机器学习: Part 6

- 深度强化学习
- AlphaGo

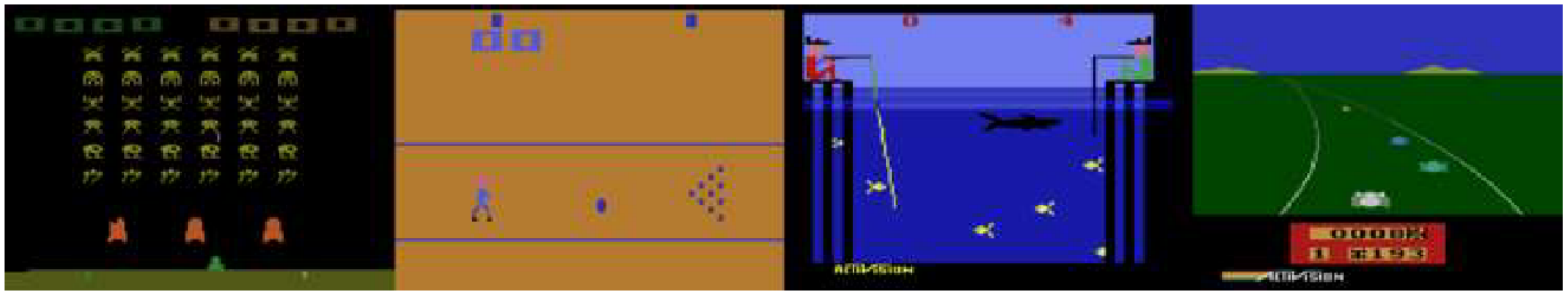
*Slides based on those of Pascal Poupart and Fei Wu

强化学习的挑战

- 要在接近现实世界复杂性的情况下成功运用强化学习，智能体面临着一个困难的任务：
- 它们必须从高维感知输入中得出环境的高效表示。
- 人类和其他动物似乎通过强化学习和分层感知处理系统的和谐组合来解决这个问题。

深度强化学习

- 将强化学习与深度神经网络相结合。
- 使用深度卷积神经网络（CNN）来近似最优动作价值函数$Q^*(s, a)$
- 在Atari 2600 游戏上进行测试。



- 在49个游戏上，达到与专业人类游戏测试者相当的水平。

快速回顾

- $Q^*(s, a)$: 在状态s 下执行动作a，然后遵循最优策略的期望价值。
- $V^*(s)$: 在状态s 下遵循最优策略的期望价值。
- TD formula: $A_k = A_{k-1} + \alpha_k(v_k - A_{k-1})$
  - 我们有一个值序列: $v_1, v_2, v_3, \ldots,$
  - 为了得到新的估计值，通过α_k 乘以TD 误差来更新旧的估计值。
- Q-learning: 经验$\langle s, a, r, s' \rangle$ 给出了对$Q^*(s, a)$ 值的新估计: $r + \gamma \max_{a'} Q[s', a']$, 因此

$$Q[s, a] \leftarrow Q[s, a] + \alpha \left(r + \gamma \max_{a'} Q[s', a'] - Q[s, a] \right)$$

Recall: Q-learning

initialize $Q[S, A]$ arbitrarily
observe current state s
repeat forever:
    select and carry out an action a
    observe reward r and state s'
    $Q[s, a] \leftarrow Q[s, a] + \alpha\left(r + \gamma \max_{a'} Q[s', a'] - Q[s, a]\right)$
    $s \leftarrow s'$

大规模状态空间

- Q-learning的复杂性取决于状态和动作的数量。
- 围棋: 3^{361} 个状态
- Atari: 210x160 像素图像，128种颜色调色板

Q-networks

- 使用权重w的Q-network 表示价值函数
  $Q(s, a, \mathrm{w}) \approx Q^*(s, a)$
- Q-learning:
  $Q[s, a] \leftarrow Q[s, a] + \alpha\left(r + \gamma \max_{a'} Q[s', a'] - Q[s, a]\right)$
- 将$r + \gamma \max_{a'} Q[s', a']$ 视为目标值
- 通过梯度下降最小化平方误差:
  $Loss(\mathrm{w}) = \left(r + \gamma \max_{a'} Q(s', a', \mathrm{w}) - Q(s, a, \mathrm{w})\right)^2$
- $\frac{\partial Loss(\mathrm{w})}{\partial w} = -\left(r + \gamma \max_{a'} Q(s', a', \mathrm{w}) - Q(s, a, \mathrm{w})\right)\frac{\partial Q(s,a,\mathrm{w})}{\partial w}$

