多智能体学习

第三讲：多智能体强化学习应用

教师：张启超

中国科学院大学
中国科学院自动化研究所

Spring, 2023

棋类游戏应用

- 完全信息零和博弈

2016 Nature，DeepMind的AlphaGo以4:1的大比分战胜了世界围棋顶级选手李世石

2013-2015 → 2016-2017 → 2017-2018 → 2019-2020

- 蒙特卡洛树搜索
- Actor-Critic
- 自我博弈

2017 Nature，DeepMind的AlphaGo Zero不依赖人类的棋谱数据，自我博弈学习达到人类顶级水平

MOBA类游戏应用

- 非完全信息多人混合博弈

2019 Science, 谷歌第一视角多个体合作游戏，雷神之锤

2019 Science, CMU的六人桌德扑Bot Pluribus

2019.5, OpenAI Five，Dota2的人机大战

2013-2015 → **2016-2017** → **2017-2018** → **2019-2020**

- 多智能体深度强化学习
- 递归神经网络

…

2019.8,微软麻将AI Suphx, 10段

2019.11 Nature，谷歌DeepMind Alpha Star，星际争霸II达到大师级水平

2019.12，腾讯绝悟AI击败王者荣耀顶尖职业玩家

实际应用

智能仓储物流配送，
亚马逊/菜鸟等

无人送餐小车
调度配送等

2013-2015 → 2016-2017 → 2017-2018 → 2019-2020 → 2021-2023

滴滴平台的派单与配送，
KDD CUP 2020 Competition

LLM;
LLM+Robotics

https://outreach.didichuxing.com/competition/kddcup2020/

实际应用

无人送餐小车调度配送

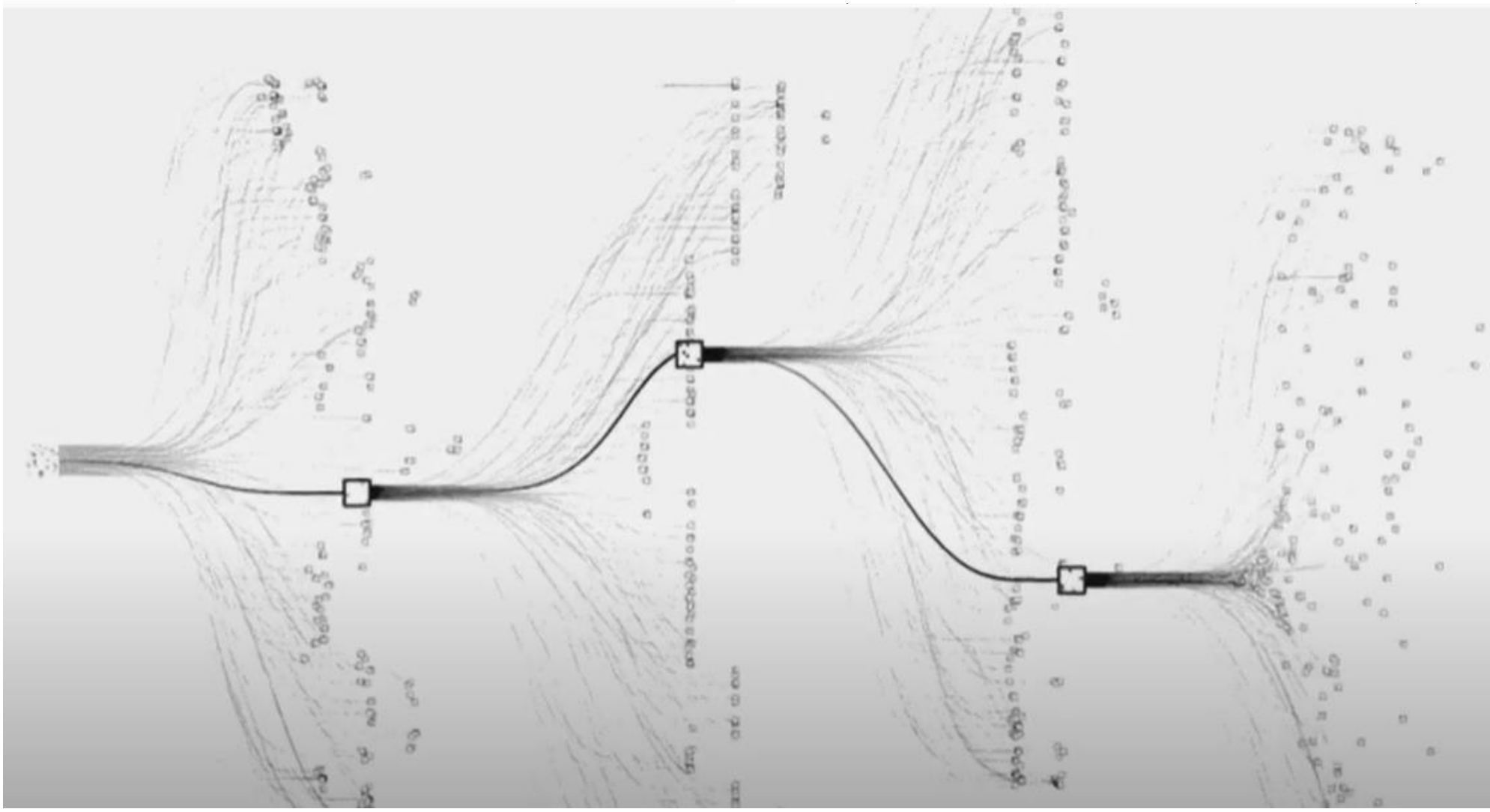
实际应用

2021,百度发布Apollo RL，真实环境下的自动驾驶

课程目录

- 围棋系列
  - AlphaGo
  - AlphaGoZero
- 实时策略游戏系列
  - AlphaStar
- 总结

AlphaGo



AlphaGo

状态：黑子、白子和空位置的排列

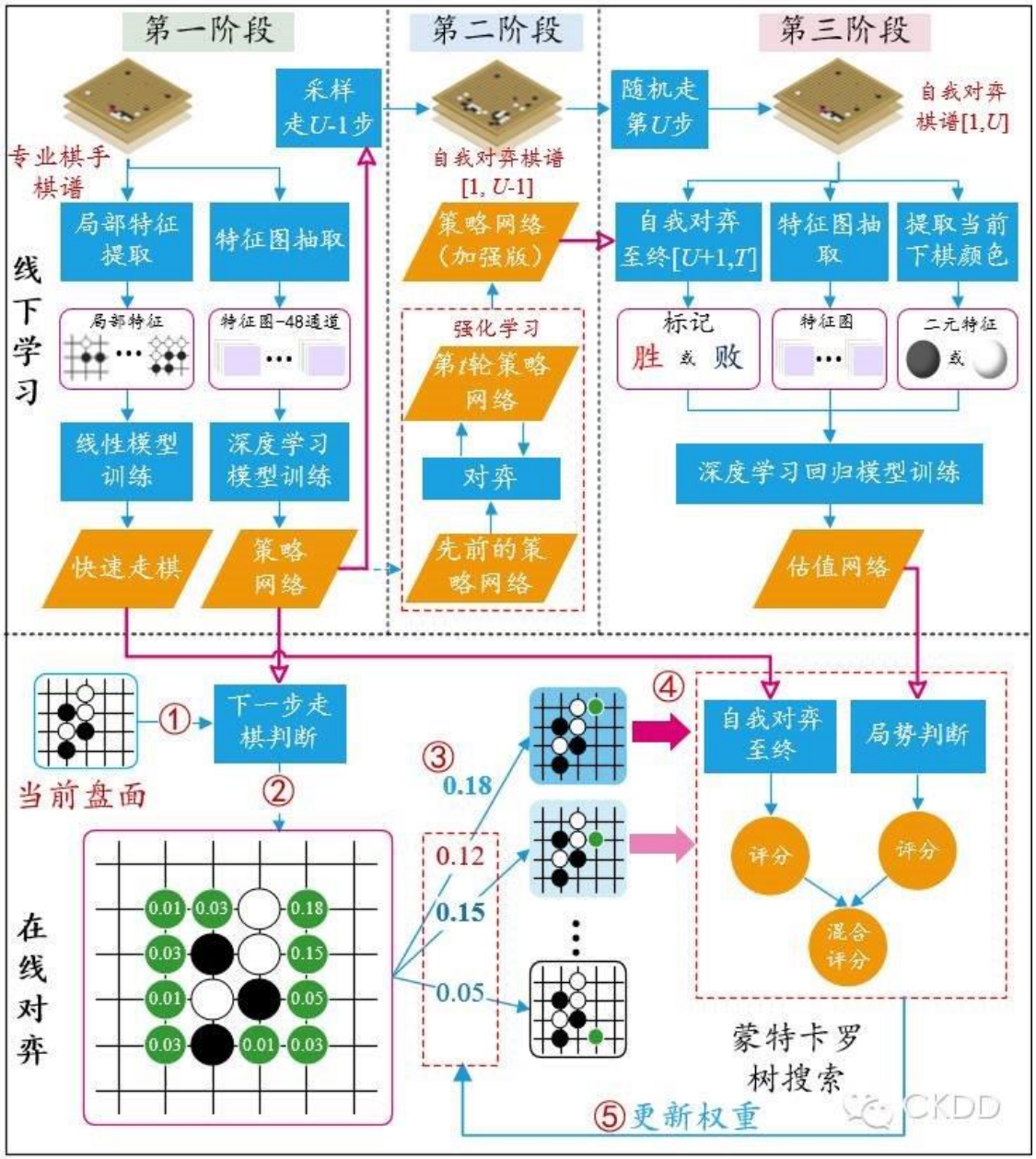
- AlphaGo实际用了19*19*49
- AlphaGo Zero由0或1组成的19*19*17的张量(黑与白)

动作：在空位置放棋子

- 动作空间：{1,2,…,361}

1. Silver and others: Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.
2. Silver and others: Mastering the game of Go without human knowledge. *Nature*, 2017.

AlphaGo



http://www.kddchina.org/#/Content/alphago

线下学习阶段

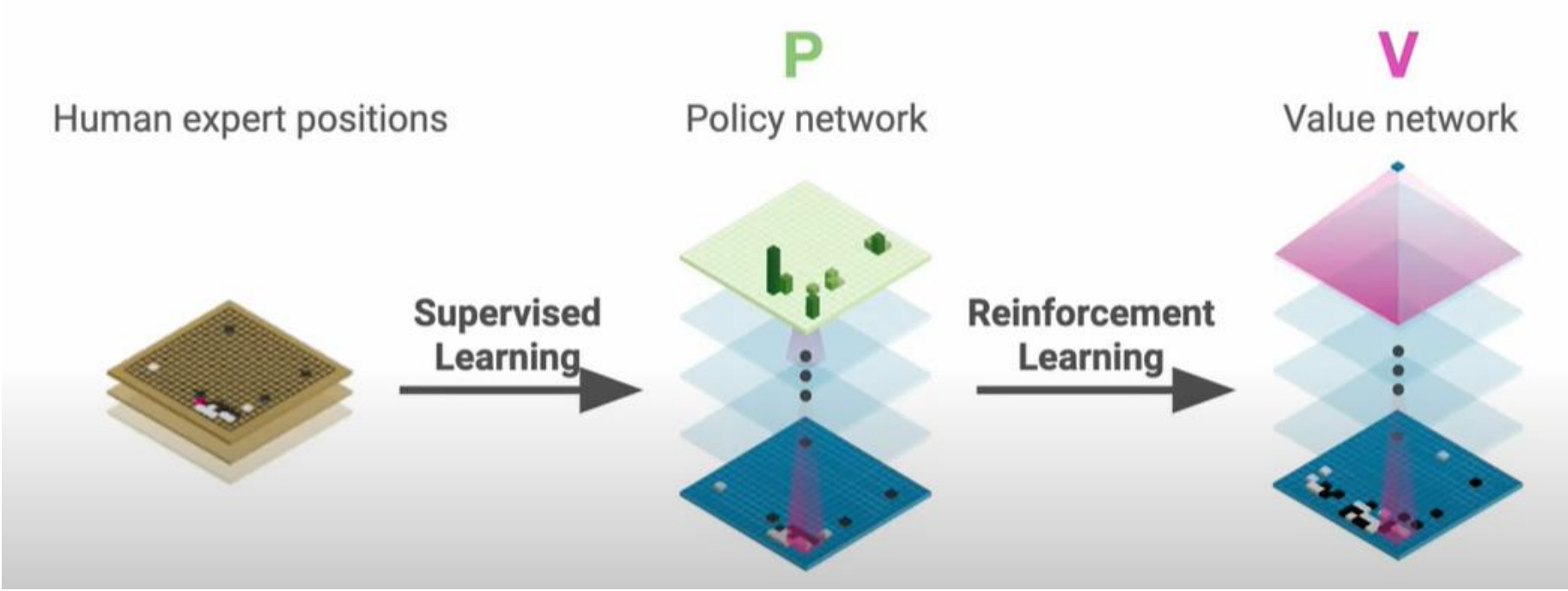
1. 利用模仿学习来初始化策略网络
   （利用人类玩家数据来监督训练）
2. 利用策略梯度来训练策略网络
   （黑白方使用两个策略网络自我博弈）
3. 训练完策略网络后，再训练值估计网络

