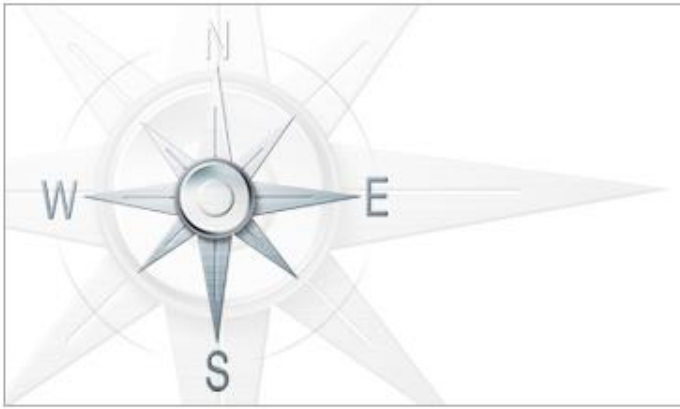


MCG LEARNING GROUP

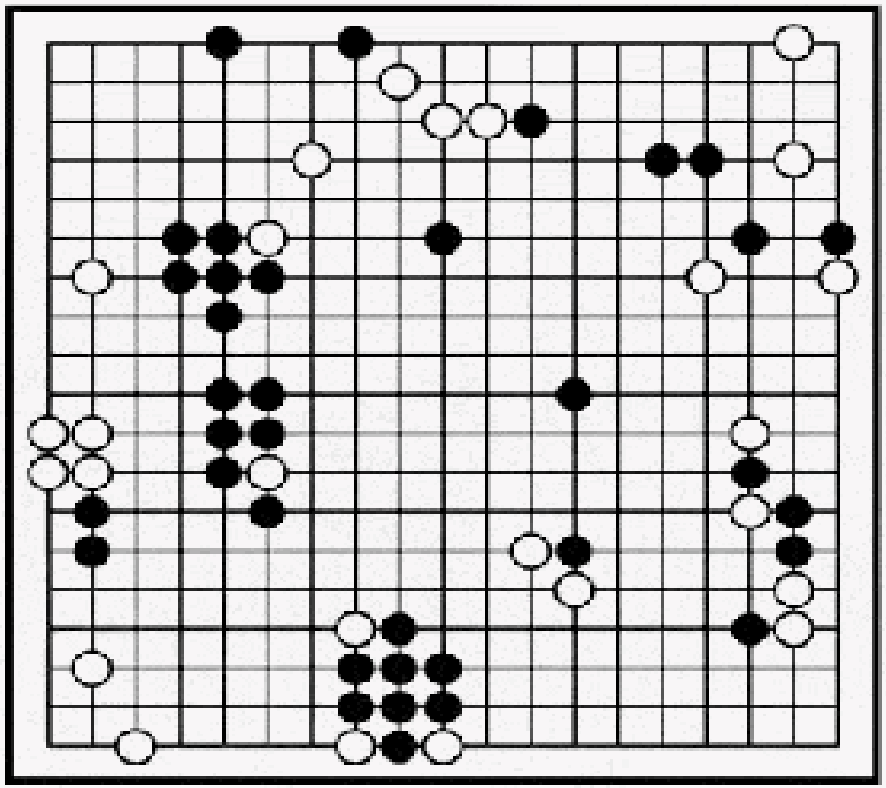
AlphaGo初探

SkyChen

中科院计算所

dzhchxk@126.com

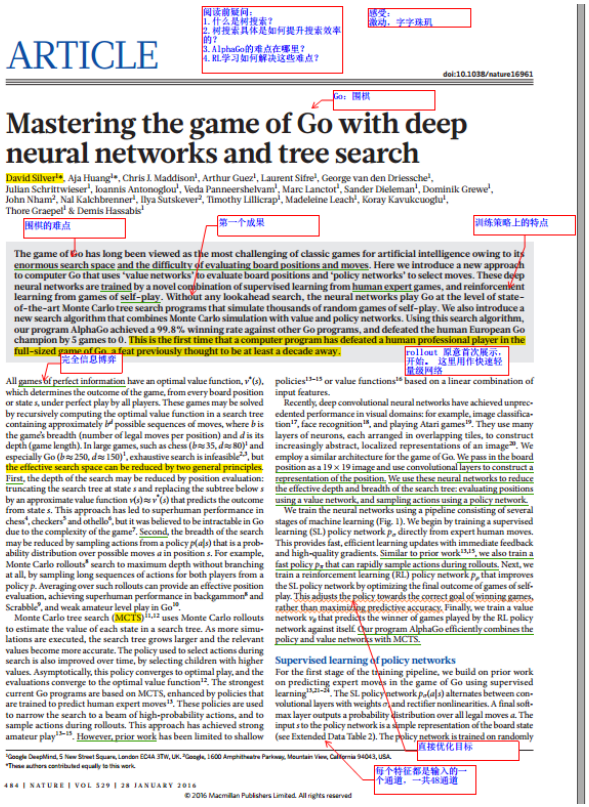
Web site : coderskychen.cn



概要介绍

- 2016年**AlphaGo**计算机围棋系统战胜顶尖职业棋手李世石，引起全世界的广泛关注。

- AlphaGO不难，但是涉及知识点很多：DL、RL、MCTS...



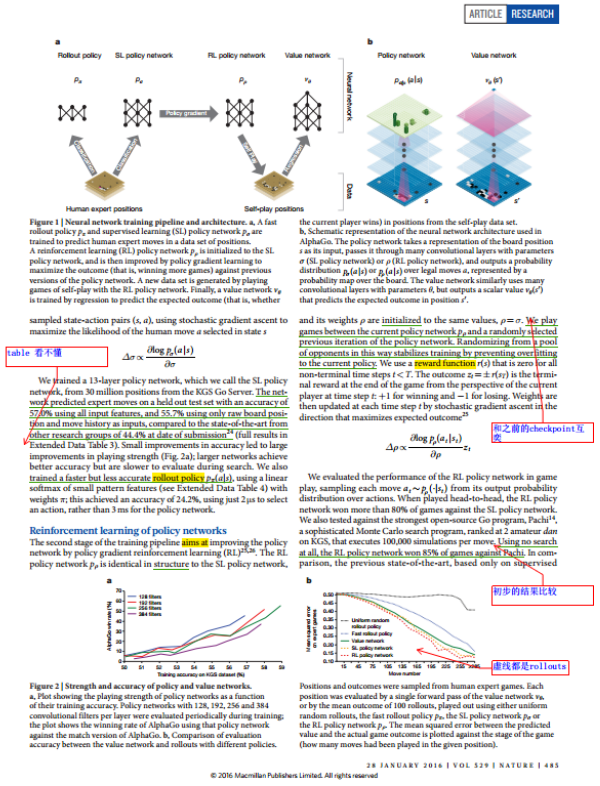
ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.



