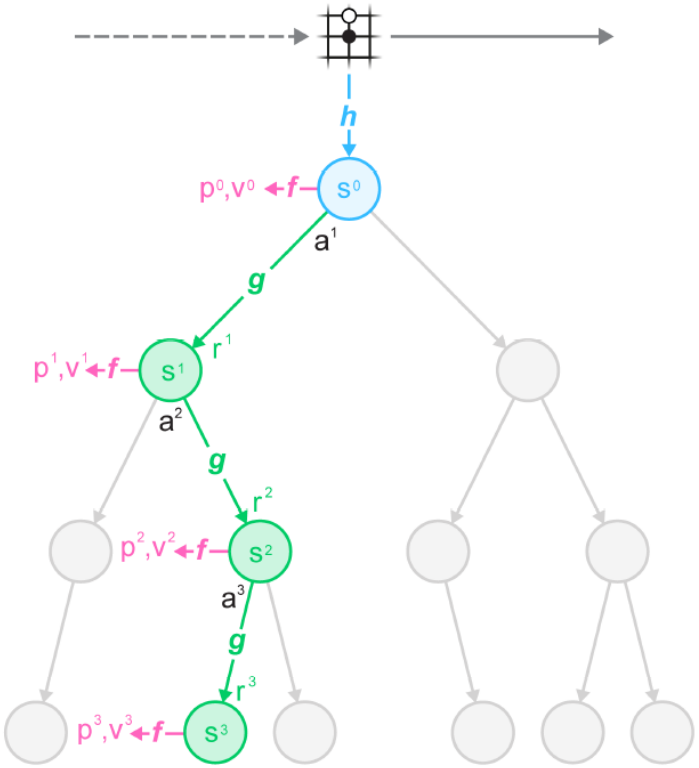
- **蒙特卡洛树搜索**



- **蒙特卡洛树搜索（Monte Carlo Tree Search，MCTS）**是一个**迭代的**，**最佳优先**（best-first）**树搜索**过程。其目标是**帮我们计算出到底应该采用什么样的动作，可以实现长期受益最大化**。
- 其过程主要分为**选择（Selection）、扩展（Expansion）、模拟（Simulation）和回溯（Backpropagation）**四个主要阶段。

算法细节

- **MuZero算法中MCTS-模拟**



- 搜索树中的每一个节点（node）代表着一个状态（MuZero算法中是隐状态）s，用边（edge）(s,a) 表示在状态 s 做出的动作 a，边中存储的信息包括 $\{N(s,a), Q(s,a), P(s,a), R(s,a), S(s,a)\}$，分别代表访问次数$N$，平均状态行为值$Q$，策略$P$，回报$R$及状态转移$S$。

- **模拟**：使用构建的**动态生成器**$g: r^k, s^k = g(s^{k-1}, a^k)$**进行共 l 步的模拟。**

算法细节

- **MuZero算法中MCTS-选择**
  - **选择：基于上确界（Upper Confidence Bound，UCB）策略的动作选择策略。**

$$a^k = \arg\max_a [Q(s,a) + P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)} (c_1 + \frac{\sum_b N(s,b) + c_2 + 1}{c2})]$$

强化学习中基于值函数的动作选择基本原则：选择使得动作行为值函数最大的动作。

对于状态 s 下的不同动作 a_i，这一项为常数。

访问次数越少的动作-行为对，尽可能多的访问。

  - **Trade-off between exploitation（利用） and exploration（探索）。**

** 访问次数N， 平均状态行为值Q， 策略P， 回报R及状态转移S。*

算法细节

- **MuZero算法中MCTS-拓展&回溯**

  - **拓展：在选择了一个动作 a 之后，在搜索树中生成一个新的节点，**对应上一个状态在执行完动作 a 之后的局面。

  - **回溯：**在模拟结束后，**从子节点开始，沿着刚刚向下的路径往回走，沿途更新各个父节点的统计信息。**每个节点都在其下保存所有值的连续均值估计，这使得UCB公式可以随着时间的推移做出越来越准确的决策，从而确保MCTS收敛到最优动作。