在线对弈阶段

利用策略网络和值网络执行蒙特卡洛树搜索

AlphaGo

线下学习阶段技术细节

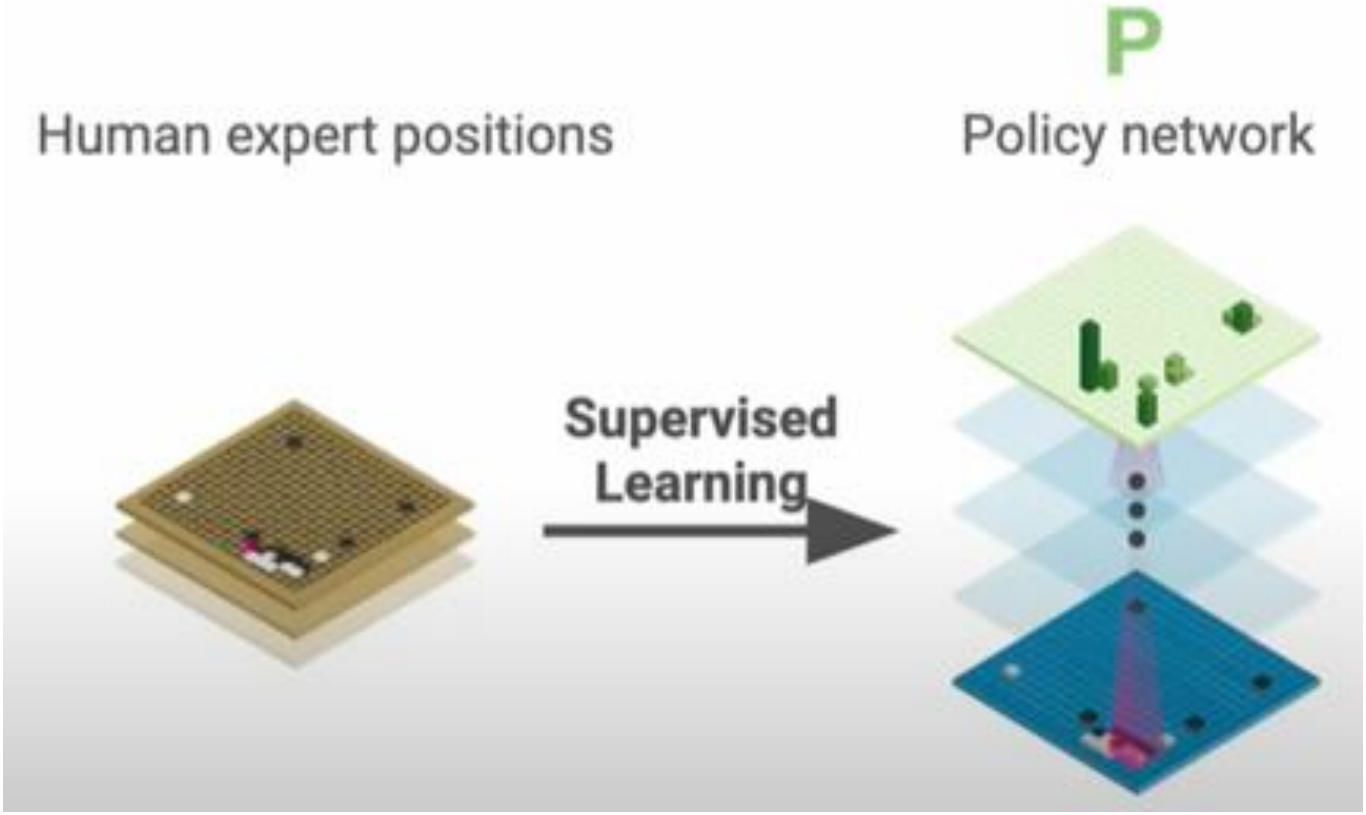


训练过程

AlphaGo

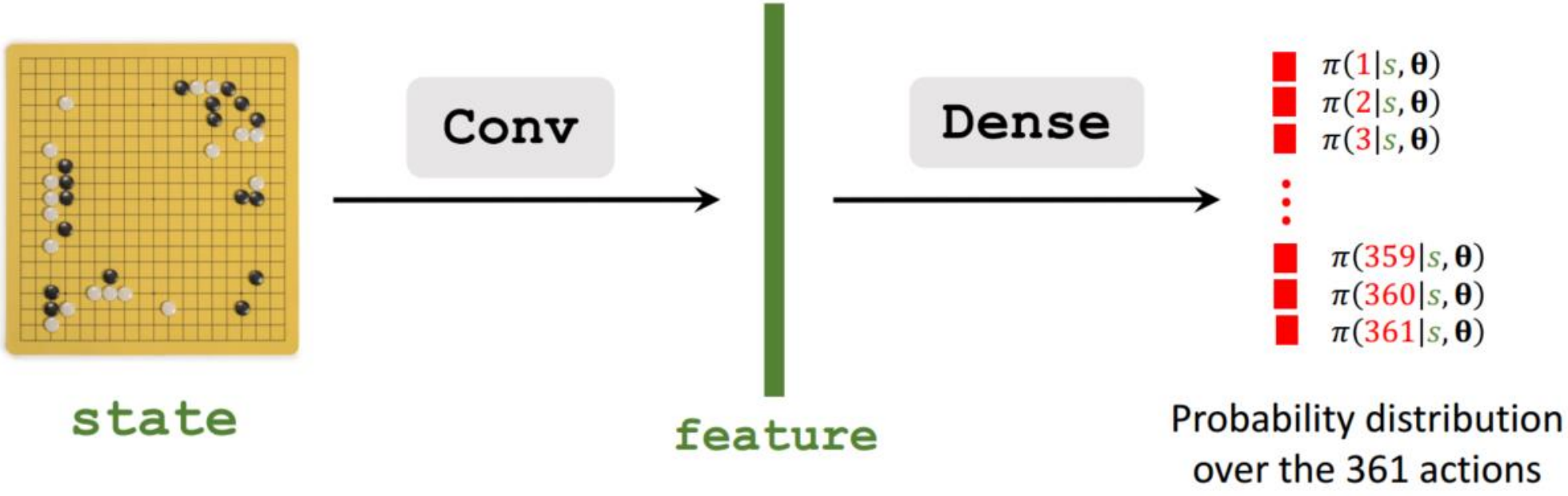
策略网络的初始化：模仿学习

策略网络的训练：自我博弈+强化学习



AlphaGo

■ 第一阶段：策略网络的初始化-模仿学习



状态输入

卷积提取特征

全连接后softmax获得每个动作的概率

AlphaGo

AlphaGo策略网络的输入状态: 19*19*49

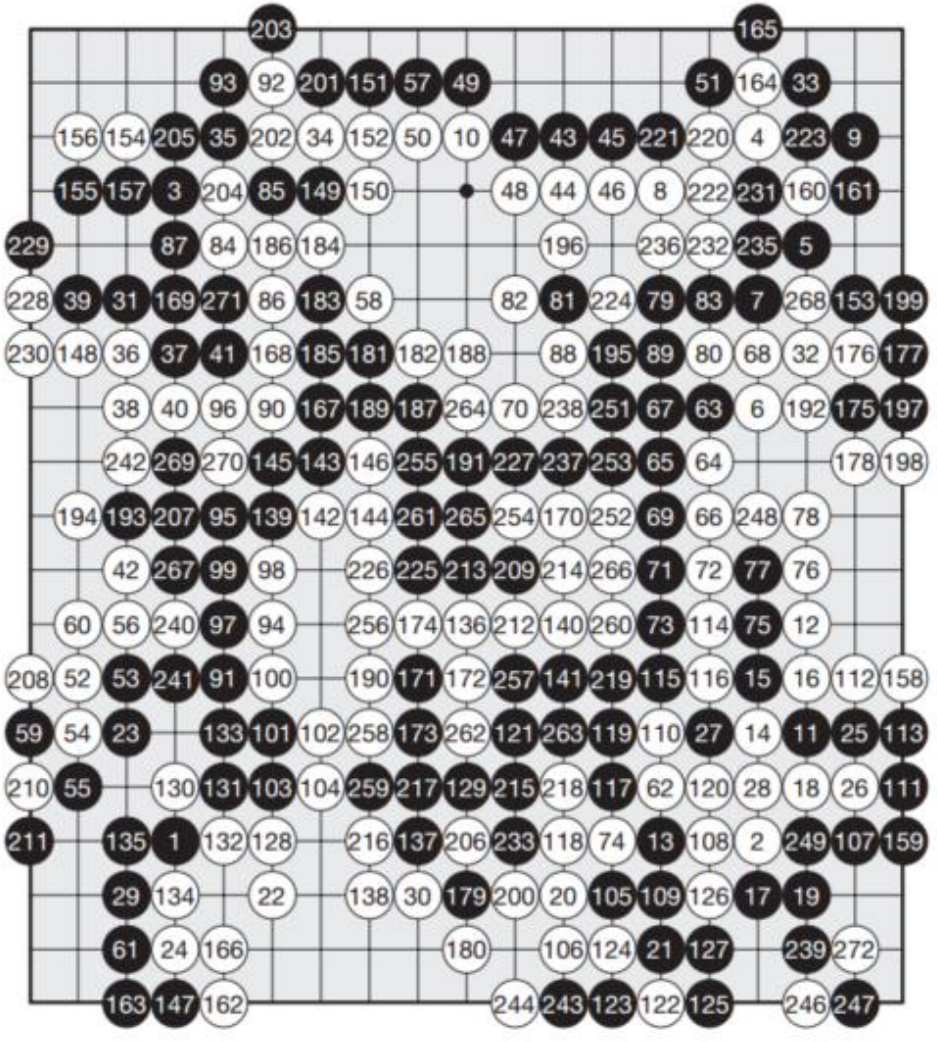
特征名称	平面数量	说明
执子颜色	3	3 个特征平面分别代表当前执子方、对手方，以及棋盘上的空点的棋子颜色
一	1	一个全部填入值 1 的特征平面
零	1	一个全部填入值 0 的特征平面
明智度	1	一个动作如果合法，且不会填补当前棋手的眼，则会在平面上填入 1，否则填入 0
动作回合数	8	这个集合有 8 个二元平面，代表一个动作落子离现在有多少个回合
气数	8	当前动作所在的棋链的气数，也分为 8 个二元平面
动作后气数	8	如果这个动作执行了之后，还会剩多少口气
吃子数	8	这个动作会吃掉多少颗对方棋子
自劫争数	8	如果这个动作执行之后，有多少己方的棋子会陷入劫争，可能在下一回合被对方提走
征子提子	1	这颗棋子是否会被通过征子吃掉
引征	1	这颗棋子是否能够逃出一个可能的征子局面
当前执子方	1	如果当前执子方是黑子，整个平面填入 1；如果是白子，则填入 0

AlphaGo

■ 第一阶段：策略网络的初始化-模仿学习

随机初始化方式

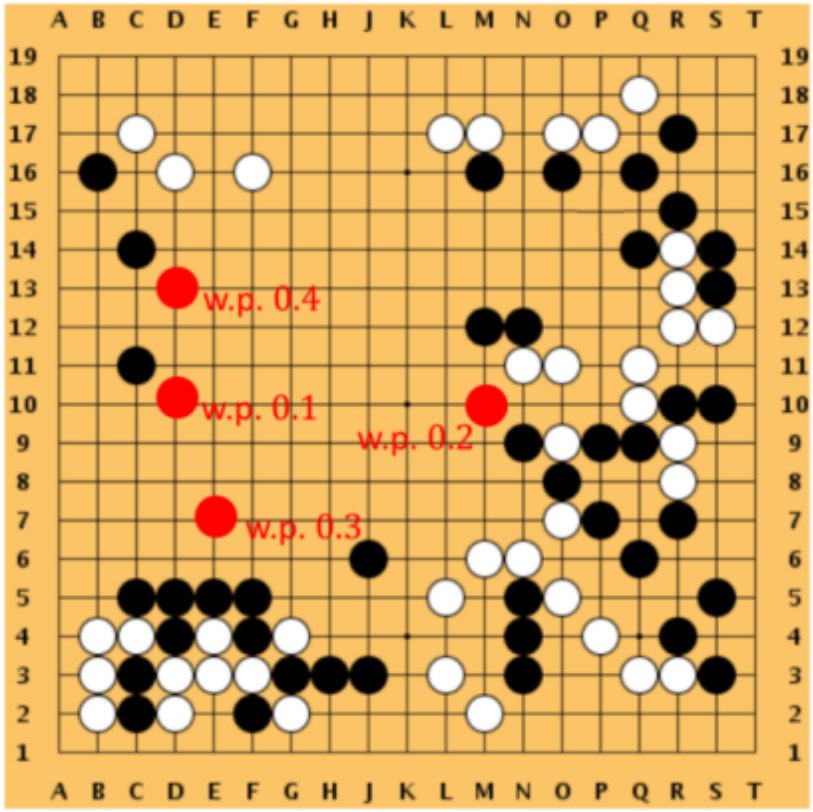
- 两个策略进行自我博弈，各自采取随机动作
- 由于围棋的复杂性，且奖赏稀疏，导致很长时间无法学习到合理策略
- KGS数据集包含160K局围棋数据（6段+）



AlphaGo

■ 第一阶段：策略网络初始化—模仿学习

利用人类专家样本进行纯监督训练得到的初始策略网络，可达到业余玩家水平



- 输入当前状态19*19*49 tensor
- 策略网络进行预测

$$\mathbf{p}_t = [\pi(1|s_t,\boldsymbol{\theta}),\cdots,\pi(361|s_t,\boldsymbol{\theta})] \in (0,1)^{361}$$

- 标签为仅专家动作为1的one-hot向量 $\mathbf{y}_t \in \{0,1\}^{361}$
- 损失函数= $\text{CrossEntropy}(\mathbf{y}_t, \mathbf{p}_t)$
- 利用梯度下降更新网络参数

AlphaGo

- 第二阶段：策略网络训练：自我博弈+强化学习

  - 训练数据难以覆盖所有状态，当遇到未训练过的状态时，策略网络难以输出好的动作；
  - 相同状态，不同选手的动作不同；
  - 模仿学习MDP问题中存在复合误差的问题；

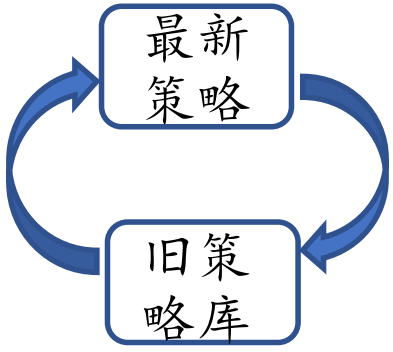
在策略网络初始化之后，利用自我博弈+强化学习技术继续训练策略网络

AlphaGo

■ 第二阶段：策略网络训练：自我博弈+强化学习

自我博弈 (naive SP:直接跟**最新**的自己对打)

- 当前智能体策略网络选用最新的参数
- 对手智能体策略网络从不同版本旧策略中随机选择



Player
(Agent)
policy network with latest param

V.S.

Opponent
(Environment)
policy network with old param

AlphaGo

- 第二阶段：策略网络训练：自我博弈+强化学习

直接跟**最新**的自己对打，随着学习的进行，可能会**陷入循环**

剪刀-石头-布游戏

$$\begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$

利用当前策略与不同版本旧策略进行自我博弈，可一定程度避免陷入循环，防止策略遗忘。

AlphaGo

- 第二阶段：策略网络训练：自我博弈+强化学习

为什么利用自我博弈可以使得策略网络迭代提升？

Transitive game⊕ Non-Transitive game(Cyclic game)

Transitive Game: the rules of winning are transitive across different players.

$$v_t \text{ beats } v_{t-1},\ v_{t+1} \text{ beats } v_t \rightarrow v_{t+1} \text{ beats } v_{t-1}$$

策略具有传导性，强者恒强

具有明确段位划分的游戏，意味着游戏传导性较强

[Czarnecki 2020] Real World Games Look Like Spinning Tops. Wojciech Marian Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, Max Jaderberg

AlphaGo

■ 第二阶段：策略网络训练：自我博弈+强化学习

奖赏信号的设定

假设一局围棋游戏在T轮后结束

奖赏为：

- $r_1 = r_2 = r_3 = \cdots = r_{T-1} = 0$
- $r_T = +1$(获胜方)
- $r_T = -1$(失败方)

回报为：$u_t = \sum_{i=t}^{T} r_i$ (不做折扣)

获胜方

$u_1 = u_2 = \cdots = u_T = +1$

失败方

$u_1 = u_2 = \cdots = u_T = -1$

AlphaGo

■ 第二阶段：策略网络训练：自我博弈+强化学习

策略网络的训练：策略梯度方法

策略梯度的无偏估计 $\frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(s_t, a_t)$

观测的回报： $Q_\pi(s_t, a_t) = \mathbb{E}\left[U_t|s_t, a_t\right]$

近似的策略梯度： $\frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot u_t$

AlphaGo

■ 第二阶段：策略网络训练：自我博弈+强化学习

重复以下步骤进行策略网络的训练

- 两个策略网络对弈一局游戏结束(Player v.s. Opponent)
- 得到复盘轨迹数据 $s_1, a_1, s_2, a_2, \cdots, s_T$
- 游戏结束后，更新玩家(Player)的策略网络

  统计各玩家的回报值 $u_1 = u_2 = \cdots = u_T$

  近似策略梯度 $\mathbf{g}_\theta = \sum_{t=1}^{T} \frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot u_t$

  更新策略网络 $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \cdot \mathbf{g}_\theta$

AlphaGo

■ 第二阶段：策略网络训练：自我博弈+强化学习

经过模仿学习和自我博弈+强化学习两个阶段后，得到训练好的策略网络π (加强版策略网络)

观测当前状态s_t, 采样动作

$$a_t \sim \pi(\cdot|s_t, \boldsymbol{\theta})$$

加强版策略网络开始能战胜初级职业玩家；

但表现不稳定，一个小失误可能导致全盘皆输。

“缺少大局观”对每一步的盘面好坏进行全局判断

AlphaGo

- 第三阶段：值网络的训练

状态值函数 $V_\pi(s) = \mathbb{E}[U_t|S_t = s]$ 其中$U_t = +1$ 或 -1

判断给定策略π下当前局面s的好坏

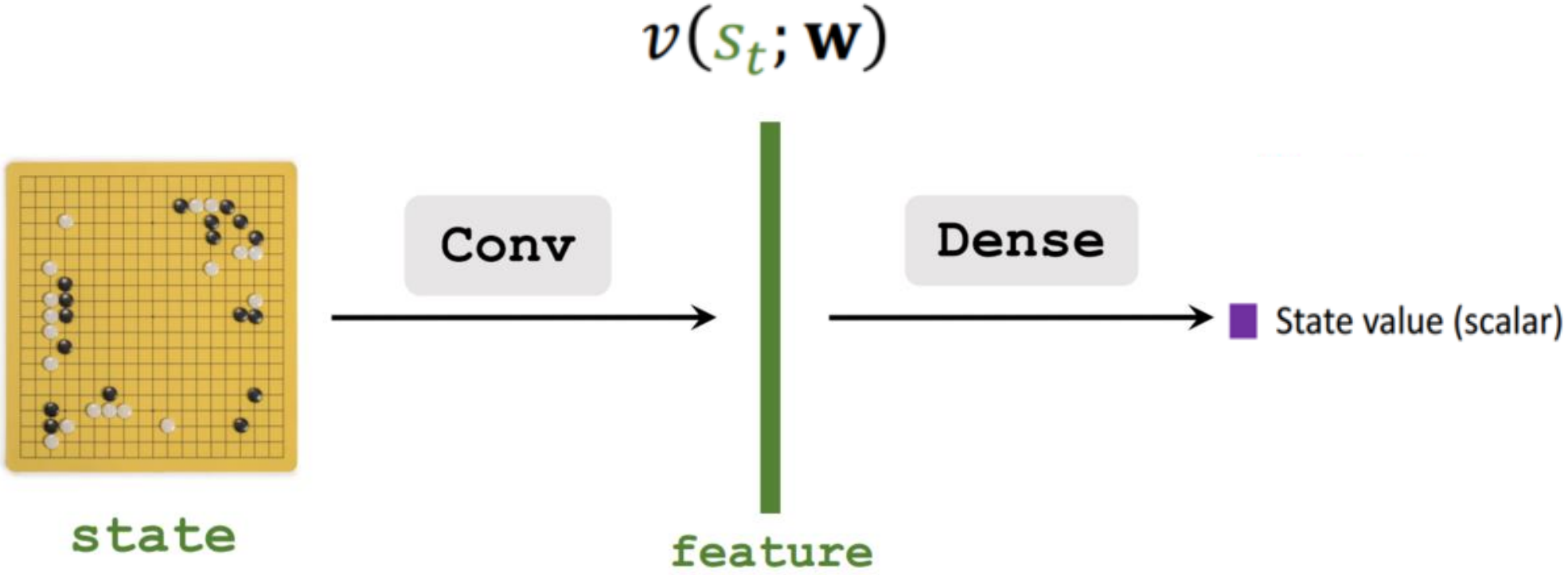
$V_\pi(s)$接近1，说明当前盘面很好，获胜概率较大

$V_\pi(s)$接近-1，说明当前盘面很差，失败概率较大

利用神经网络来逼近$V_\pi(s)$，来评估当前盘面的好坏

AlphaGo

■ 第三阶段：值网络的训练



AlphaGo

■ 第三阶段：值网络的训练

策略网络与值网络是分别训练的，得到加强版策略网络后，再训练值网络

重复以下步骤

1. 开始对弈一局游戏直至结束(加强版策略网络)

   获胜 $u_1 = u_2 = \cdots = u_T = +1$

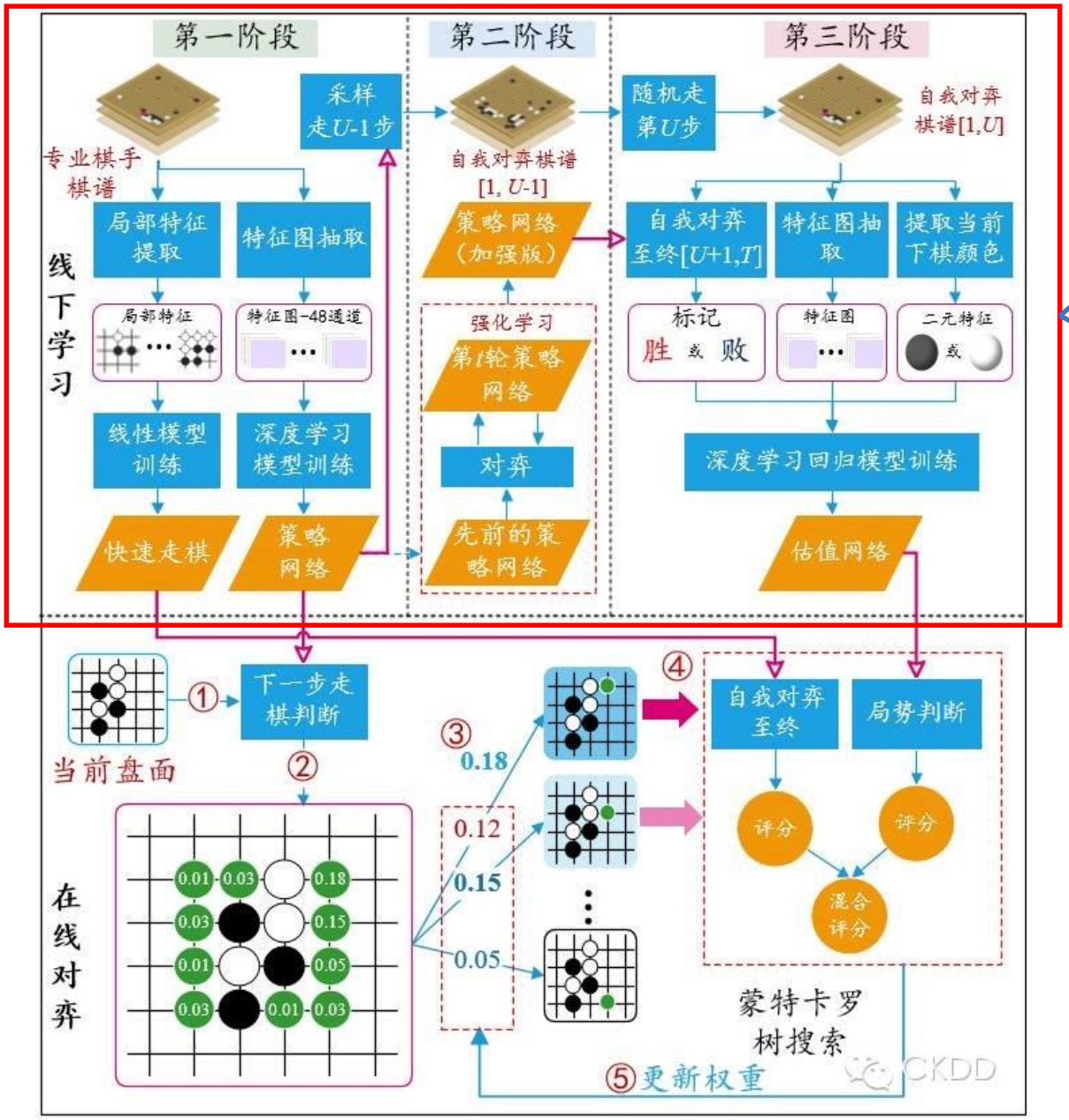
   失败 $u_1 = u_2 = \cdots = u_T = -1$

2. 损失函数 $L = \sum_{t=1}^{T} \frac{1}{2}[v(s_t; \mathbf{w}) - u_t]^2$

3. 更新值网络参数 $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \frac{\partial L}{\partial \mathbf{w}}$