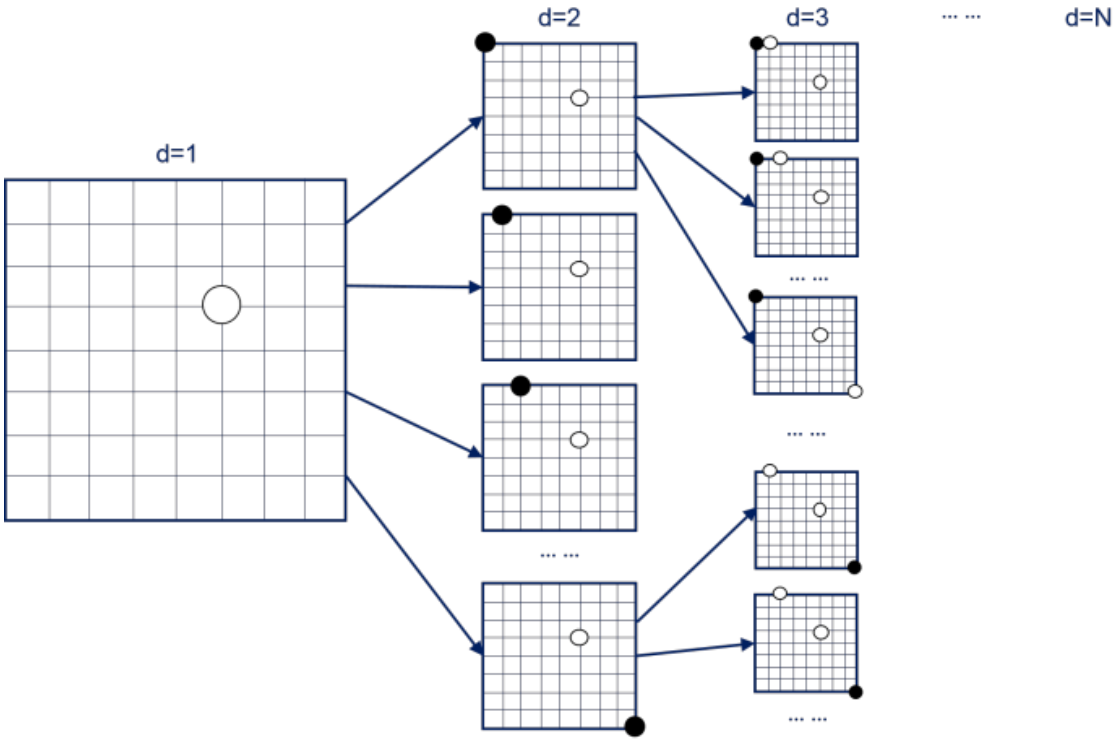
ARTICLE RESEARCH

Figure 1 | Neural network training pipeline and architecture. a, A fast rollout policy p_π and supervised learning (SL) policy network p_σ are trained to predict human expert moves in a data set of positions. A reinforcement learning (RL) policy network p_ρ is initialized to the SL policy network, and is then improved by policy gradient learning to maximize the outcome (that is, winning more games) against previous versions of the policy network. A new data set is generated by playing games of self-play with the RL policy network. Finally, a value network v_θ is trained by regression to predict the expected outcome (that is, whether the current player wins) in positions from the self-play data set. b, Schematic representation of the neural network architecture used in AlphaGo. The policy network takes a representation of the board position s as its input, passes it through many convolutional layers with parameters σ (SL policy network) or ρ (RL policy network), and outputs a probability distribution $p_\sigma(a|s)$ or $p_\rho(a|s)$ over legal moves a, represented by a probability map over the board. The value network similarly uses many convolutional layers with parameters θ, but outputs a scalar value $v_\theta(s')$ that predicts the expected outcome in position s'.

Figure 2 | Strength and accuracy of policy and value networks. a, Plot showing the playing strength of policy networks as a function of their training accuracy. Policy networks with 128, 192, 256 and 384 convolutional filters per layer were evaluated periodically during training; the plot shows the winning rate of AlphaGo using that policy network against the match version of AlphaGo. b, Comparison of evaluation accuracy between the value network and rollouts with different policies. Positions and outcomes were sampled from human expert games. Each position was evaluated by a single forward pass of the value network v_θ, or by the mean outcome of 100 rollouts, played out using either uniform random rollouts, the fast rollout policy p_π, the SL policy network p_σ or the RL policy network p_ρ. The mean squared error between the predicted value and the actual game outcome is plotted against the stage of the game (how many moves had been played in the given position).

从挑战开始讲起

- 搜索空间巨大
  - 棋类游戏可能的局面约有b^d 围棋b≈250合法的下棋位置数，d≈150下棋步数
  - 暴力搜索不可行：超过了目前可观测宇宙中所有原子总数
- 评估局面很困难



解决方法

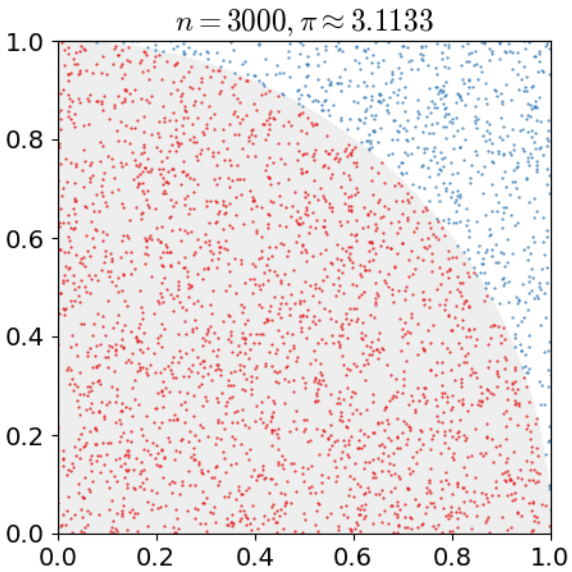
- 引用文中的原话：

"AlphaGo combines the **policy** and **value** networks with **MCTS**"