算法细节

- **MuZero算法中MCTS-回溯**

  - **回溯**中如何更新节点的信息?
    - 回报：即平均累计奖励，通过n步自举（n-step bootstraping）方法获取。

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

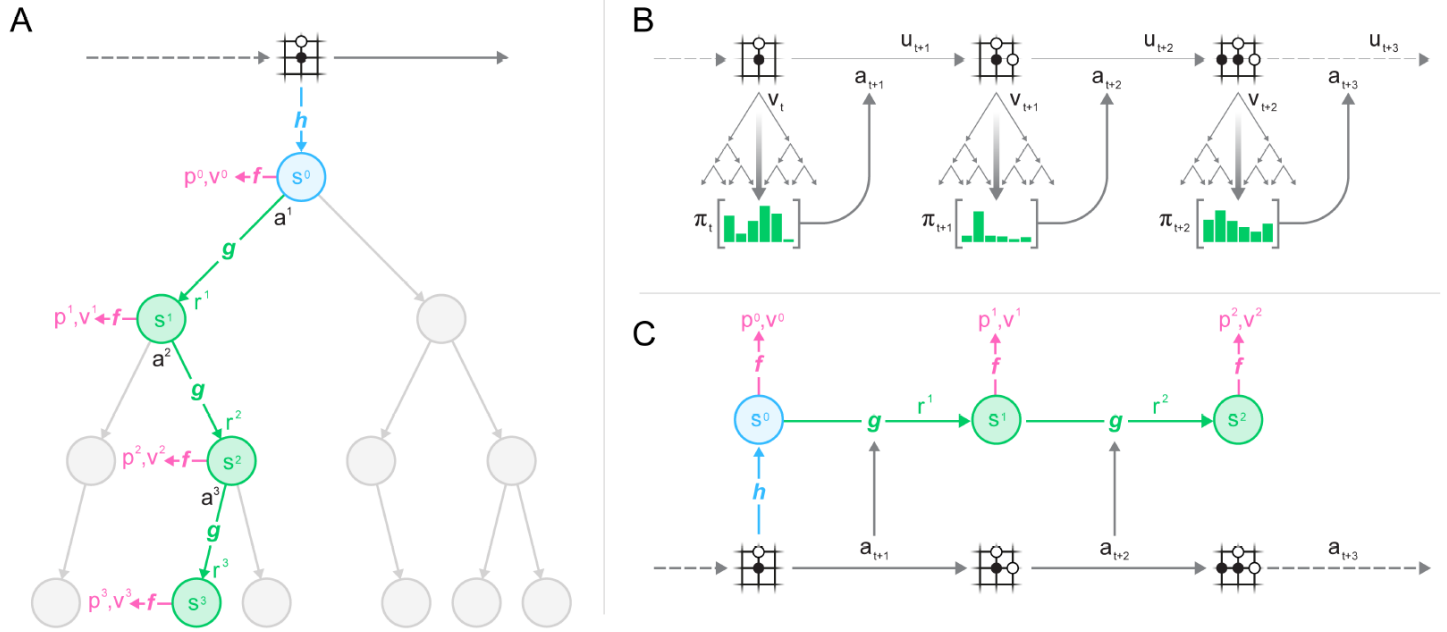
    - 状态行为值函数：

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

$$N(s^{k-1}, a^k) := N(s^{k-1}, a^k) + 1$$

算法细节

- 总结

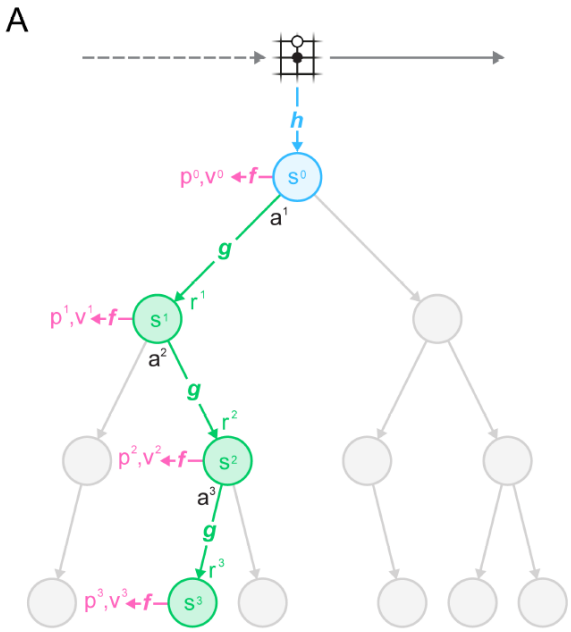


- A：MuZero中模型的组成　B：MuZero如何与环境交互　C：MuZero如何训练模型

https://www.bilibili.com/video/BV1Av411t7oU

算法细节

- 总结

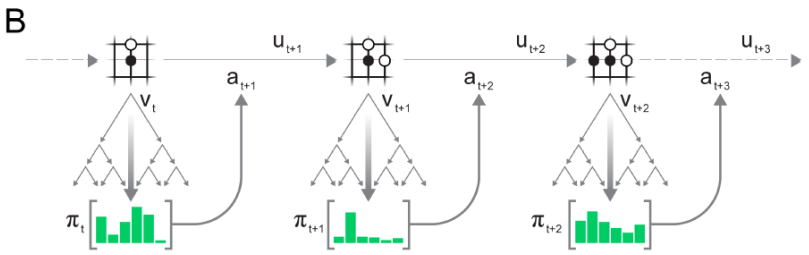


- **模型的三部分：**
- Representation：表征编码器h: $s^0 = h(o^1, \ldots, o^t)$
- Dynamics：动态生成器g: $r^k, s^k = g(s^{k-1}, a^k)$
- Prediction：预测器f: $p^k, v^k = f(s^k)$

算法细节

• 总结

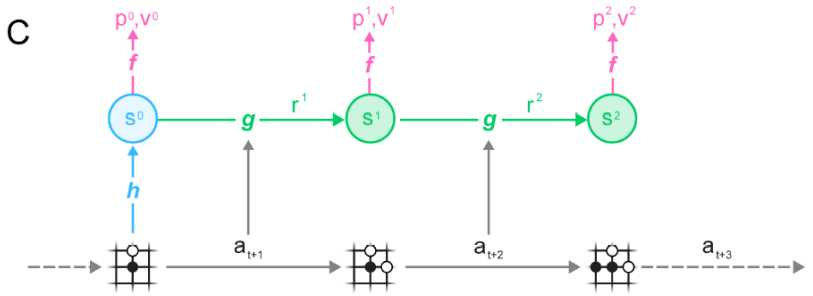
- 利用**MCTS算法**去选择当前状态下的动作。
- MCTS中树的拓展根据MuZero算法中学到的**动态生成器g**来完成。
- 将**轨迹放入经验缓存区**供模型训练使用。



算法细节

• 总结

- **表征编码器h**、**动态生成器g**和**预测器f**的参数，进行**端到端的联合训练**，输出分别对应于真实环境中的策略 π、采样回报/状态价值 z 以及回报 u。



- **联合训练总体loss**:

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

谢 谢!