AlphaGo

线下学习训练结束



加强版策略网络

值网络

AlphaGo

怎么利用策略网络和值网络进行在线对弈？

蒙特卡洛树搜索

AlphaGo

■ 蒙特卡洛树搜索MCTS

人类是如何来进行对弈的？

人类往往会进行多步虚拟推演

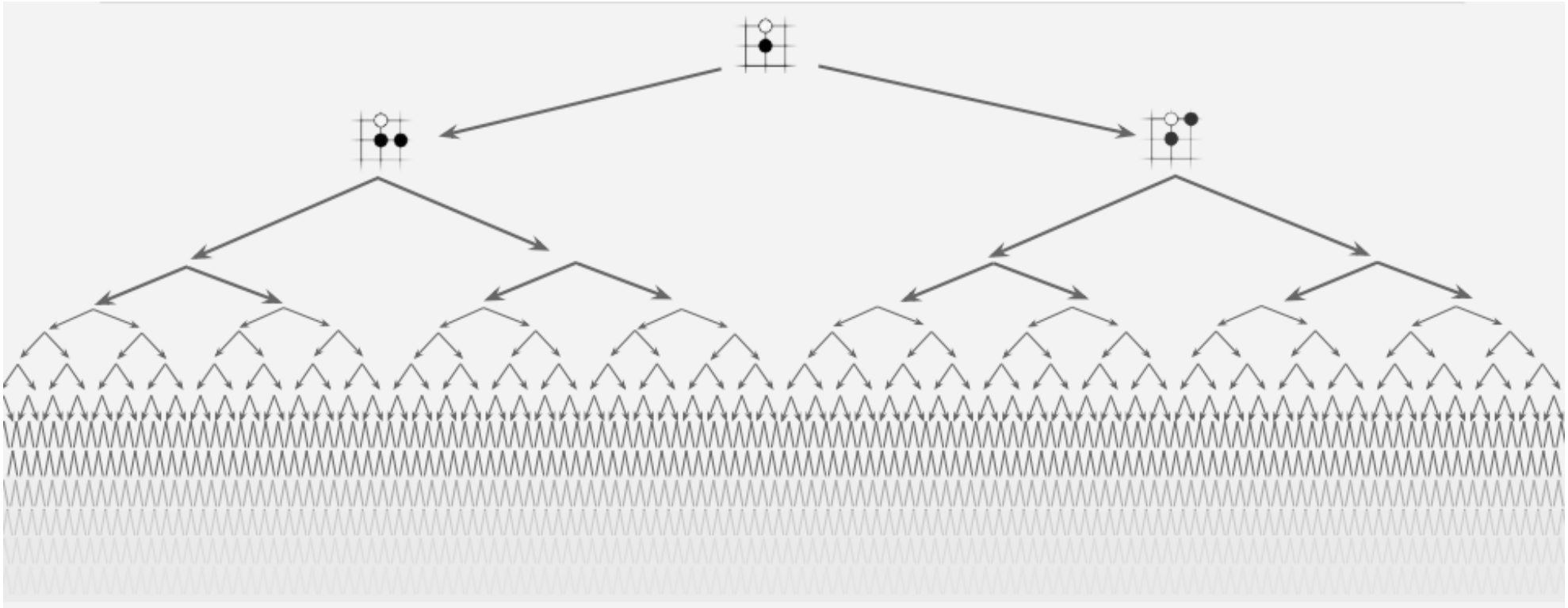
- 假如我选择动作a_t
- 对手最有可能采取什么动作？达到新状态s_{t+1}
- 在新一轮状态s_{t+1}下，假如我选择动作a_{t+1}
- 对手最有可能采取什么动作？达到新状态s_{t+2}

…

暴力搜索：如果在有限时间内可以穷尽所有可能，则玩家将立于不败之地

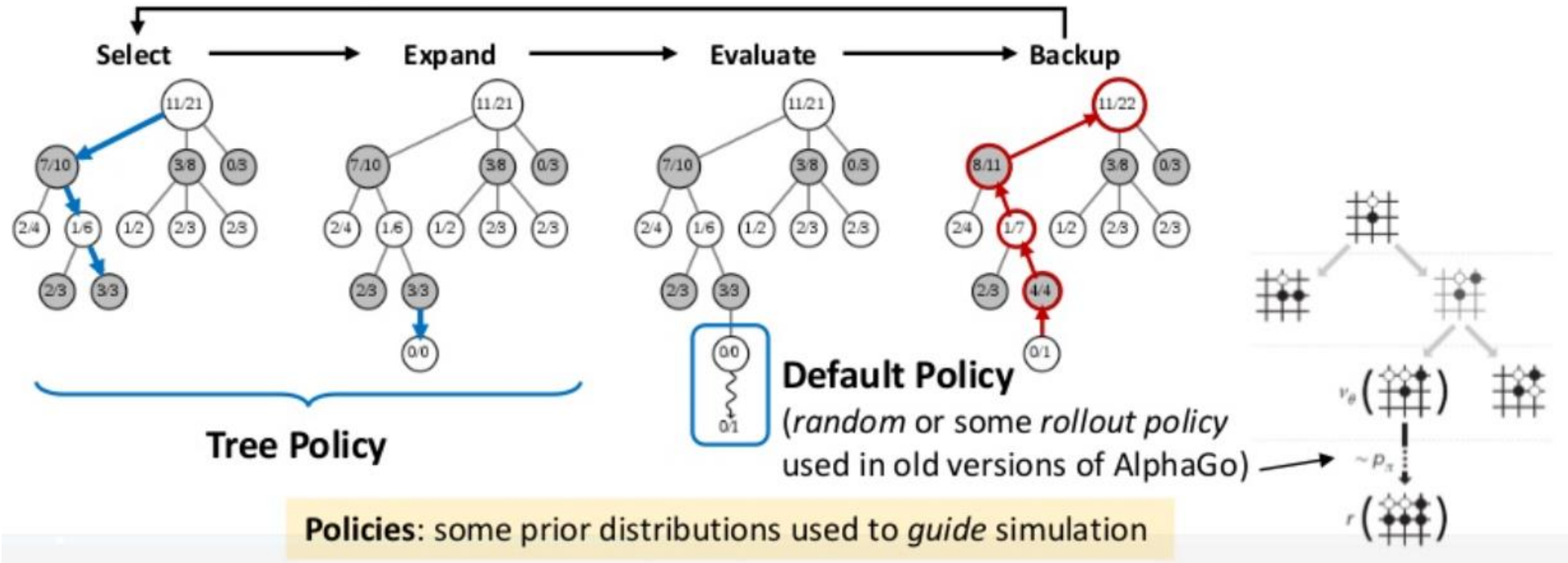
AlphaGo

- 随机选择动作a
- 向下搜索，统计当前动作a后续是输是赢
- 大量重复
- 统计不同动作获胜概率，选择胜率最高的动作



AlphaGo

■ 蒙特卡洛树搜索



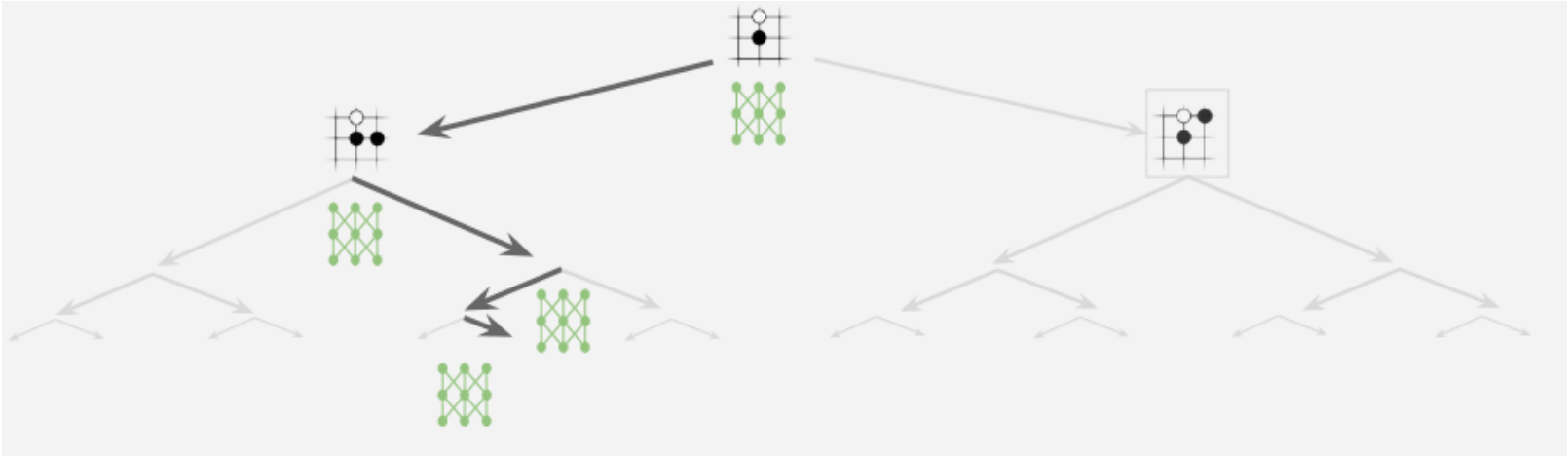
每个节点代表一个状态，而 A/B 代表这个节点被访问 B 次，黑棋胜利了 A 次。

AlphaGo

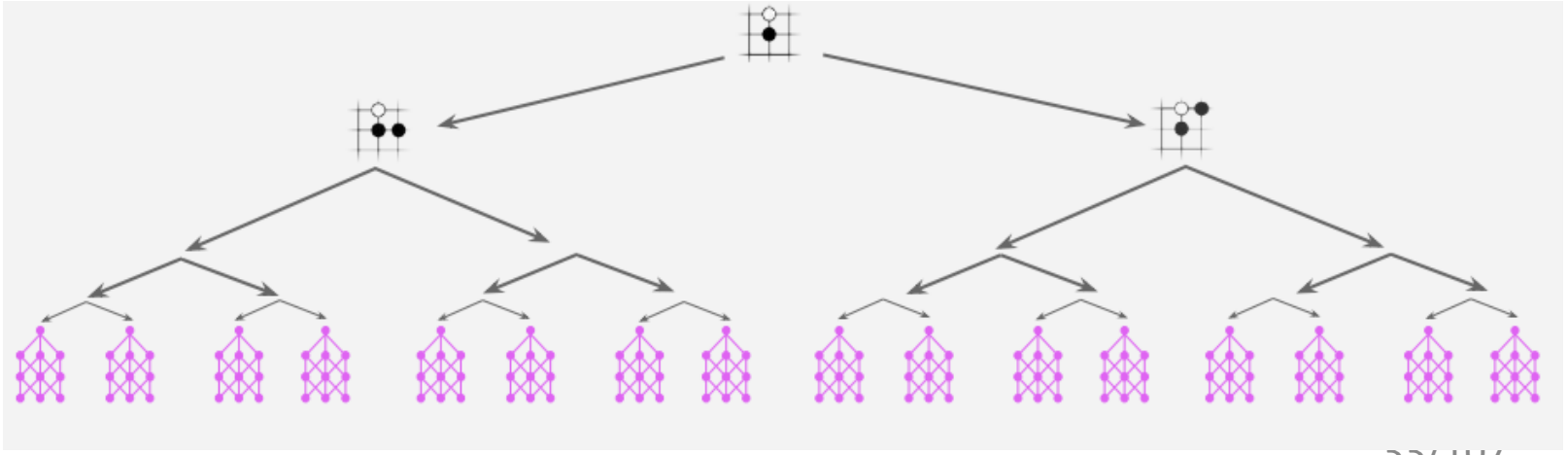
核心想法

在有限时时间内，借助策略网络和值网络来辅助搜索

利用策略网络减少搜索宽度



利用值网络增加估计的精度



AlphaGo

■ 蒙特卡洛树搜索

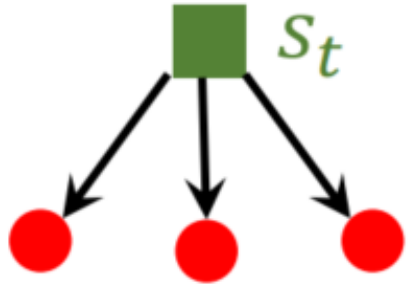
MCTS模拟的4个步骤

- 选择：玩家根据各动作的分数选择动作a;
- 扩展：对手执行动作(策略网络)，状态更新;
- 评估：评估状态值函数，得到评分ϑ，策略网络自我博弈至游戏结束，获得奖赏信号r, 给动作a打分$(\vartheta+r)/2$
- 回溯：利用得分来打分$(\vartheta+r)/2$更新动作值

AlphaGo

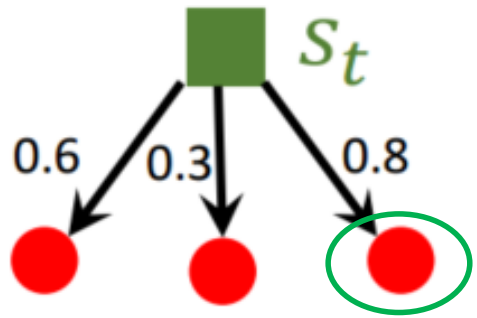
■ 蒙特卡洛树搜索

第一步：选择动作



1. 首先对于所有可行动作，计算其得分

$$\text{score}(a) = Q(a) + \eta \cdot \frac{\pi(a \mid s_t;\boldsymbol{\theta})}{1+N(a)}$$



$Q(a)$: 蒙特卡洛计算的动作平均值

$\pi(a \mid s_t;\boldsymbol{\theta})$: 在给定的状态 s_t 下 策略网络输出动作a的概率，

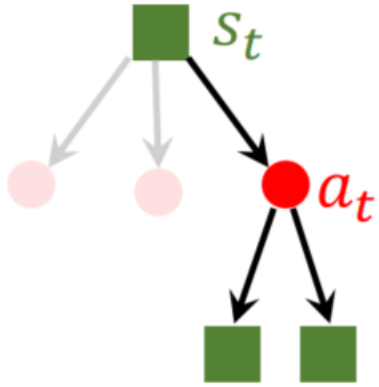
N(a)动作a选择的次数

2. 选择得分最高的动作

AlphaGo

■ 蒙特卡洛树搜索

第二步：扩展

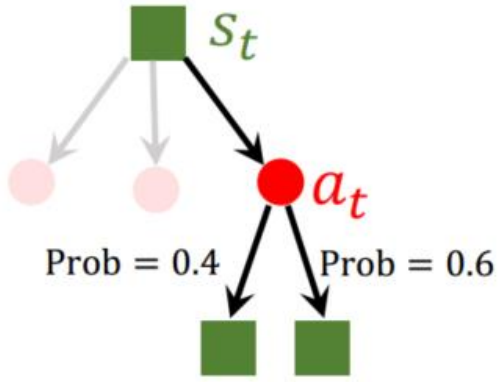


对手怎么采取动作呢？

给定我方动作a_t，对手采取动作a_t'后跳转至下一状态s_{t+1}

对手的动作a_t'根据策略网络采样

$$a_t' \sim \pi(\cdot \mid s_t'; \boldsymbol{\theta})$$



s_t'为对手观测到的状态

AlphaGo

■ 蒙特卡洛树搜索

第三步：评估

1. 模拟快速走子至游戏结束

快速走子网络

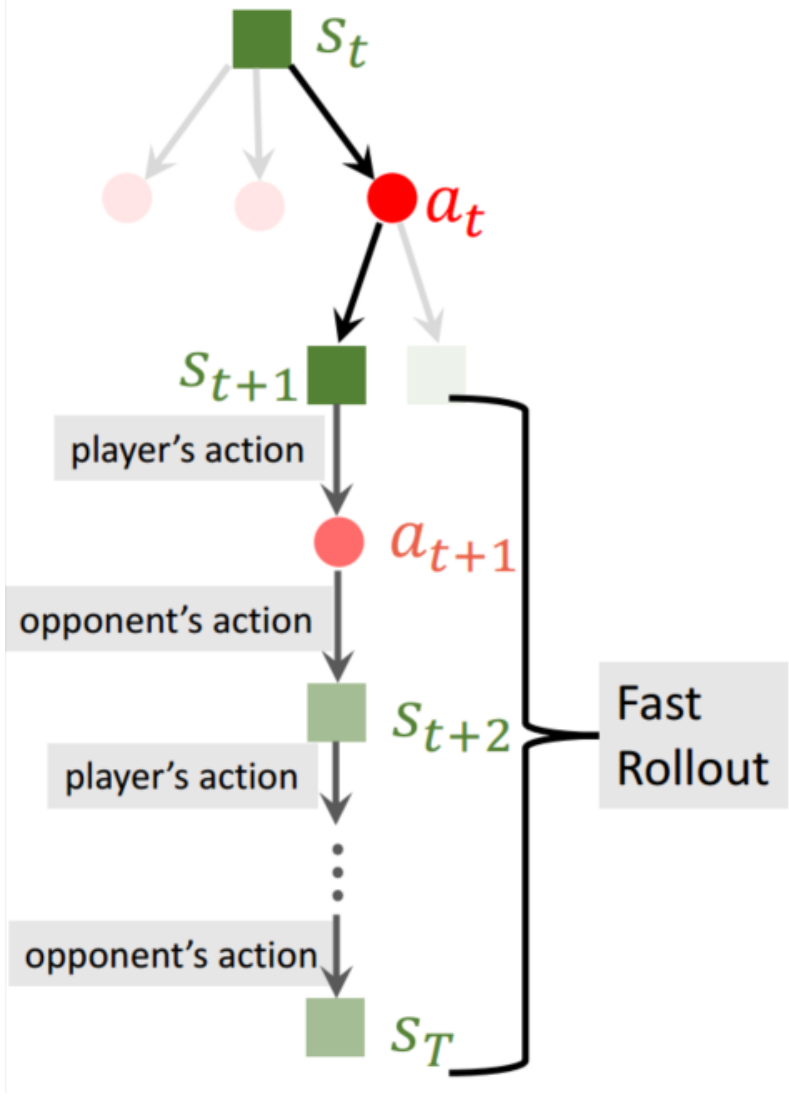
玩家的动作：$a_k \sim \pi(\cdot \mid s_k; \boldsymbol{\theta})$