- MCTS蒙特卡洛树搜索+剪枝
  - 减小搜索的**深度**：使用value network评估当前局面，不用推演到终局
  - 减小搜索的**广度**：使用policy network对合法的动作采样

相关技术

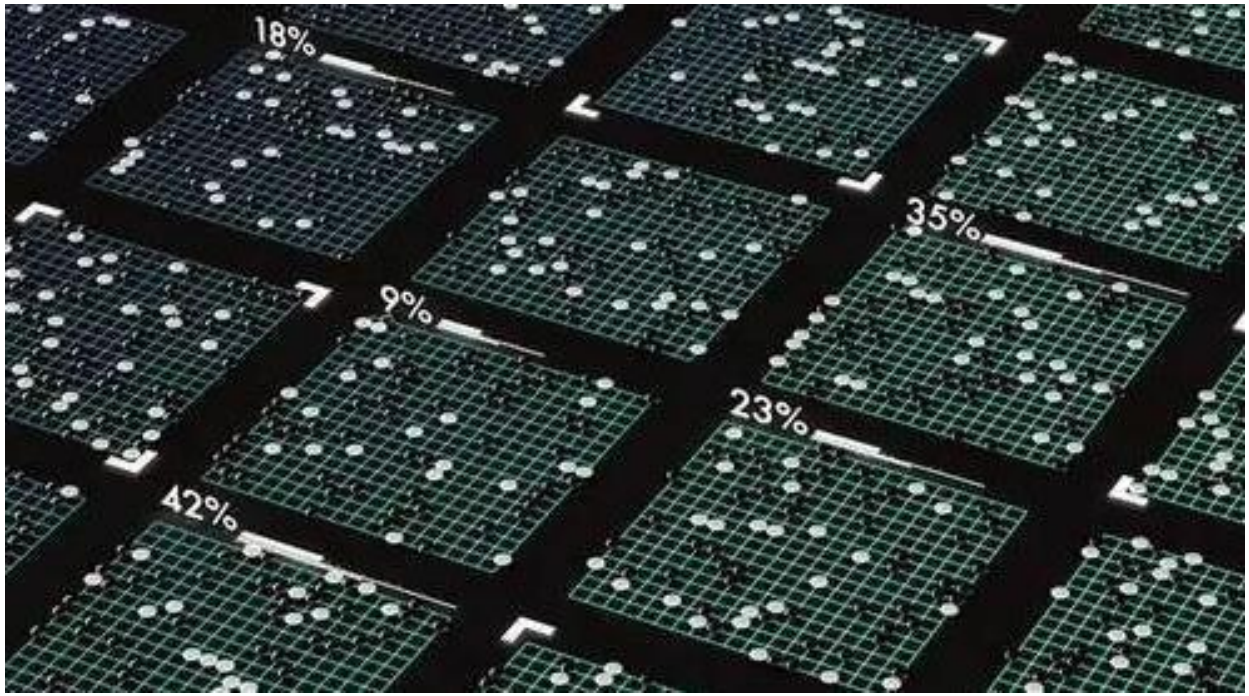
- MCTS：蒙特卡洛树搜索
  - 蒙特卡洛方法：又称统计试验方法，使用计算机统计模拟，以获得问题的近似解。