对手的动作：$a'_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$

2. 得到最终奖赏

获胜 $r_T = +1$

失败 $r_T = -1$



AlphaGo

■ 蒙特卡洛树搜索

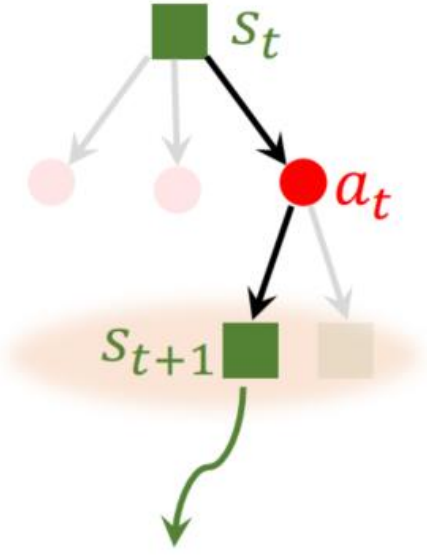
第三步：评估

1. 模拟快速走子至游戏结束

快速走子网络

玩家的动作：$a_k \sim \pi(\cdot \mid s_k; \boldsymbol{\theta})$

对手的动作：$a'_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$



2. 得到最终奖赏

获胜 $r_T = +1$

失败 $r_T = -1$

$$V(s_{t+1}) = \frac{1}{2} v(s_{t+1}; \mathbf{w}) + \frac{1}{2} r_T$$

3. 利用值网络 $v(s_{t+1}; \mathbf{w})$ 评估状态 s_{t+1}

AlphaGo

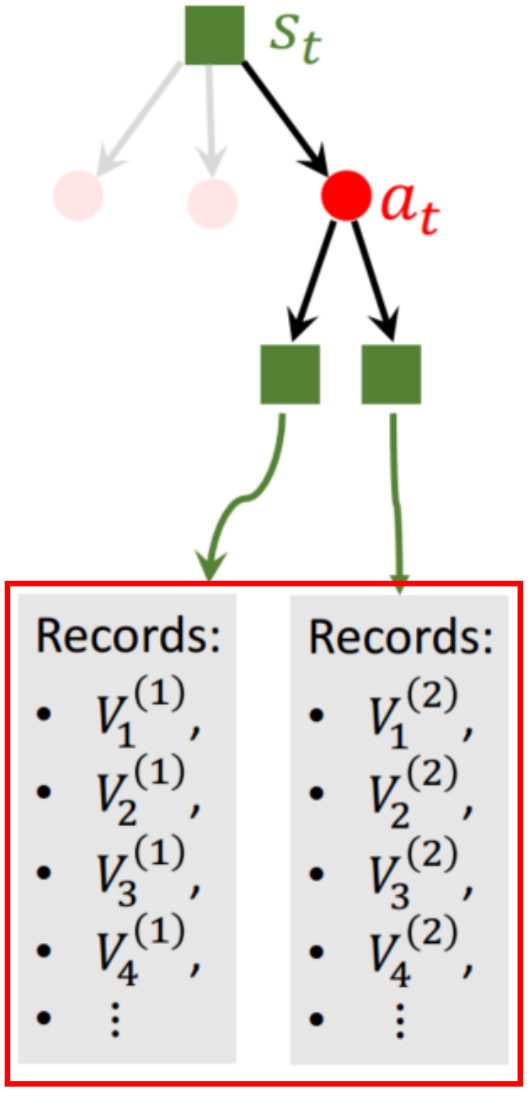
■ 蒙特卡洛树搜索

第四步：回溯

- MTCS重复大量次数的模拟
- 每一个动作节点a_t得到多次值估计
- 更新动作值

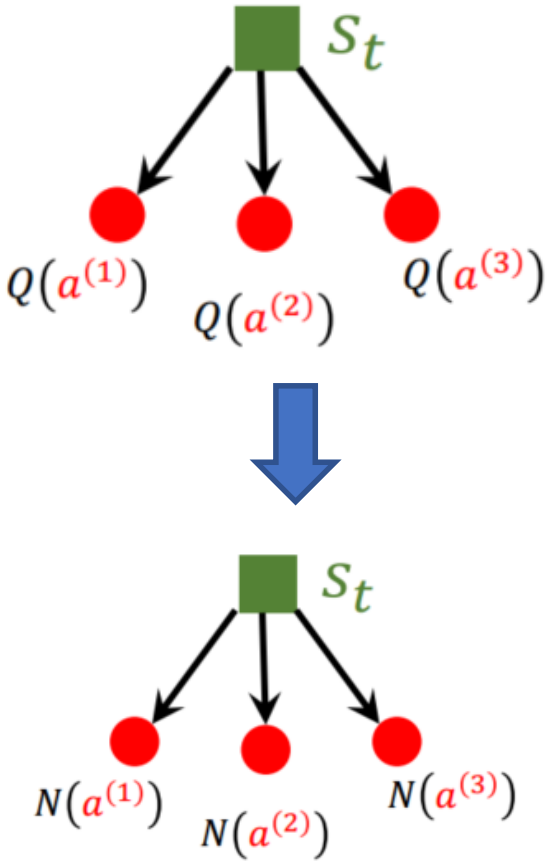
$$Q(a_t) = \text{mean}(\text{the recorded } V's)$$

- 该动作值Q(a_t) 用于第1步选择



AlphaGo

总结



选择得分最高的动作

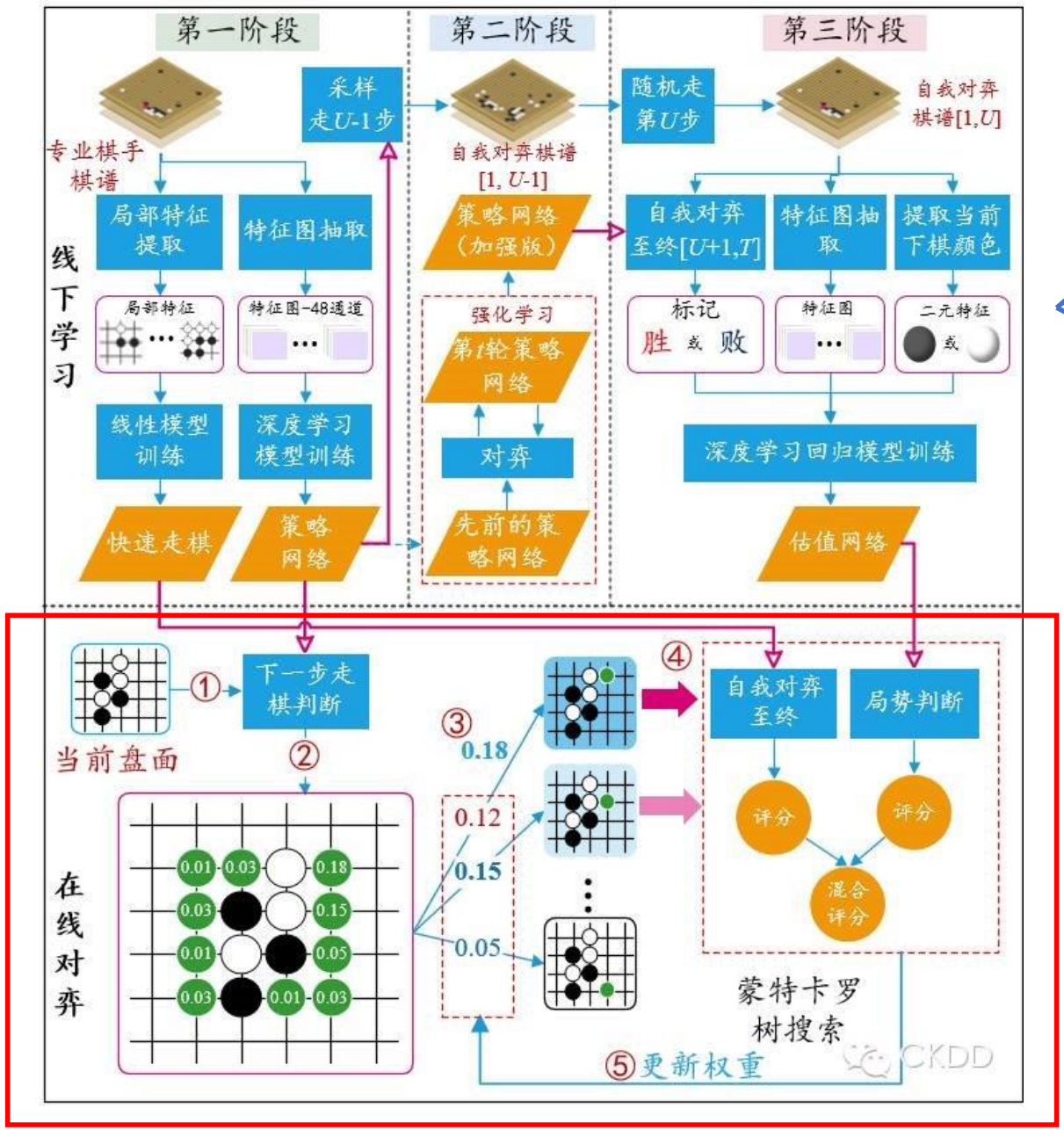
$$\text{score}(a) = Q(a) + \eta \cdot \frac{\pi(a \mid s_t;\boldsymbol{\theta})}{1+N(a)}.$$

MCTS搜索完后，玩家执行的动作为

$$a_t = \underset{a}{\text{argmax}}\, N(a)$$

AlphaGo

线下学习训练结束



加强版策略网络

值网络

在线对弈阶段

每一步均需要维护一个蒙特卡洛树来选取最终真正执行的动作

AlphaGo Zero

1年后DeepMind发表AlphaGo Zero

Versions	Hardware (inference time)	Elo rating	Matches
AlphaGo Fan	176 GPUs, distributed	3,144	5:0 against Fan Hui
AlphaGo Lee	48 TPUs, distributed	3,739	4:1 against Lee Sedol
AlphaGo Master	4 TPUs v2, single machine	4,858	60:0 against professional players; Future of Go Summit
AlphaGo Zero	4 TPUs v2, single machine	5,185	100:0 against AlphaGo Lee 89:11 against AlphaGo Master
AlphaZero	4 TPUs v2, single machine	N/A	60:40 against AlphaGo Zero

	AlphaGo Zero	AlphaGo Master	AlphaGo Lee	AlphaGo Fan
神经网络	1个共享网络	策略、价值、走子网络	策略、价值、走子网络	策略、价值、走子网络
Elo	5,185	4,858	3,739	3,144
运行阶段硬件需求	单机4块TPU	单机4块TPU	48块TPU+176块GPU	48 块TPU+176 块GPU
训练时间	40天(36小时Elo超越Lee)	未说明	数月	未说明
专家棋谱作用	未使用	AlphaGo Lee产生的棋谱	KGS数据集	KGS数据集

AlphaGo Zero

区别在于：

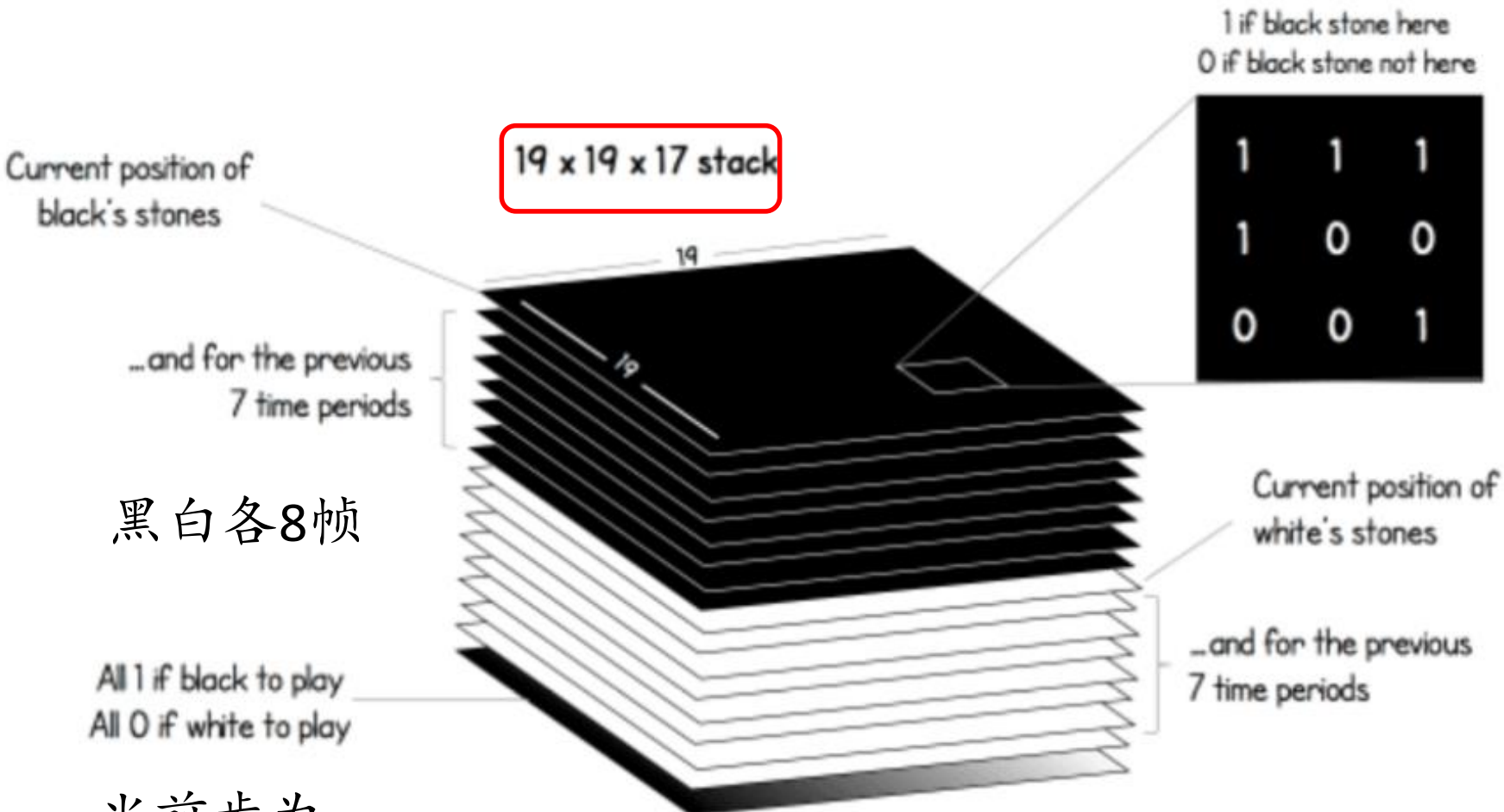
AlphaGo在线对弈时MCTS评分环节结合快速走子策略，只有业余玩家水平，会给MCTS的估计引入误差

AlphaGo Zero不再使用人类棋谱数据(没有模仿学习)，网络完全随机初始化，没有快速走子网络

- 状态由19*19*49 变更为 19*19*17，不再需要手工特征
- 策略网络与值网络不是单独的，参数共享+Multi-head
- 直接利用MCTS训练策略网络和值网络