示例：估算圆周率π

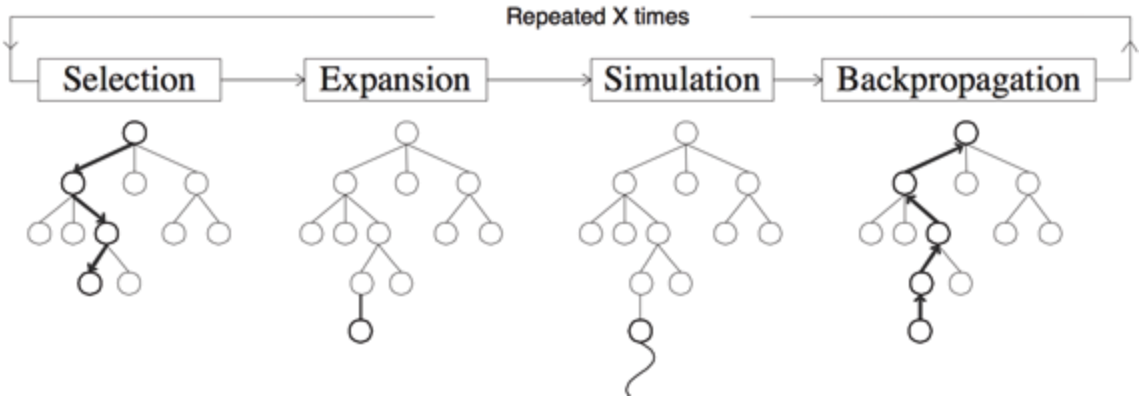
相关技术

- MCTS：蒙特卡洛树搜索
  - 树搜索：一种具有树形结构的搜索方法，通常包括根结点、中间节点和叶子节点
  - 若能暴力穷举，就能获得*必胜策略*



相关技术

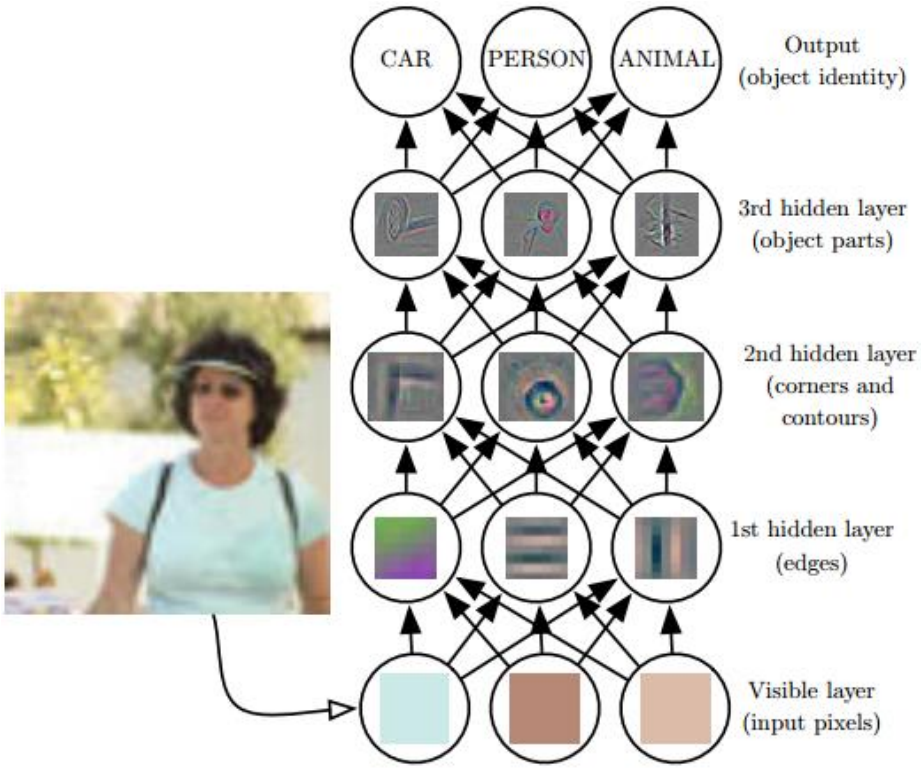
- MCTS：蒙特卡洛树搜索
  - 适合搜索空间巨大的问题，不能计算得到所有子树价值→通过蒙特卡洛方法模拟



1. 选择：从根节点开始，递归选择最优子节点，直到叶子节点L
2. 扩展：若L是非终止节点，那么创建更多子节点
3. 模拟：从其中一个子节点模拟输出，直到终局
4. 信息反向传播：用模拟的结果更新所有相关路径节点信息

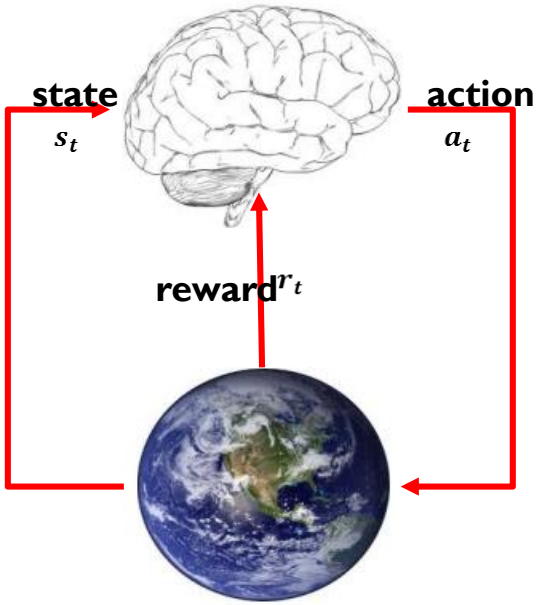
相关技术

- 深度强化学习
  - 深度学习CNNs：基于多层次组合的思想，一类有效的表示学习方法



相关技术

- 深度强化学习
  - 强化学习：与监督学习和无监督学习并驾齐驱
    - 它关注如何使智能体在与环境交互的马尔科夫决策过程中获得最大的**累积收益**。
  - 五要素
    - S，状态集
    - A，动作集
    - P，状态转移概率
    - R，回报函数
    - γ，折扣因子



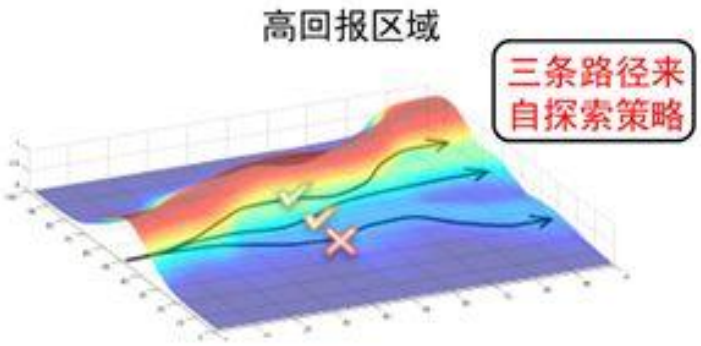
相关技术

- 深度强化学习
  - 强化学习在围棋中的形式化描述：
    - $s_t \in S$，时刻t的棋盘局面
    - $a_t \in A$，时刻t的下法
    - $p(s_{t+1}|s_t,a_t)$，是确定性的，做完下棋动作后局面会从s_t确定性地转移到s_{t+1}
    - r_t，只在终局有回报，±1
    - γ，由于回报只在终局产生，不需要考虑折扣因子