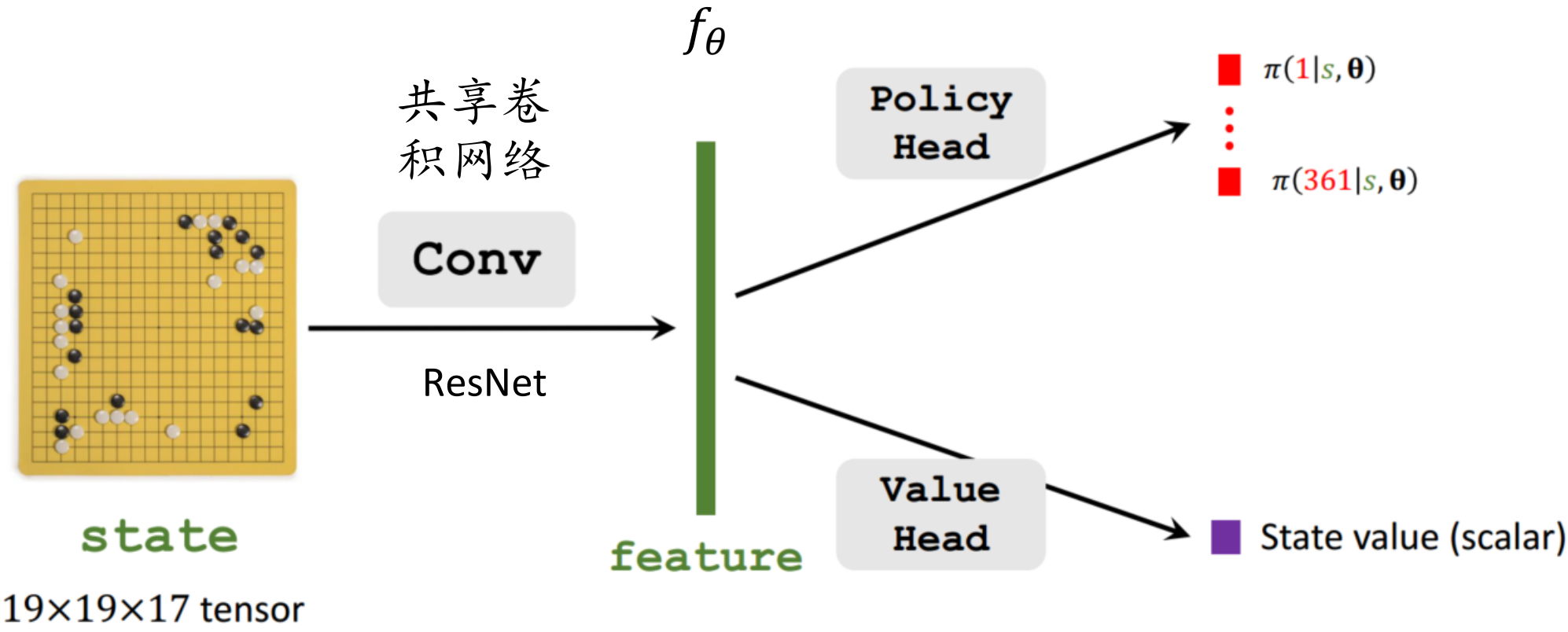
AlphaGo

- AlphaGo Zero网络的输入状态 19*19*17



AlphaGo Zero

■ AlphaGo Zero神经网络结构：参数共享+多头网络



AlphaGo Zero

■ AlphaGo Zero的整体训练流程

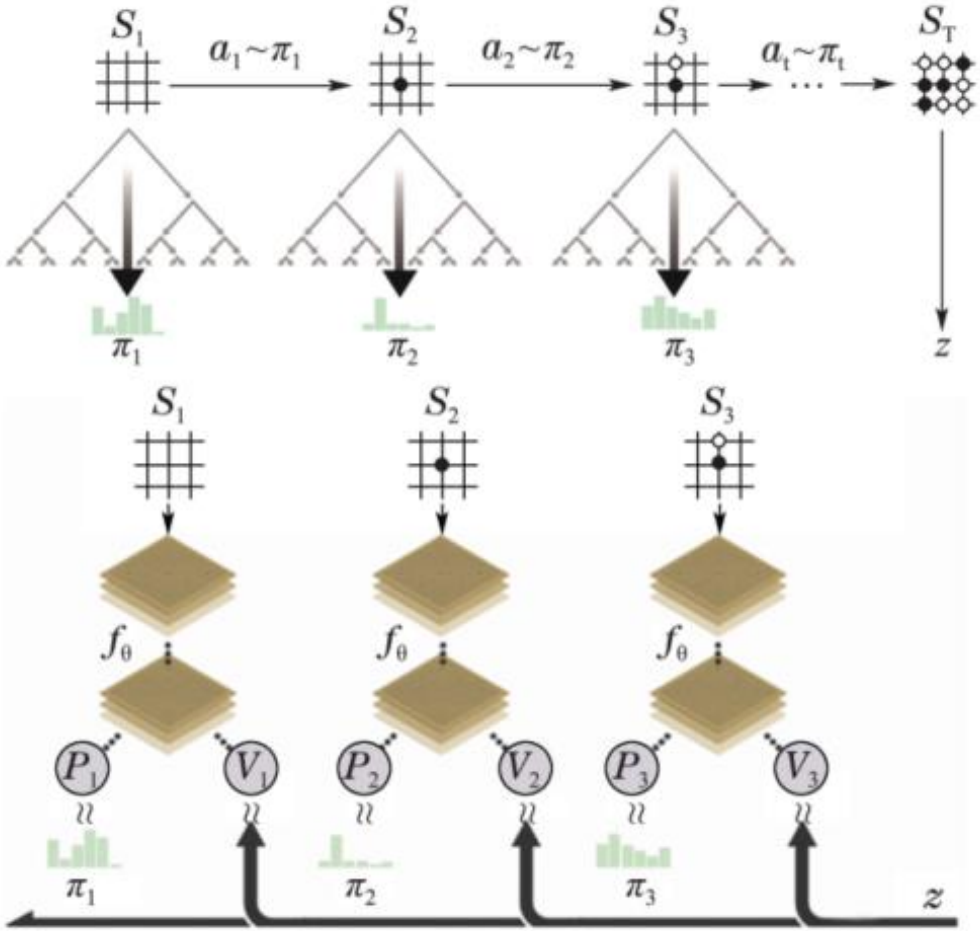
1. 随机初始化神经网络f_θ；

2.在每一步s下经神经网络引导产生MCTS策略π (策略提升)

3.自我博弈至终局，得到最终胜负奖励z (策略评估)

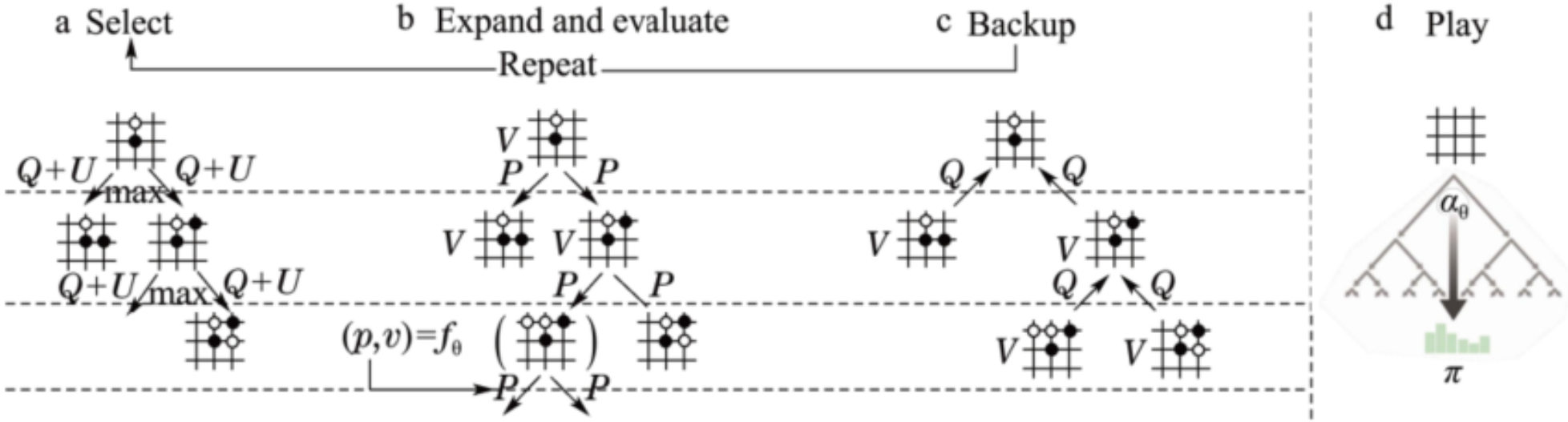
4.得到大量(s,z,π)训练数据训练策略网络和值网络

$$l = (z - v_t)^2 - \pi_t^T \log p_t + c||\theta||^2$$



AlphaGo Zero

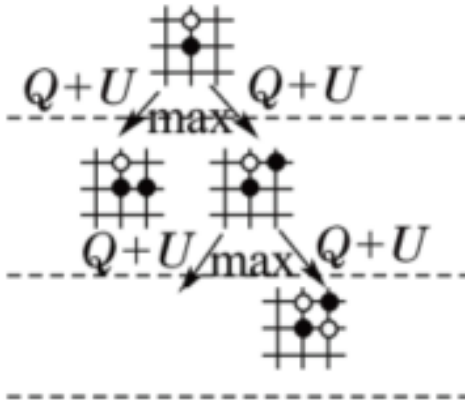
- AlphaGo Zero中神经网络引导下的MCTS策略π



AlphaGo Zero

■ AlphaGo Zero中神经网络引导下的MCTS策略π

第1步：选择



$$a_t = \arg\max_a (Q(s_t,a) + U(s_t,a))$$

$$U(s_t,a) = c_{\text{puct}} P(s_t,a) \frac{\sqrt{\sum_b N(s_t,b)}}{1+N(s_t,a)}$$

各条边$e(s,a)$存储四元集

- 遍历次数$N(s,a)$
- 动作累计值$W(s,a)$
- 动作平均值$Q(s,a)$
- 策略输出概率$P(s,a)$

$Q = W/N$，反映胜率

c_{puct}权衡探索与利用间的权重分配，当c_{puct}较大时，树搜索倾向于向未知区域探索

$\sum_b N(s_t,b)$表示经过状态s_t的所有次数

AlphaGo Zero

■ AlphaGo Zero中神经网络引导下的MCTS策略π

第2步：扩展与评估

搜索树的叶子节点s_l，进行扩展与评估

由神经网络f_θ得到s_l下动作a的概率值p_l和值网络输出v_l

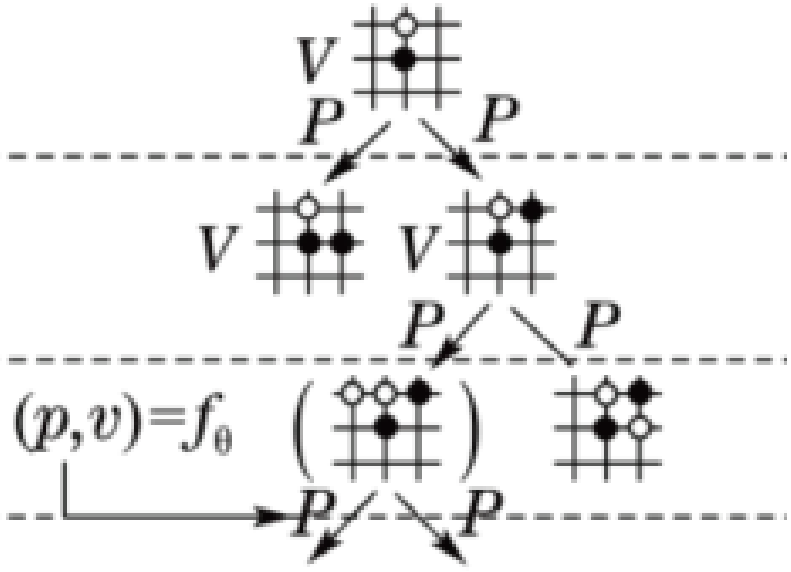
初始化边$e(s_l, a)$中的四元集

遍历次数$N(s_l, a) = 0$

累计动作价值$W(s_l, a) = 0$

平均动作价值$Q(s_l, a) = 0$

策略输出概率$P(s_l, a) = p_l$



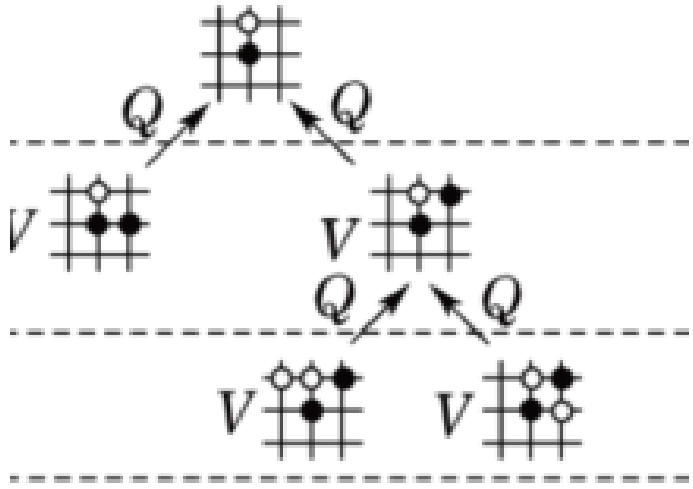
AlphaGo Zero

- AlphaGo Zero中神经网络引导下的MCTS策略π

第3步：回溯

叶子节点s_l的信息到根节点整个搜索路径上的节点均更新

$$N(s_t,a_t)=N(s_t,a_t)+1,$$
$$W(s_t,a_t)=W(s_t,a_t)+v_t,$$
$$Q(s_t,a_t)=\frac{W(s_t,a_t)}{N(s_t,a_t)},$$



随着模拟次数的增加，动作平均值$Q(s_t,a_t)$将逐渐趋于稳定。

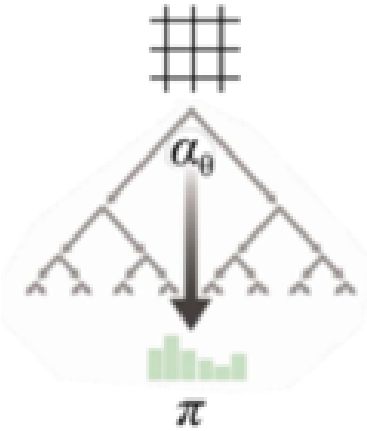
AlphaGo Zero

- AlphaGo Zero中神经网络引导下的MCTS策略π

落子执行Play阶段使用MCTS策略π

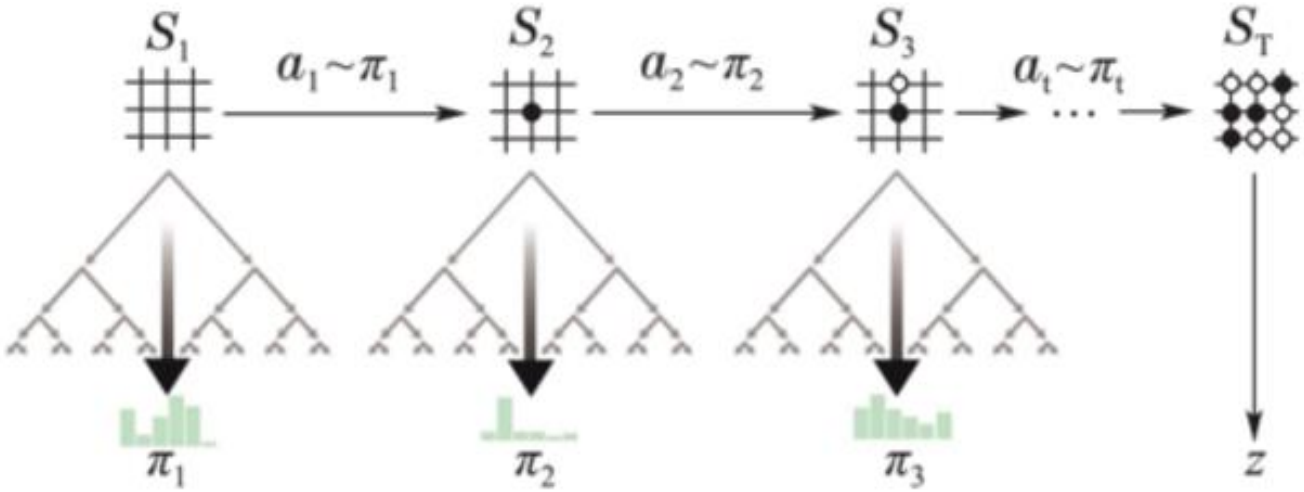
落子前经过1600次蒙特卡洛树搜索，树中各边存储着历史信息，根据这些信息得到落子概率分布π

$$\pi(a|s_0) = \frac{N(s_0,a)^{\frac{1}{\tau}}}{\sum_b N(s_0,b)^{\frac{1}{\tau}}}$$