相关技术

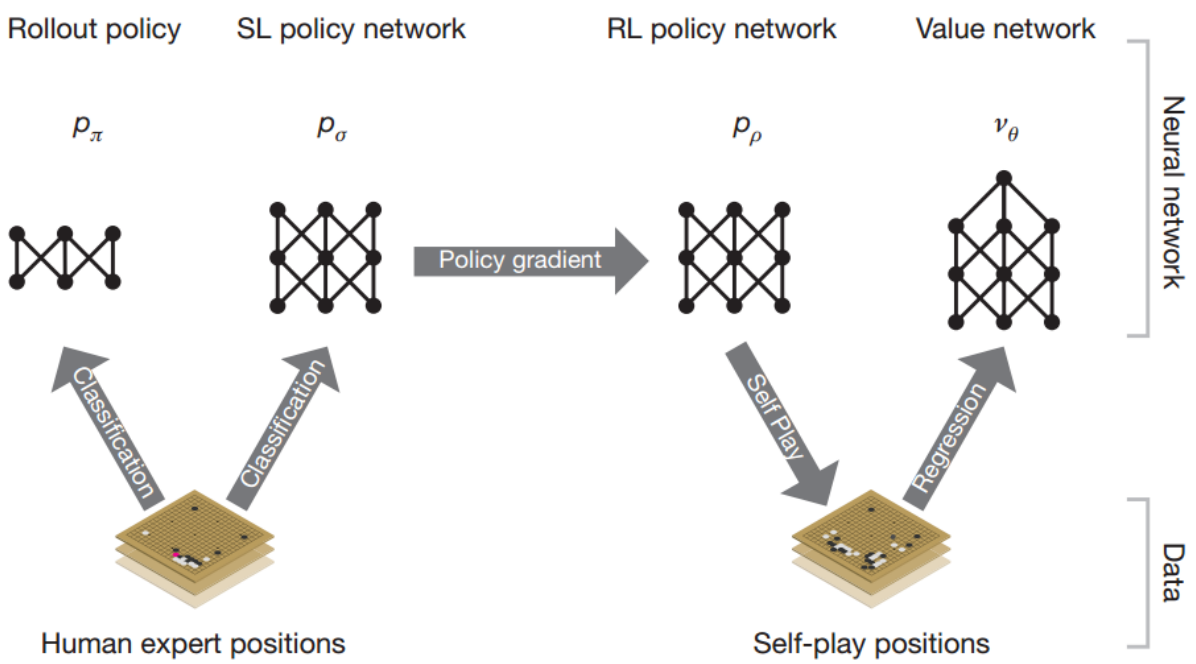
- 基于策略函数的强化学习
  - 该类方法在策略空间直接搜索来得到最佳策略

  - 用τ表示一组状态-行为序列：$\tau = s_0, a_0, \ldots, s_T, a_T$
  - 用$R(\tau)$表示轨迹τ的回报，P$(\tau;\theta)$表示此轨迹出现的概率
  - 目标函数：$J(\theta) = E_{\tau \sim p(\theta)} R(\tau)$，最大化回报的期望
  - 使用梯度上升优化参数θ使得目标函数J(θ)最大
  - $\frac{\partial J(\theta)}{\partial \theta} = E_{\tau \sim p(\theta)} \{ \sum_t \frac{\partial \log p_\theta(a_t|s_t)}{\partial \theta} * R(\tau) \}$



策略梯度增加高回报路径出现的概率

AlphaGo的训练流程



- 包括四个网络
- 这四个网络与MCTS有效结合完成任务

神经网络的训练

- 经过三个训练阶段
  - 第一阶段：**模仿高手**
    1. SL policy network
    2. Rollout policy network，轻量级
  - 第二阶段：强化学习提升棋力
    3. RL policy network，获得更好的策略
  - 第三阶段：学习评估局面函数
    4. Value network

第一阶段：模仿高手

- 监督学习范式，从人类棋谱中随机采样训练数据对: (局面s,下棋动作a)
- 15万职业棋手棋谱+百万业余棋手棋谱

  - 输入：某一时刻棋盘局面s
  - 输出：预测专家下棋走法
  - 目标：极大似然p(a|s)，拟合人类下法a

  - 是一个*分类任务*

第一阶段：模仿高手

- 网络输入
  - 19*19*48的图像
  - 48为通道数量，不同角度对局面的描述

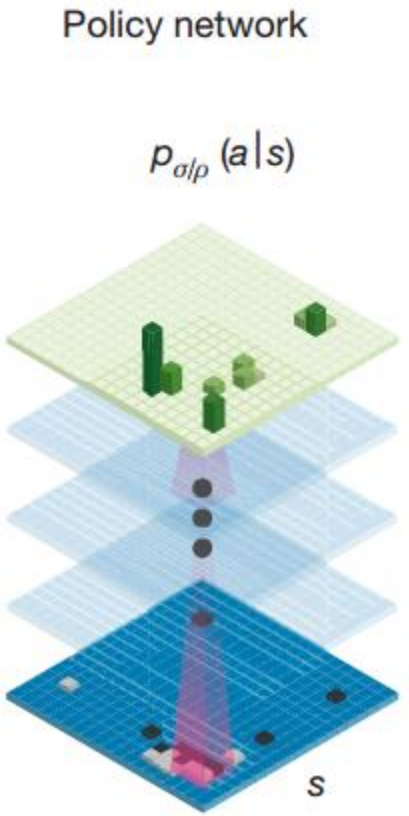
Extended Data Table 2 | Input features for neural networks

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

Feature planes used by the policy network (all but last feature) and value network (all features).

第一阶段：学习人类棋谱

- 网络结构
  - 全卷积神经网络，13层，没有pooling层
  - 最后一层softmax输出各位置的下棋概率



第一阶段：学习人类棋谱

- 得到两个网络：
  - **Rollout** policy network: $p_\pi(a|s)$
    - 轻量级，测试集准确率24.2%，耗时2微妙
    - **快速的**蒙特卡洛模拟
  - **SL** policy network: $p_\sigma(a|s)$
    - 测试集准确率57.0%，耗时3毫秒
    - 高质量的先验概率P
  - 两个网络的区别仅在于参数量，卷积核个数不同
  - 因为它们有不同的用途

第二阶段：强化学习提升棋力

- 基于策略梯度的强化学习：REINFORCE 算法

- 状态：$s_t \in S$，时刻t的棋盘局面，19*19*48
- 动作：$a_t \in A$，时刻t的下法
- 回报：$z_t = r(s_T)$，只在终局有回报，±1

➢1）将基于**监督学习**得到的网络作为强化学习的初始网络；
➢2）将当前版本的网络和之前某个随机的版本**对局**，得到棋局和输赢结果；
➢3）根据棋局和结果基于**强化学习**的REINFORCE算法**更新**网络参数，最大化期望结果；
➢4）每500次迭代就**复制**当前网络参数到模型池中，用于2）。

➢强化学习**本质**上通过不断的**试错**进行学习，在试错的过程中与环境交互，获取大量经验，策略梯度则是以梯度的方式提升高回报经验出现的概率！

- ➢步骤 2）即为试错的过程，有输有赢
- ➢步骤 3）即为提升的过程，增强获胜走法出现的概率

第二阶段：强化学习进一步提升棋力

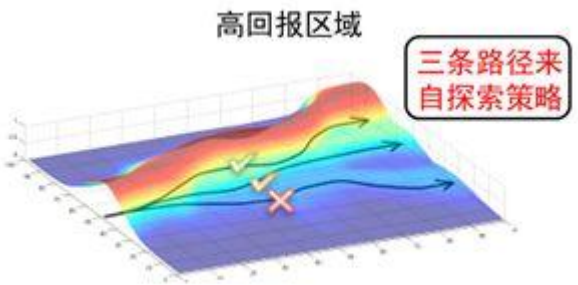
- 强化学习：左右互搏，不断**进化**

模型 v1.0	VS	模型 v1.1	胜，负？
模型 v1.1	VS	模型 v1.5	胜，负？
……		……	……
模型 v1000	VS	模型 v1099	胜，负？

回顾：策略梯度

- 基于策略函数的强化学习
  - 该类方法在策略空间直接搜索来得到最佳策略
  - 使用梯度上升优化参数θ使得目标函数J(θ)最大

  - 用τ表示一组状态-行为序列：$\tau = s_0, a_0, \ldots, s_T, a_T$
  - 用$R(\tau)$表示轨迹τ的回报，P(τ;θ)表示此轨迹出现的概率
  - 目标函数：$J(\theta) = E_{\tau \sim p(\theta)} R(\tau)$
  - $$\frac{\partial J(\theta)}{\partial \theta} = E_{\tau \sim p(\theta)} \{ \sum_t \frac{\partial \log p_\theta(a_t|s_t)}{\partial \theta} * R(\tau) \}$$



策略梯度增加高回报路径出现的概率

第二阶段：强化学习进一步提升棋力

- REINFORCE 算法核心：
  - $\{s_t, a_t\}_{t=1}^{T}$ 称为一条轨迹或路径
  - REINFORCE算法使用**梯度上升**增加高回报路径出现的概率

$$\Delta\rho = \sum_{t=1}^{T} \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} z_t \xrightarrow[\text{降低方差}]{\text{减去baseline}} \Delta\rho = \sum_{t=1}^{T} \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} (z_t - v(s_t))$$

对于AlphaGo而言，在一盘棋中，如果这盘棋赢了，那么这盘棋下的每一步都是认为是好的，如果输了，那么都认为是不好的。好的z_t就是1，不好的就-1。所以在这里，如果a被认为是好的，那么目标就是最大化这个好的动作的概率，反之亦然。这就是Policy Gradient最基本的思想。

第二阶段：强化学习进一步提升棋力

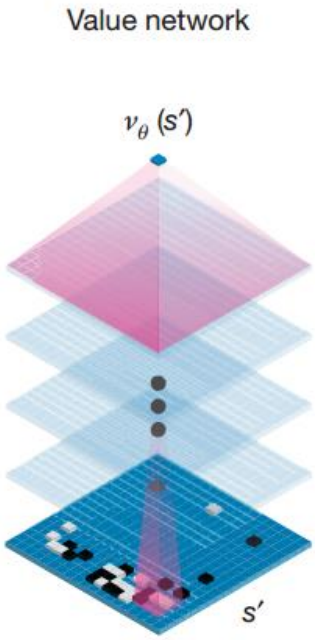
- 效果：该阶段就达到了state-of-the-art
  - 强化学习得到的走子网络RL policy network：
    - ✓ 在80%的对局中战胜了SL policy network
    - ✓ 与当时最强开源软件Pachi相比：在10w盘测试中，赢得了85%的对局
    - ◆ 注意：此时RL policy network没有进行任何搜索，而Pachi应用了非常复杂的蒙特卡洛搜索算法
    - ◆ 还不足以击败人类职业选手

第三阶段：学习局面评估函数

- 局面评估函数：评估在某策略下局面的好坏
  - 输入：局面s， 在RL policy network下产生的
  - 输出：对最终结果的预测，-1~1
  - 网络结构：与策略网络相似，最后一个输出神经元

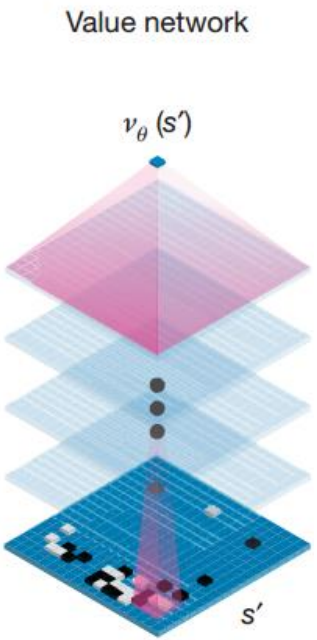
- 训练方法：**回归任务**，以MSE损失训练网络
  - 网络输出$v_\theta(s)$拟合最终结果z
  - 梯度下降：

$$\Delta\theta = \frac{\alpha}{m}\sum_{k=1}^{m}\left(z^k - v_\theta(s^k)\right)\frac{\partial v_\theta(s^k)}{\partial\theta}$$

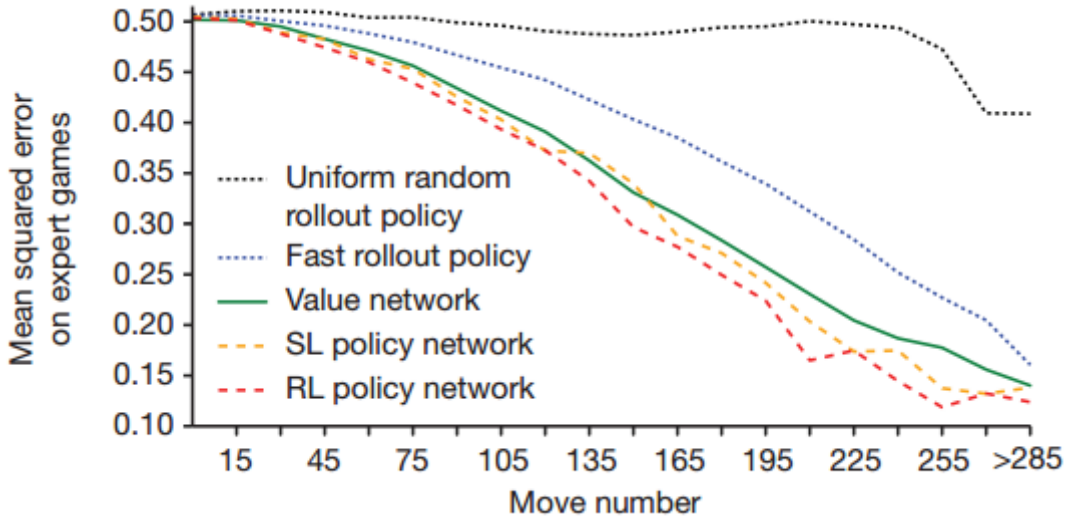


第三阶段：学习局面评估函数

- 存在挑战：过拟合
  - 仅一步棋的差异，就可能导致最终输出发生巨大变化
  - 数据规模小：仅使用人类棋谱训练：
    - 训练集MSE：0.19
    - 测试集MSE：0.37
- 解决方法：self-play增广数据
  - 使用RL policy network自我对弈
    - 增加了3000w不同的局面，每一个局面都来自于一个新的对弈，去相关性
    - 训练集MSE：0.226
    - 测试集MSE：0.234