引入了模拟退火算法，可极大地丰富围棋开局的变化情况。

AlphaGo Zero



MCTS策略π性能优于神经网络策略p

使用基于自对弈最终对局的z作为价值

$$l = (z - v_t)^2 - \pi_t^T \log p_t + c||\theta||^2$$

策略评估　　策略提升

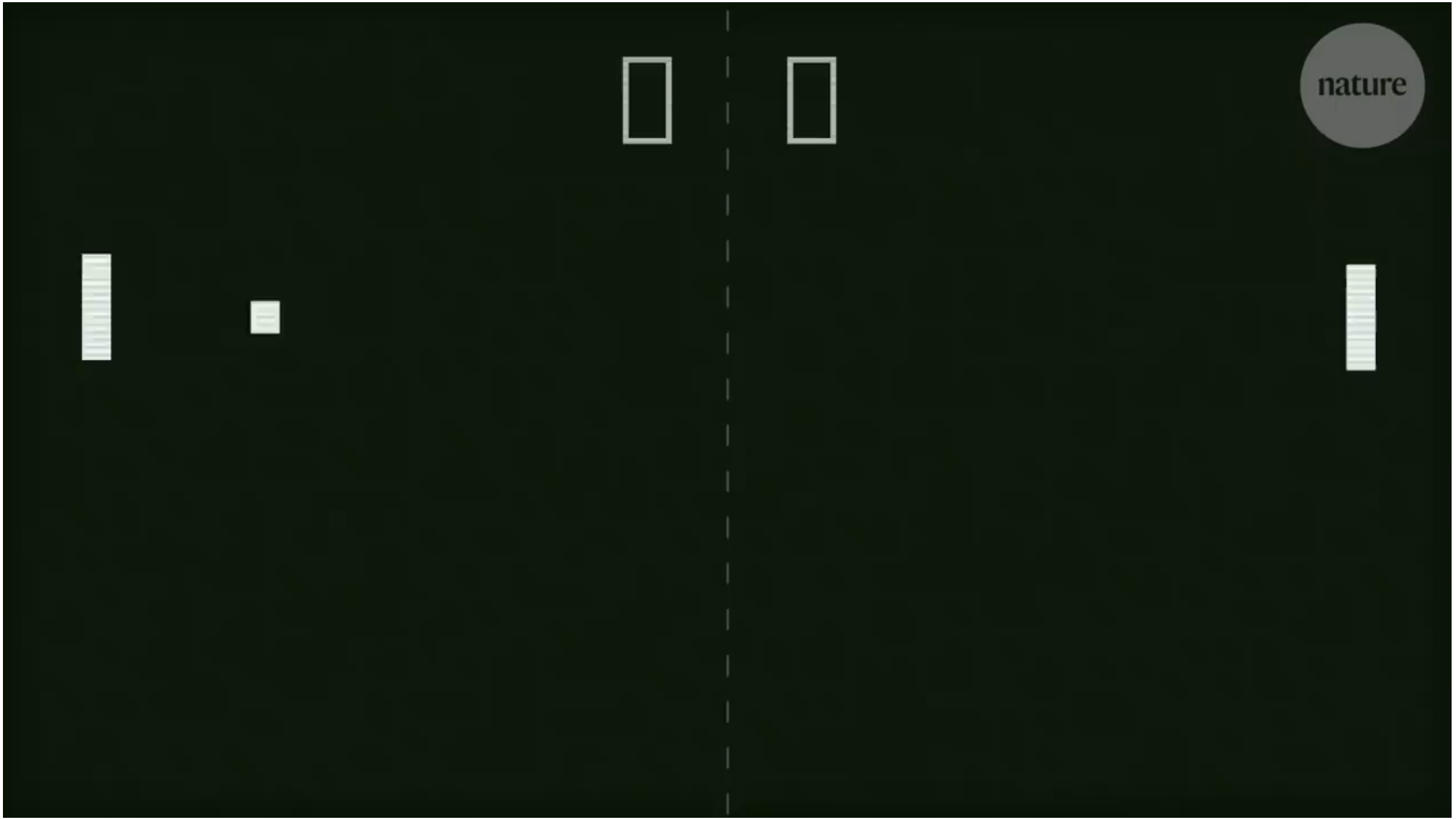
以π,z 为神经网络的标签

课程目录

- 围棋系列
  - AlphaGo
  - AlphaGoZero
- 实时策略游戏系列
  - AlphaStar
- 总结

AlphaStar

AlphaStar



AlphaStar

星际争霸2：人工智能的重大挑战

1、 博弈——星际争霸具有丰富的策略博弈过程，没有单一的最佳策略。因此，智能体需要不断的探索，并根据实际情况谨慎选择对局策略。

2、 非完全信息——战争迷雾和镜头限制使玩家不能实时掌握全场局面信息和迷雾中的对手策略。

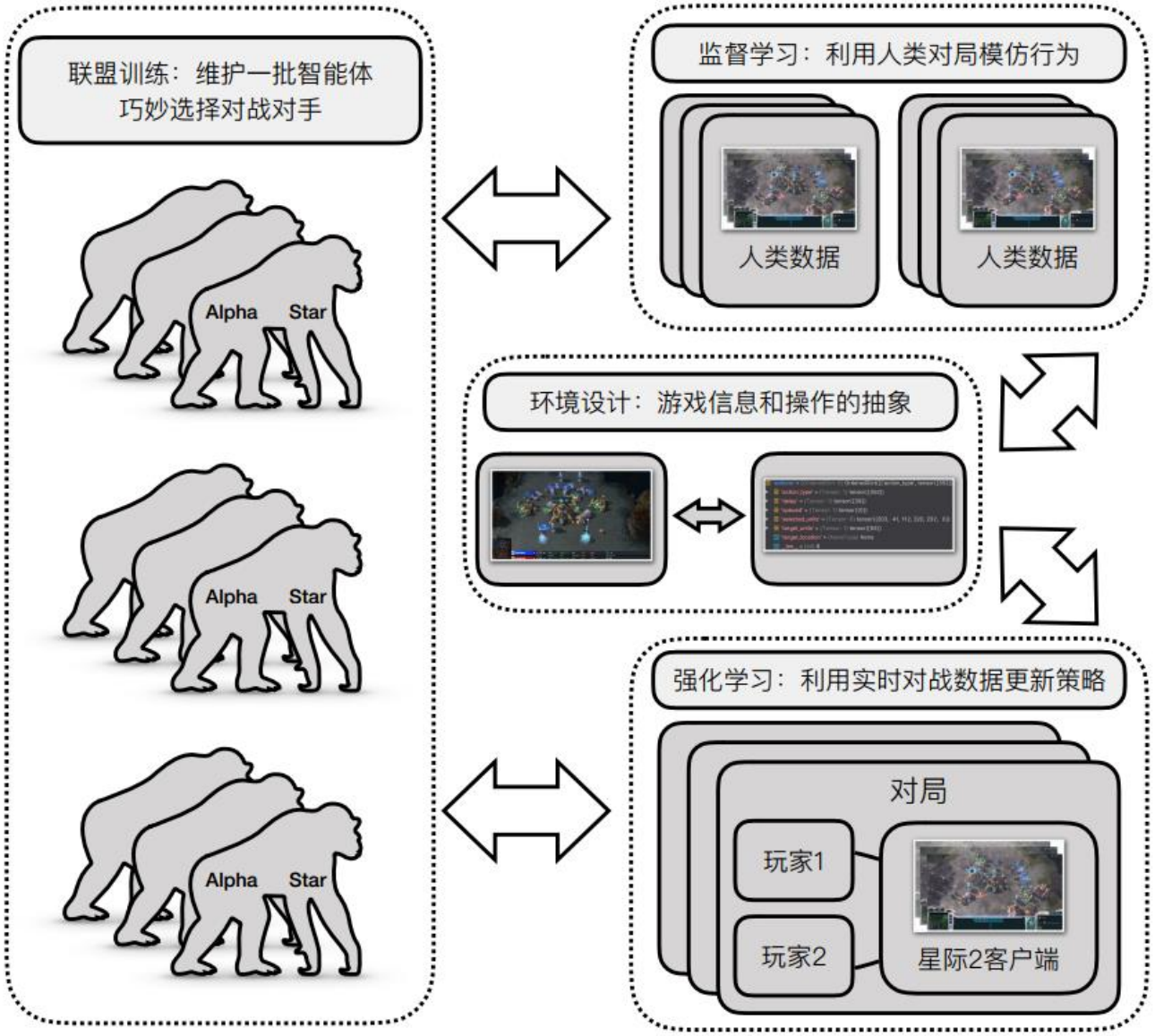
3、 长期规划——与国际象棋和围棋等不同，星际争霸的因果关系并不是实时的，早期不起眼的失误可能会在关键时刻暴露。

4、 实时决策——星际争霸的玩家根据实时情况进行决策动作。

5、 巨大动作空间——必须实时控制不同区域下的数十个单元和建筑物，并且可以组成数百个不同的操作集合。因此由小决策形成的可能组合动作空间巨大，每局游戏数千次动作。

6、 三种不同种族——不同种族的宏机制对智能体的泛化能力提出挑战。

AlphaStar



Vinyals, Oriol, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." Nature, 2019, https://doi.org/10.1038/s41586-019-1724-z.

AlphaStar

AlphaStar的设计过程

1. 状态与动作空间及网络设计
2. 监督学习/模仿学习
3. 强化学习
4. 虚拟自我博弈与联盟学习

AlphaStar

■ AlphaStar的状态表征

Category	Field	Description
实体 (列表形式) Entities: up to 512	Unit type	E.g. Drone or Forcefield
	Owner	Agent, opponent, or neutral
	Status	Current health, shields, energy
	Display type	E.g. Snapshot, for opponent buildings in the fog of war
	Position	Entity position
	Number of workers	For resource collecting base buildings
	Cooldowns	Attack cooldown
	Attributes	Invisible, powered, hallucination, active, in cargo, and/or on the screen
	Unit attributes	E.g. Biological or Armored
	Cargo status	Current and maximum amount of cargo space
	Building status	Build progress, build queue, and add-on type
	Resource status	Remaining resource contents
	Order status	Order queue and order progress
	Buff status	Buffs and buff durations
地图 (图像形式) Map: 128x128 grid	Height	Heights of map locations
	Visibility	Whether map locations are currently visible
	Creep	Whether there is creep at a specific location
	Entity owners	Which player owns entities
	Alerts	Whether units are under attack
	Pathable	Which areas can be navigated over
	Buildable	Which areas can be built on
玩家数据 Player data	Race	Agent and opponent requested race, and agent actual race
	Upgrades	Agent upgrades and opponent upgrades, if they would be known to humans
	Agent statistics	Agent current resources, supply, army supply, worker supply, maximum supply, number of idle workers, number of Warp Gates, and number of Larva
游戏统计 Game statistics	Camera	Current camera position. The camera is a 32x20 game-unit sized rectangle
	Time	Current time in game

AlphaStar

- AlphaStar的动作空间

分层动作空间

Field	Description
Action type	Which action to execute. Some examples of actions are moving a unit, training a unit from a building, moving the camera, or no-op. See PySC2 for a full list[7]
Selected units	Entities that will execute the action
Target	An entity or location in the map discretised to 256x256 targeted by the action
Queued	Whether to queue this action or execute it immediately
Repeat	Whether or not to issue this action multiple times
Delay	The number of game time-steps to wait until receiving the next observation

动作类型

选中执行单位

目标

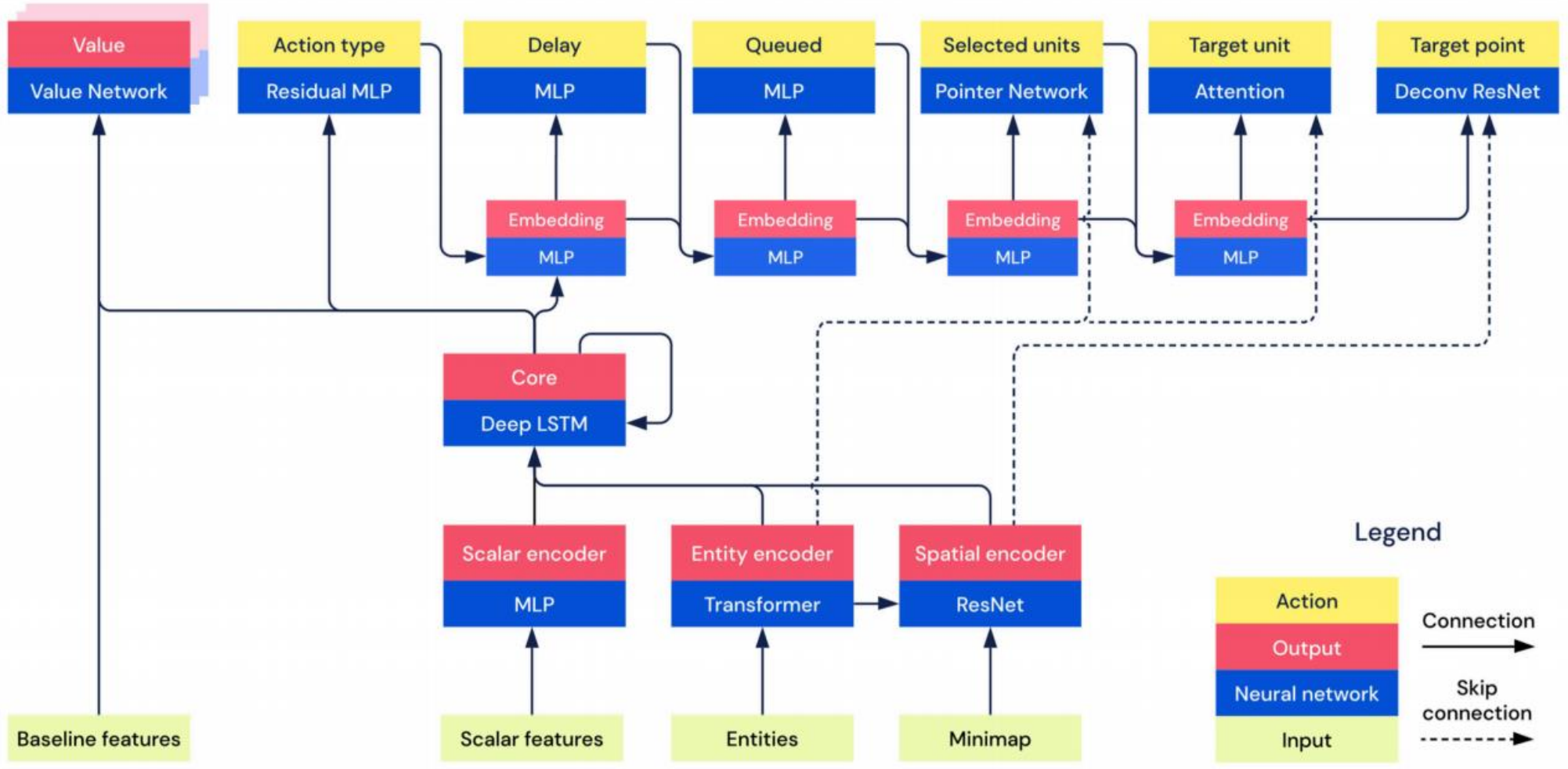
是否立即执行

是否重复动作

等待多久后接收下一次输入

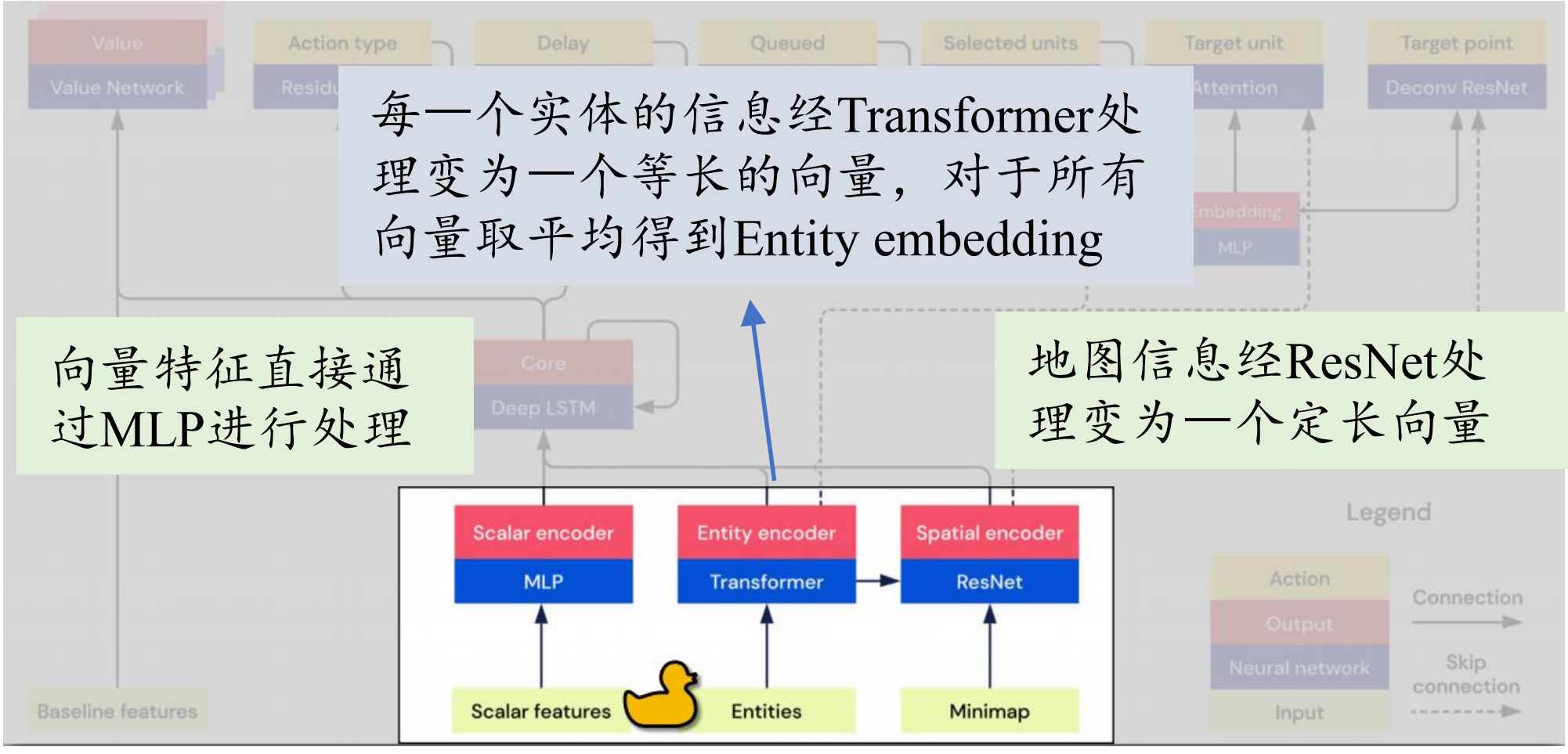
AlphaStar

- AlphaStar的网络结构



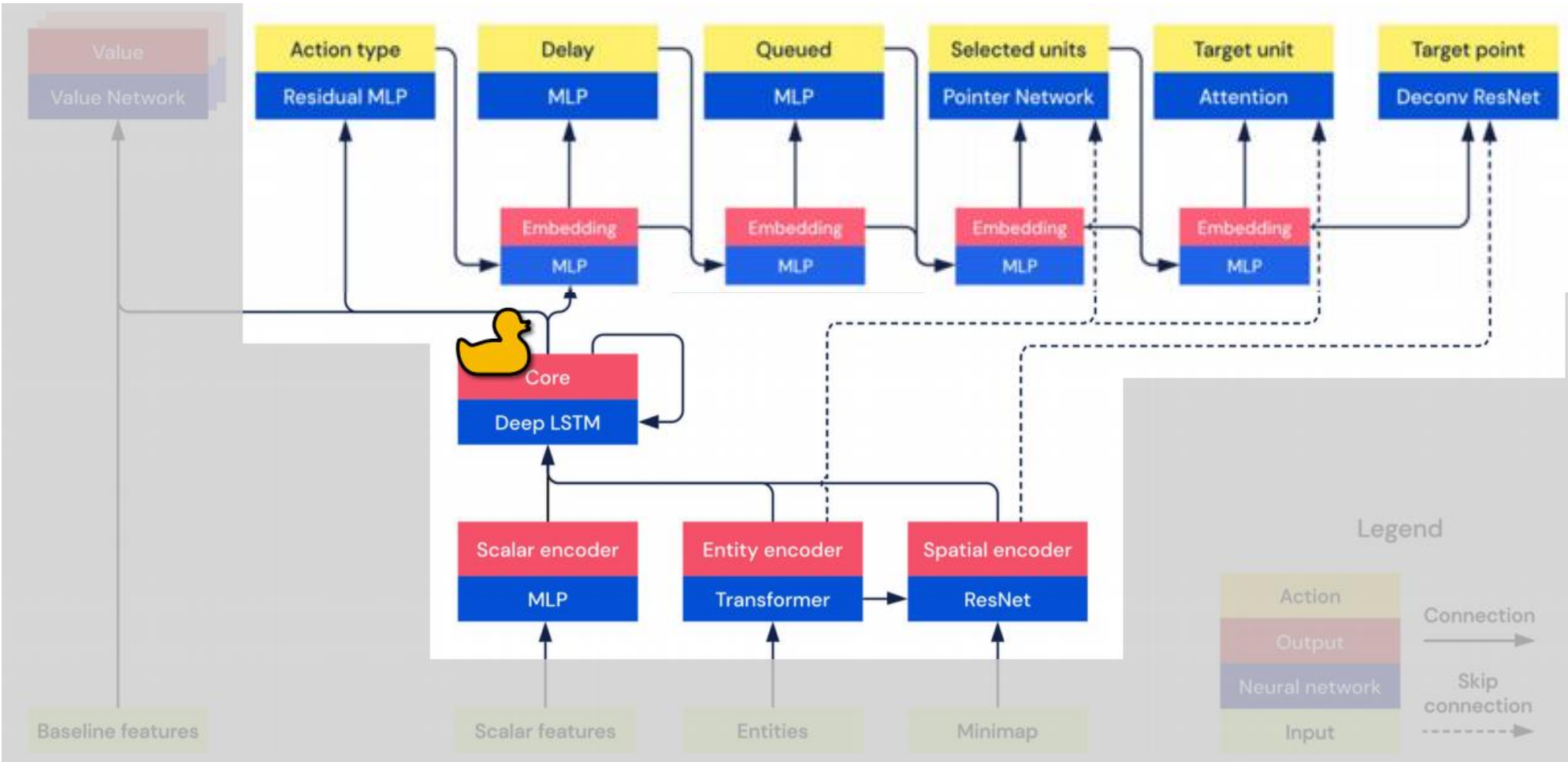
AlphaStar

- AlphaStar的网络结构



AlphaStar

- AlphaStar的网络结构



AlphaStar

■ 监督学习：解决神经网络初始化的问题

目的：与AlphaGo类似，利用大量的人类数据学习一个比较好的网络初始化参数

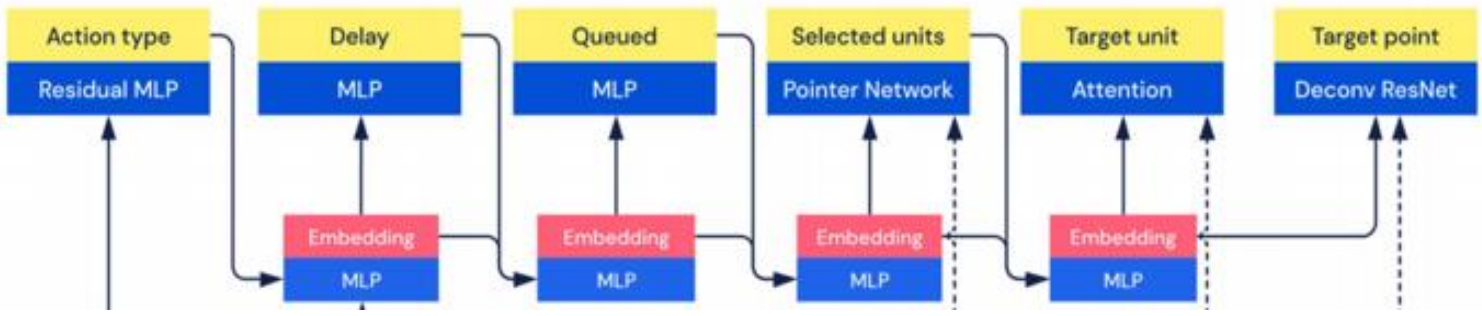
具体流程：

1. 解码人类对局，得到每帧一致的状态表征

2. 将该状态表征喂给神经网络，得到输出动作的概率分布(每个head前面的head采取人类选择)