第三阶段：学习局面评估函数

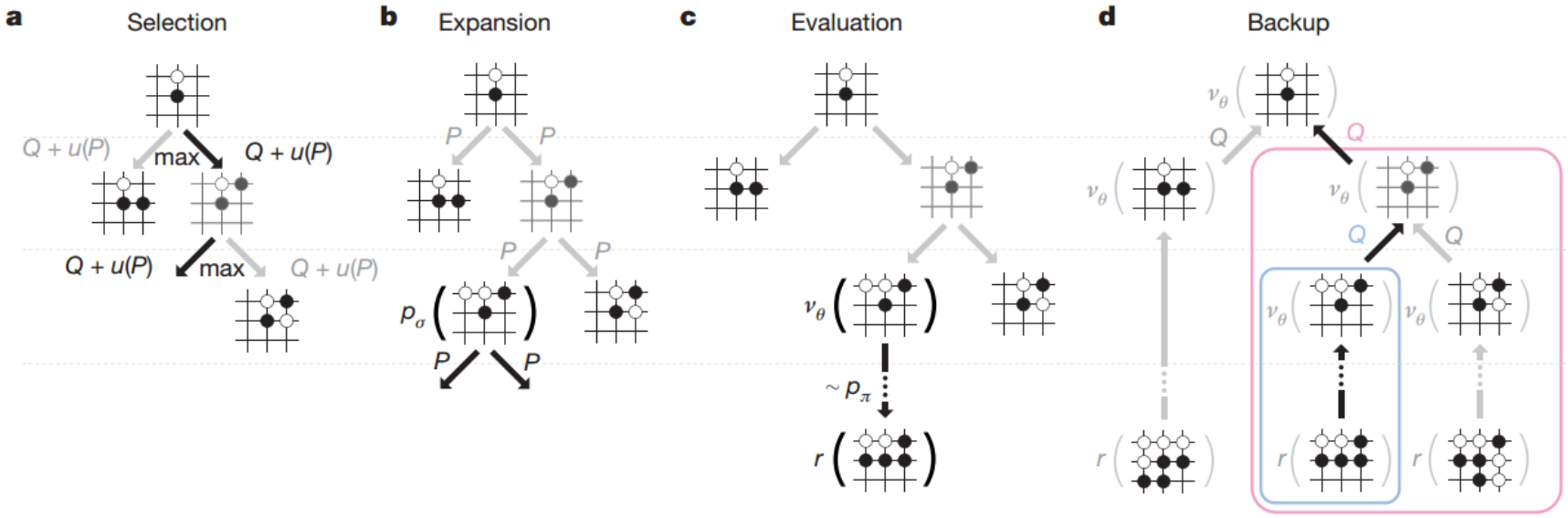


- 纵坐标：MSE
- 横坐标：局面上已经走完的步数
- 虚线：不同策略下的蒙特卡洛模拟的结果，模拟100次，求平均
- 实线：value network的结果，仅仅需要**1次**前馈计算

- Value network（实线）的结果与RL policy（红色虚线）下的模拟结果非常接近，说明value network的效果很好有效的，因为RL policy的模拟结果是上限

蒙特卡洛树搜索与神经网络的结合

- 下棋时，从当前局面开始模拟n次，模拟过程中通过四步走策略完成**前向搜索,** 选择在模拟中被选择次数最多的走法

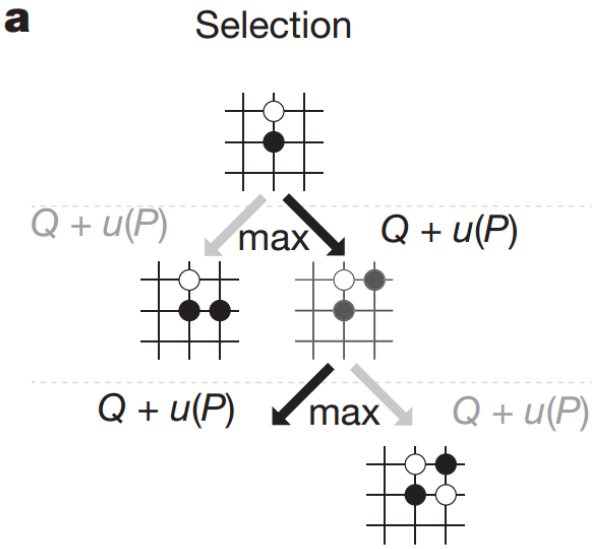


蒙特卡洛树搜索与神经网络的结合

- 第一步a：选择
- 在已经历过的局面中选择最优局面，局面向前推进
  - 从根节点（当前局面）开始，递归的根据如下公式选择动作
  - 每一个动作a_t产生一个分支，即(s, a)表示一条边

$$a_t = \underset{a}{\operatorname{argmax}}(Q(s_t,a) + u(s_t,a))$$

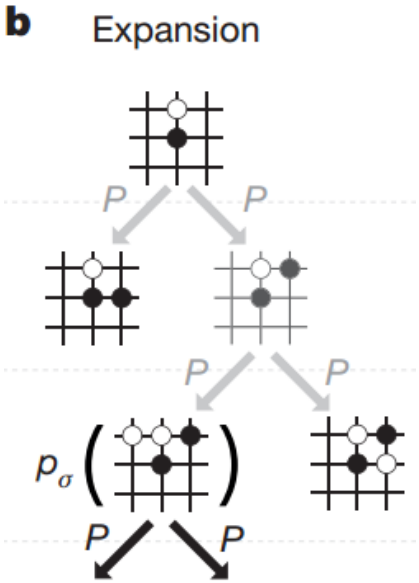
$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$



  - Q为动作价值、N为访问计数，P为先验概率

蒙特卡洛树搜索与神经网络的结合

- 第二步b：扩展
- 走到了从未经历过的局面时，探索新行为，生成新的子树
  - 进行到叶子节点时，触发此步骤
  - 此时，使用走子网络计算所有合法的动作的概率，被存储为先验概率P
  - 实验发现，在AlphaGo中走子网络选择SL policy比RL policy结果好
    - 作者**推测**，人类棋谱中走法具有多样性

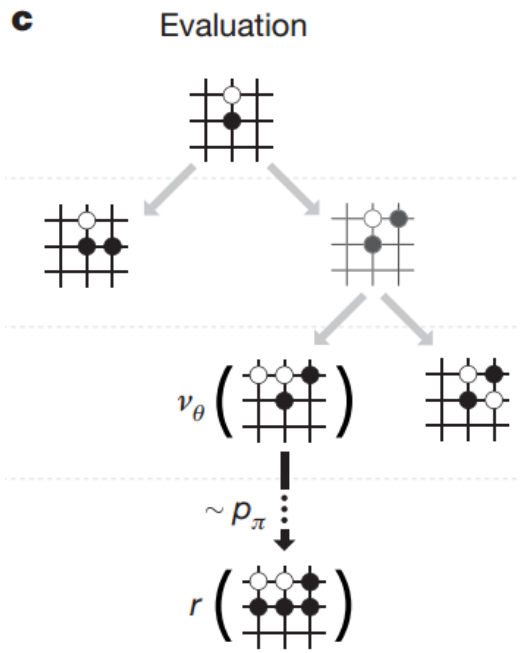


蒙特卡洛树搜索与神经网络的结合

- 第三步c：局面评估
- 评估一系列行为的回报
  - 在模拟结束时，对叶子节点进行评估

① $v_\theta(s_L)$：使用value network直接估计叶子局面
② z_L：使用轻量级策略：rollout policy network迅速推演到终局，获得胜负结果

$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L$$



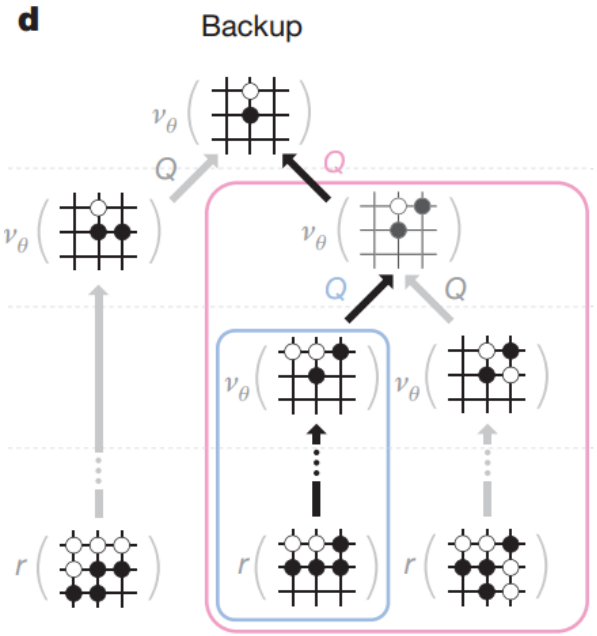
蒙特卡洛树搜索与神经网络的结合

- 第四步d：回溯更新
- 更新路径上所有结点的信息
  - 更新根节点下所有节点的访问次数N、动作价值Q

$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$$

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$$

  - n为模拟次数



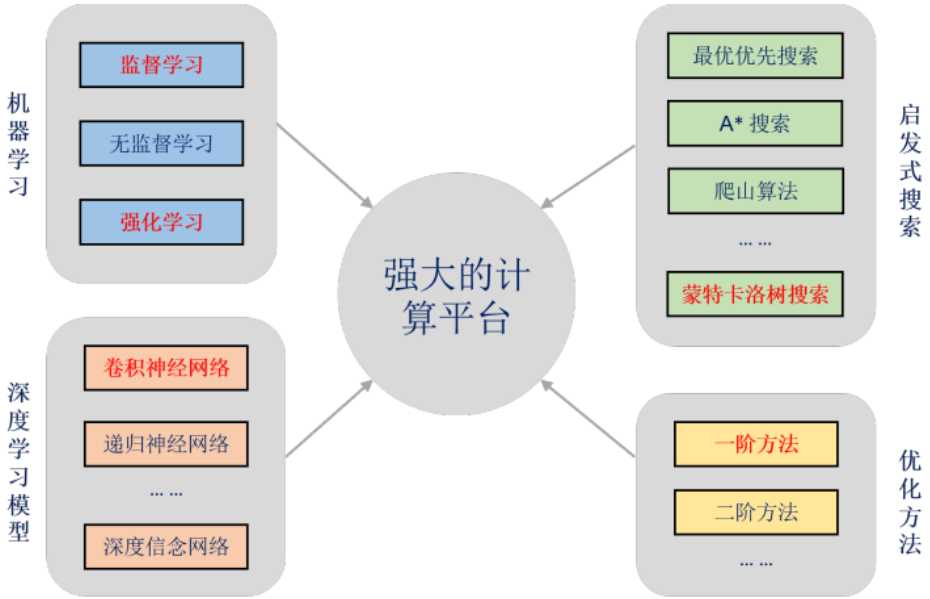
蒙特卡洛树搜索与神经网络的结合

- 下棋时，从当前局面开始，在比赛时长允许的情况下，根据四步走的策略，建立蒙特卡洛搜索树，结束时选择被访问次数最多的动作
  ① Maximum visit count，作者认为该方式**更鲁棒**
  ② Maximum action value
- 这颗树在一定程度可以**复用**
  - 例如：如果当前局面在上一步出现过，那么就可以复用之前建立搜索树的部分子树，其他部分可以丢弃

小结

- 一句话概括AlphaGo：MCTS与深度神经网络的有效结合
- 强化学习在AlphaGo中的作用：
  - 更强走子策略：相比于SL以预测专家下棋为目标，RL以赢棋为目标
  - 好的走子策略→高质量的value network→更准确的局面评估

- AlphaGo是现有技术集大成的产物：
  - 监督学习、强化学习、MCTS、CNN、一阶优化、分布式训练1202个CPU，176个GPU。。。



感谢各位聆听

Thanks for Listening

SkyChen
中科院计算所
dzhchxk@126.com
Web site : coderskychen.cn