3. 利用人类动作标签计算KL loss，利用梯度下降更新网络参数 (每个head分别计算loss)

交叉熵 loss



MSE loss

AlphaStar

■ 强化学习：利用Actor-Critic提升主智能体的性能

强化学习奖励

奖励	用途	计算	备注
胜负奖励	鼓励胜利	赢+1，平0，输–1	只在最后一刻给
建造顺序(Build order)	模仿人类	编辑距离	人类数据给出参考
建造单位(Built Units)	模仿人类	汉明距离	同上
科技升级(Upgrades)	模仿人类	汉明距离	同上
技能(Effects)	模仿人类	汉明距离	同上

AlphaStar

■ 强化学习：利用Actor-Critic提升主智能体的性能

编辑距离(Edit Distance)

人类建造顺序是ABCD

智能体的建造顺序是BACD

把BACD变成ABCD所需最小操作是2次

(BACD变成AACD再变成ABCD)

所以建造顺序奖励是-2

汉明距离(Hamming Distance)

场上有ABCD四种建筑，各用0或1表示是否建造

人类造了ABC，表示为1110

智能体造了ABD，表示为1101

两者汉明距离为2（两处不同）

“建造单位奖励”为-2

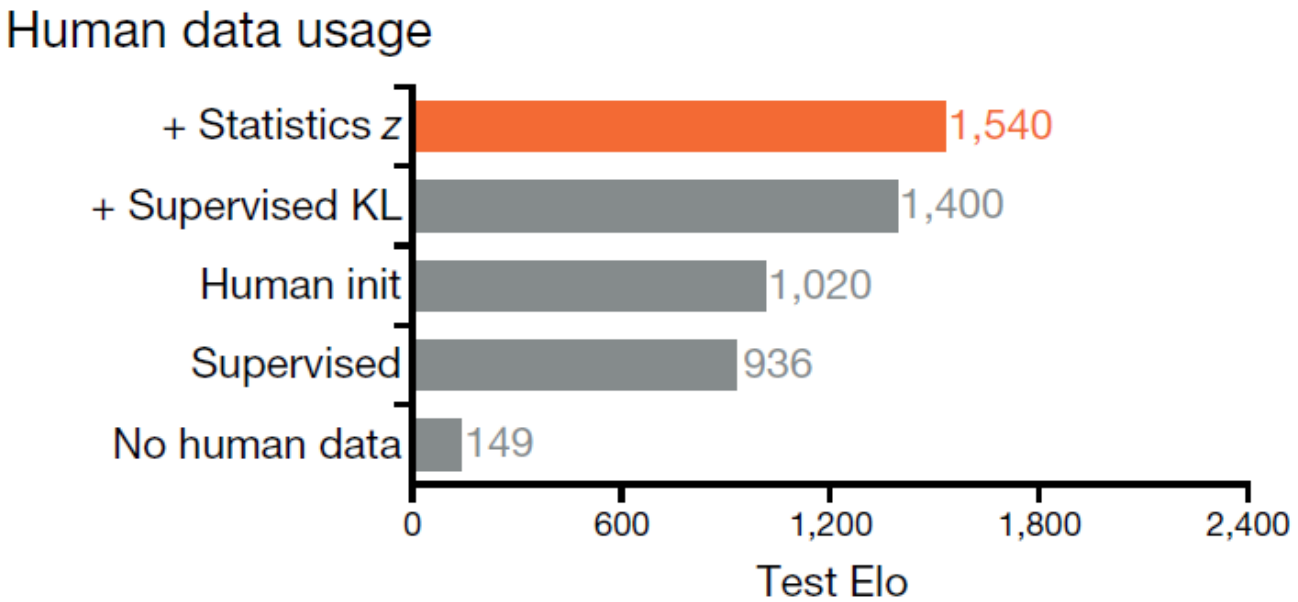
AlphaStar

统计量Z实际上是一批数据：

- 前20个建造的建筑物和单位
- 整局游戏中建造的单位、建筑，升级的科技和技能（用布尔向量表示）

统计量Z作为神经网络的输入传给策略网络和价值网络

统计量Z的引入有助于学习多种战术



AlphaStar

■ 强化学习

动作空间 高度复杂	奖励 稀疏	非完全信息/长时决策 等导致价值函数难拟合

1. V-trace：解决大动作空间异步并行时的数据利用问题

2. UPGO用于改善稀疏回报问题。当状态动作值估计高于状态值估计时，UPGO会直接跳过当前状态值而采用下个状态的估计值进行反传。

3. 利用TD(λ)来训价值网络，并同时输入对手数据，处理非完全信息并改善稀疏回报问题

AlphaStar

■ V-trace

优势函数 $A^{\pi_\theta}(s_t, a_t) = [r(s_t, a_t) + V^{\pi_\theta}(s_{t+1})] - V^{\pi_\theta}(s_t)$

策略梯度 $\nabla_\theta J = \mathbb{E}_{\pi_\theta}[A^{\pi_\theta}(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)]$

当采集数据的行为策略π_μ与优化策略π_θ不一致时，
可利用重要性采样off-policy学习进行训练

$$\nabla_\theta J = \mathbb{E}_{\pi_\mu}[\frac{\pi_\theta(a_t|s_t)}{\pi_\mu(a_t|s_t)} A(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)]$$

V-trace

$$\nabla_\theta J = \mathbb{E}_{\pi_\mu}[\rho_t A^{\pi_\theta}(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)]$$

$$\rho_t = \min(\frac{\pi_\theta(a_t|s_t)}{\pi_\mu(a_t|s_t)}, 1)$$

AlphaStar

- UPGO：upgoing policy update

优势函数 $A^{\pi_\theta}(s_t, a_t) = [r(s_t, a_t) + V^{\pi_\theta}(s_{t+1})] - V^{\pi_\theta}(s_t)$

$$A^{\pi_\theta}(s_t, a_t) = G_t^U - V_{\pi_\theta}(s_t)$$

$$G_t^U = \begin{cases} r_t + G_{t+1}^U & \text{if } Q(s_{t+1}, a_{t+1}) \geq V(s_{t+1}) \\ r_t + V(s_{t+1}) & \text{otherwise} \end{cases}$$

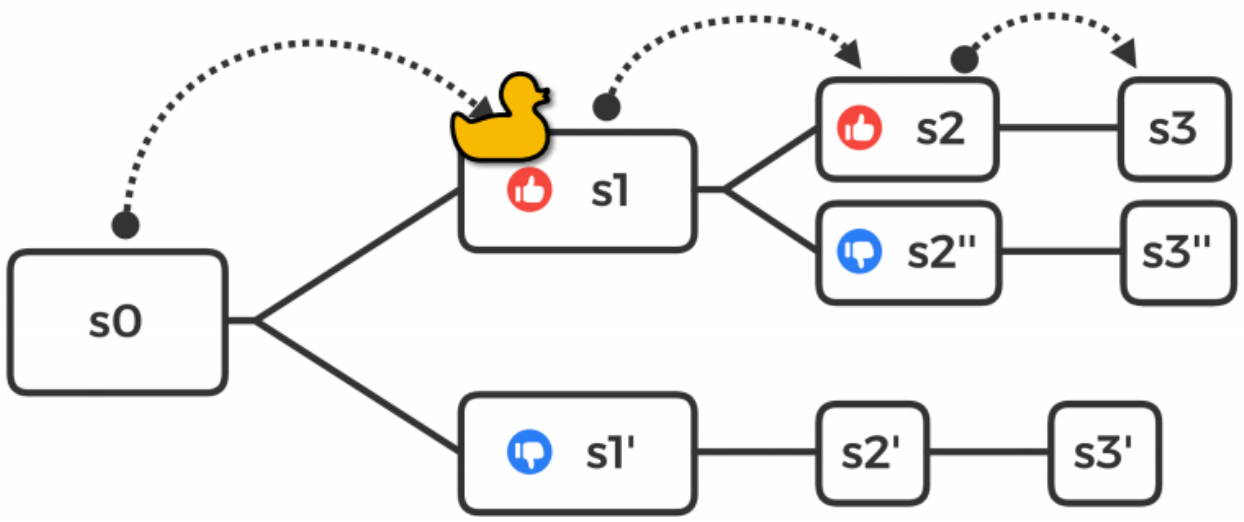
乐观估计

$$V(s_t) = \mathbb{E}_{a_t}[r(s_t, a_t) + V(s_{t+1})]$$

$$Q(s_t, a_t) = r(s_t, a_t) + V(s_{t+1})$$

AlphaStar

- UPGO：upgoing policy update



若a_0是个好动作，此时G_0变成$r_0 + G_1$;
若a_1仍是好动作，此时G_0变成$r_0 + r_1 + G_2$;

将多步以后未来的信息纳入现在的Advantage估计中

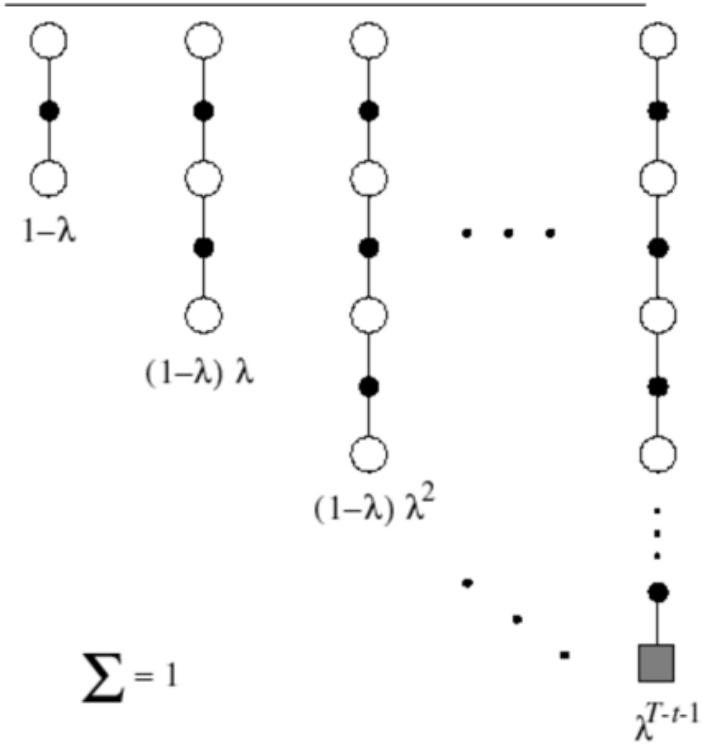
AlphaStar

- TD(λ)

$$V^{\pi_\theta}(s_t) = \mathbb{E}_{\pi_\theta} \sum_{t'=t} \gamma^{t'-t} r(s_t, a_t) = \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)} [r(s_t, a_t) + \gamma V(s_{t+1})]$$

TD(0) $\quad L = [(r_t + \gamma V_{t+1}) - V_t]^2$

TD(λ), λ-回报



- λ-回报 G_t^λ 将所有的 n-步回报 $G_t^{(n)}$ 整合在一起，$0 < \lambda < 1$
- 对每项使用权重 $(1-\lambda)\lambda^{n-1}$

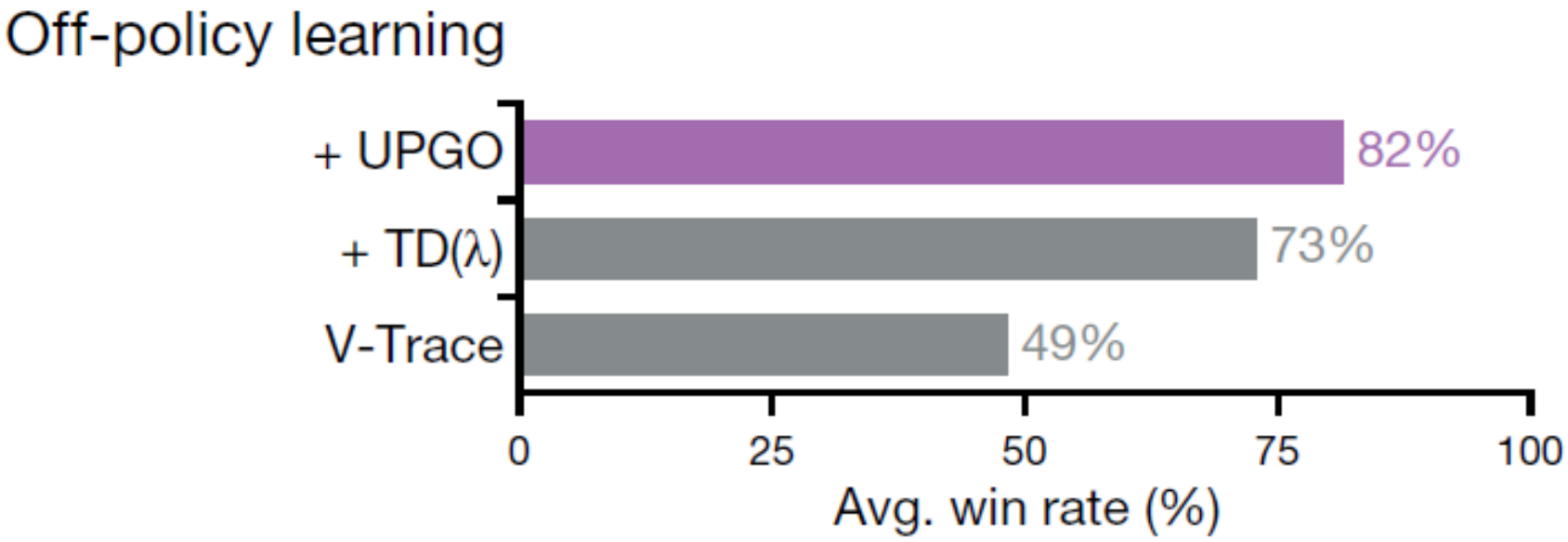
$$G_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

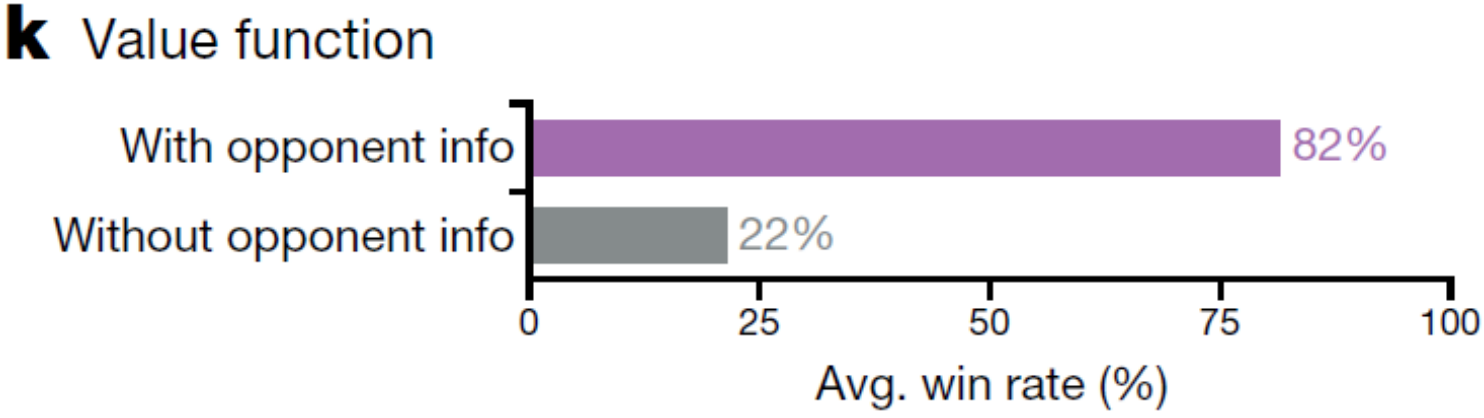
- 前向更新的 TD(λ) 形式

$$V(s_t) \leftarrow V(s_t) + \alpha\left(G_t^\lambda - V(s_t)\right)$$

AlphaStar

性能分析





4. 虚拟自我博弈与联盟学习

AlphaStar

Transitive game⊕ Non-Transitive game

1. 实力/阵容差距过大，博弈很难得到提升

2. 在一定阶段，玩家开始学会不同策略风格，利用相生相克关系，操作能力得到提升

3. 随着玩家遇到过很多阵容/策略后，能力进一步提升，进入传导博弈阶段

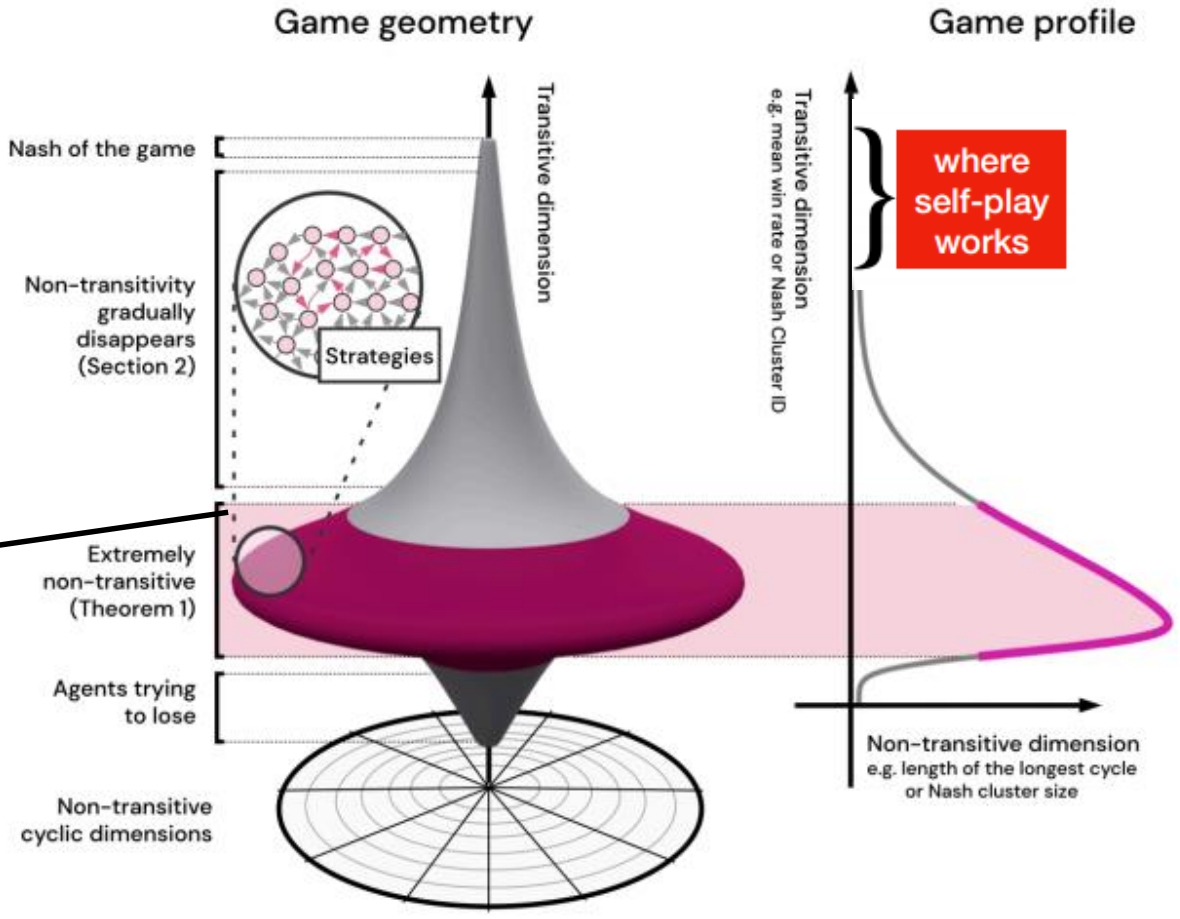


Figure 1: High-level visualisation of the geometry of Games of Skill. It shows a strong transitive dimension, that is accompanied by the highly cyclic dimensions, which gradually diminishes as skil grows towards the Nash Equilibrium (upward), and diminishes as skill evolves towards the wors possible strategies (downward). The simplest example of non-transitive behaviour is a cycle of length 3 that one finds e.g. in the Rock Paper Scissors game.

AlphaStar

自我博弈对于具有较强non-transitive特性的游戏：策略循环

虚拟自我博弈Fictitious Self-Play:

训练过程中，每隔一段时间给自己存档，得到一个种群。均匀随机地从种群中选出对手与正在训练的智能体对战。

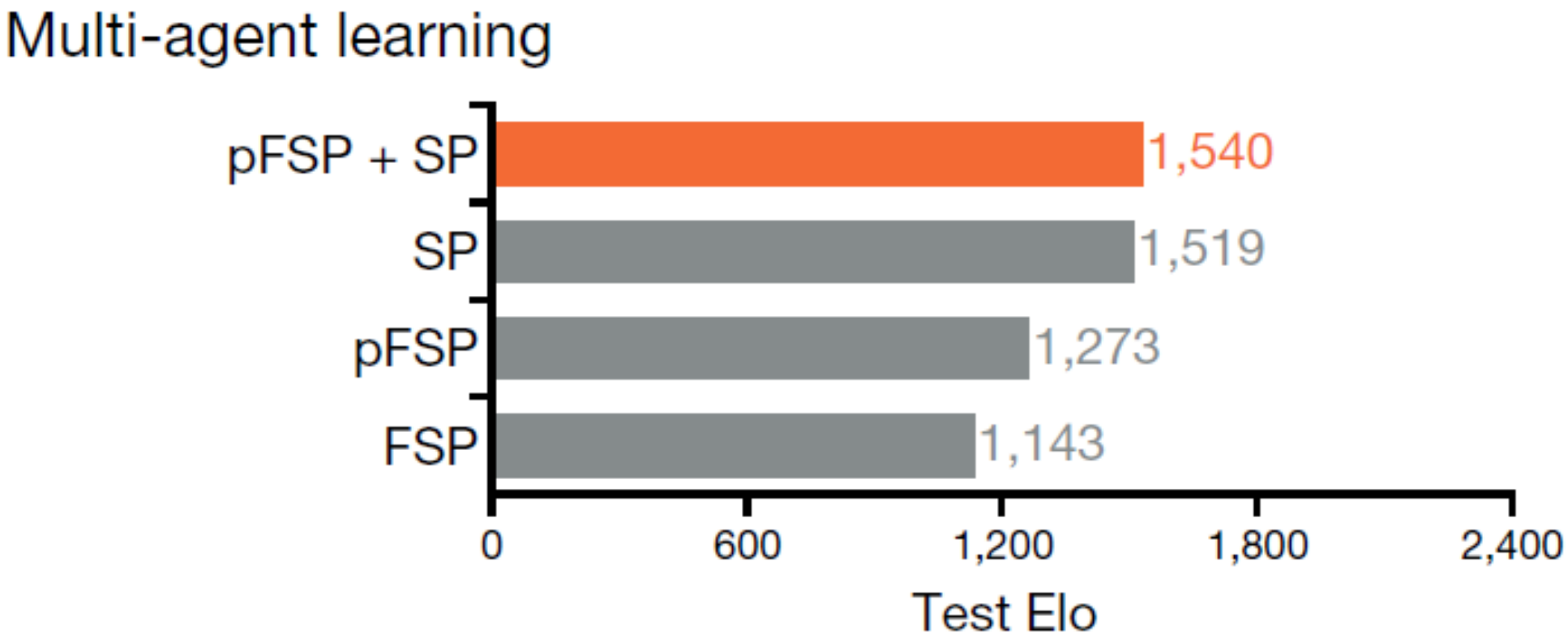
问题：智能体与能力太弱的对手对局，无法提升浪费时间

优先级虚拟自我博弈Prioritized Fictitious Self-Play:

$$\frac{f(\mathbb{P}[A \text{ beats } B])}{\sum_{C \in \mathcal{C}} f(\mathbb{P}[A \text{ beats } C])}$$

按照胜率确定从种群中挑对手的概率，常打败智能体的对手(克制当前策略)概率高

AlphaStar


Multi-agent learning
pFSP + SP
SP
pFSP
FSP
1,540
1,519
1,273
1,143
0
600
1,200
1,800
2,400
Test Elo

AlphaStar

AlphaStar称“对手池”为联盟 (league)，并将其智能体分为以下三类：

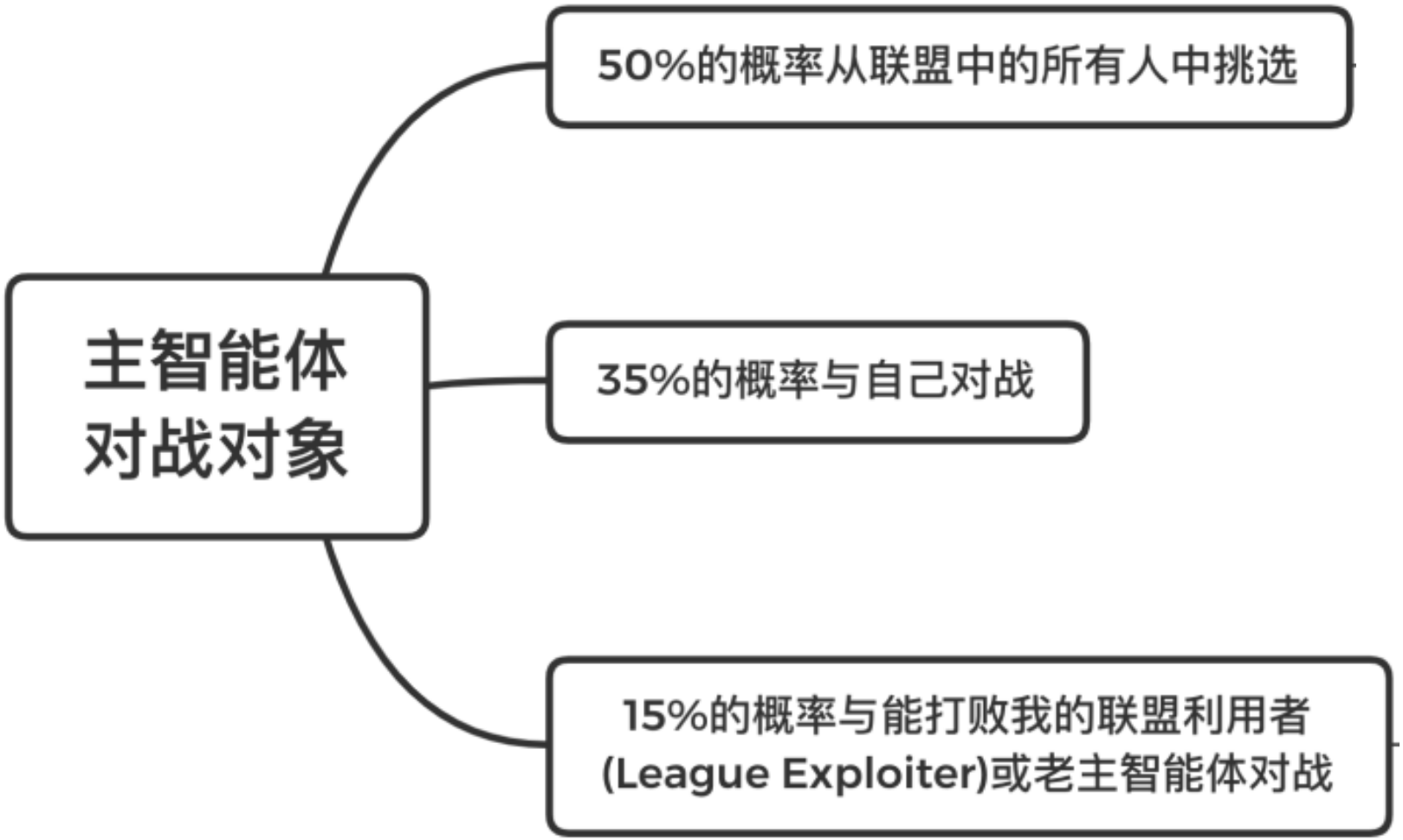
主智能体 (Main Agent)：正在训练的智能体及其祖先，重点培养对象

联盟利用者 (League Exploiter)：能打败联盟里的所有智能体，它发现了全局盲点

主利用者 (Main Exploiter)：能打败正在训练的主智能体，它发现了他们的弱点

AlphaStar

三类智能体如何选取训练过程中对战的对象？

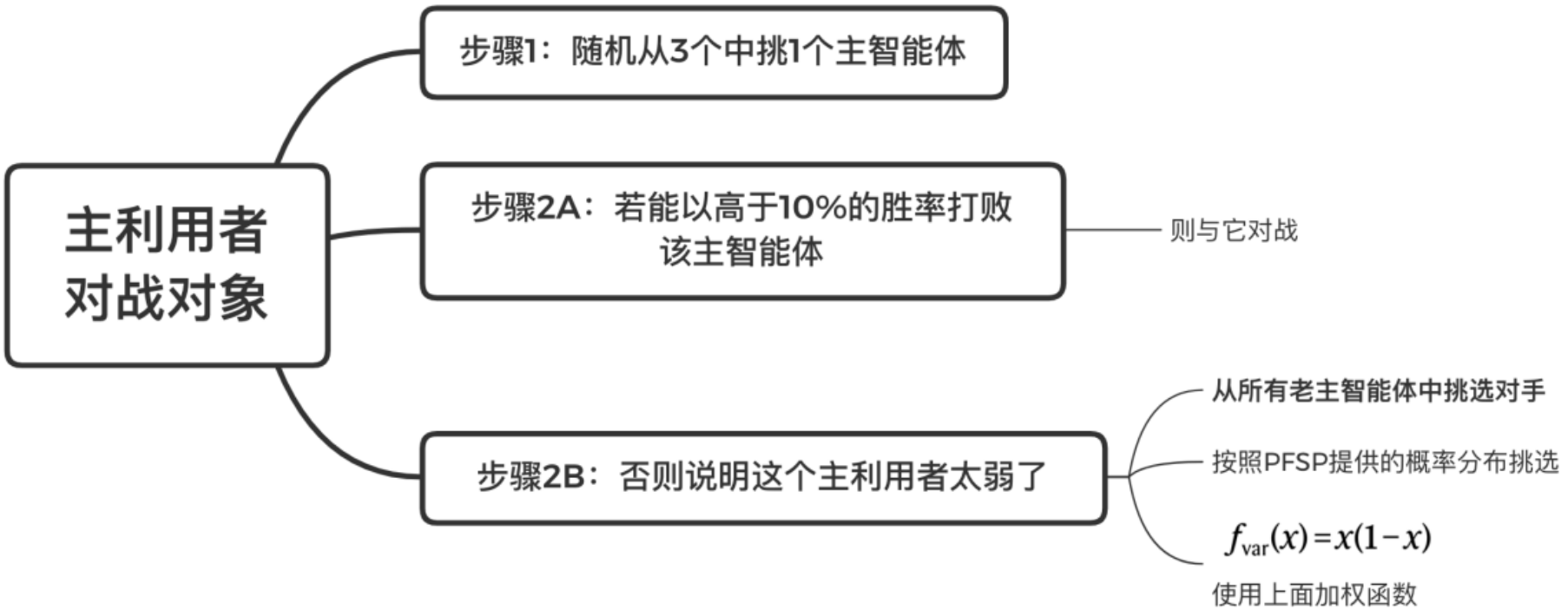


AlphaStar

三类智能体如何选取训练过程中对战的对象？

联盟利用者对战对象：按照PFSP给的概率与全联盟的对手对战

主利用者对战对象：



AlphaStar

三类智能体在什么情况下存档当前策略？

主智能体：每隔2×10^9个时间步之后就存

联盟利用者：以70%胜率打败联盟中的所有人，
或者距上次存档2×10^9个时间步之后就存

主利用者：以70%的胜率打败全部三个正在学习的策略
主智能体，或者距上次存档2×10^9个时间步之后就存

AlphaStar

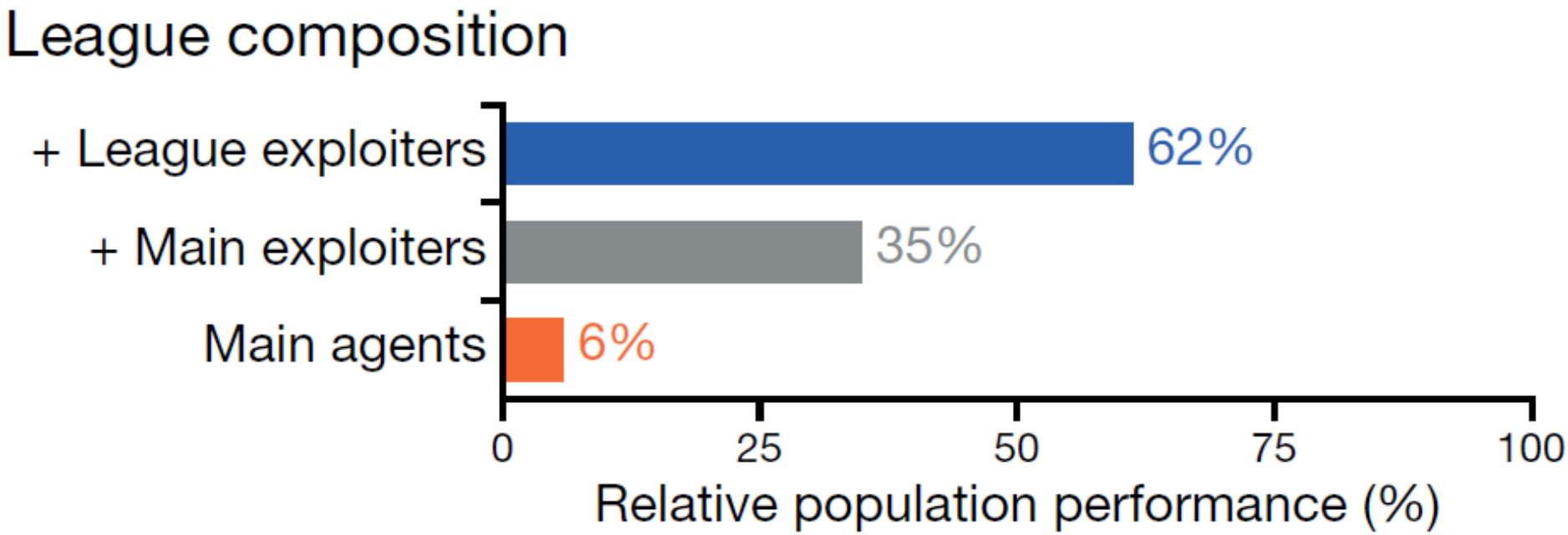
三类智能体多大的概率将策略的参数重设为监督训练给出的初始化？

主智能体：永不

联盟利用者：在存档的时候，有25%概率把场上的联盟利用者的策略重设成监督学习给出的初始化

主利用者：每次存档之后就重设初始化

AlphaStar


League composition
+ League exploiters
62%
+ Main exploiters
35%
Main agents
6%
0
25
50
75
100
Relative population performance (%)

总结

多智能体学习

1. 矩阵博弈
   - 线性规划求解零和博弈均衡策略
   - 二次规划求解一般和博弈均衡策略
2. 马尔可夫博弈
   - Minimax-Q learning/Nash-Q learning/JAL
   - COMA/QMIX/MADDPG
3. MARL的应用
   - AlphaGo系列：MCTS+NN
   - AlphaStar系列：联盟训练

Self-play

感谢大家
的聆听！