梯度Q学习

initialize weights w at random in [-1,1]

observe current state s

repeat forever:

select and carry out an action a

observe reward r and state s'

$$\frac{\partial Loss(\mathrm{w})}{\partial w} = -(r + \gamma \max_{a'} Q(s', a', \mathrm{w}) - Q(s, a, \mathrm{w}))\frac{\partial Q(s,a,\mathrm{w})}{\partial w}$$

$$\mathrm{w} \leftarrow \mathrm{w} - \alpha \frac{\partial Loss(\mathrm{w})}{\partial w}$$

$$s \leftarrow s'$$

线性梯度Q学习的收敛性

- 回顾：Q学习在以下条件下收敛到最优Q函数：

$$\sum_{k=1}^{\infty} \alpha_k = \infty \text{ and } \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

- 线性梯度Q-Learning 在相同的条件下收敛。
  - $Q(s, a, \mathrm{w}) = \Sigma_i w_i s_i$, where $s = (x_1, \ldots, x_n)$

非线性Q学习的发散性

- 即使满足以下条件

$$\sum_{k=1}^{\infty} \alpha_k = \infty \text{ and } \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

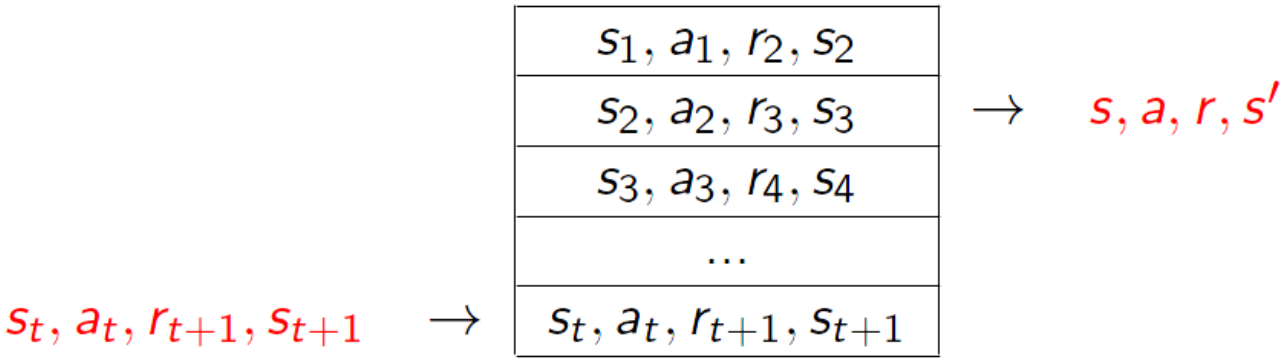
  非线性Q学习可能会发散。

- 直观理解：调整权重w 以增加在(s, a) 处的Q 值可能会引入在附近状态-动作对的错误。

- 缓解发散：通常使用两种技巧
  - 经验回放（Experience replay）
  - 使用两个网络：Q网络和目标网络（Target network）

经验回放

- 思路：将先前的经验(s, a, r, s')存储到缓存中，并在每个步骤中从先前的经验中随机采样一个小批量来进行Q-learning学习。



- 优势
  - 打破连续更新之间的相关性（更加稳定的学习）
  - 较少的与环境的交互次数即可收敛（更高的数据效率）

目标网络

- 思路：使用一个单独的目标网络，定期进行更新

  对于每个小批量中的经验$(\hat{s}, \hat{a}, \hat{r}, \hat{s}')$

  $$\frac{\partial Loss(\mathrm{w})}{\partial w} = -(\hat{r} + \gamma \max_{\hat{a}'} Q(\hat{s}', \hat{a}', \bar{\mathrm{w}}) - Q(\hat{s}, \hat{a}, \mathrm{w}))\frac{\partial Q(\hat{s},\hat{a},\mathrm{w})}{\partial w}$$

  $$\mathrm{w} \leftarrow \mathrm{w} - \alpha \frac{\partial Loss(\mathrm{w})}{\partial w}$$

- 优势：缓解发散

Deep Q-network

- 谷歌Deep Mind
- Deep Q-network: 梯度Q-learning with：
  - 深度神经网络
  - 经验回放
  - 目标网络
- 突破：在许多Atari 视频游戏中达到人类水平。

Deep Q-network

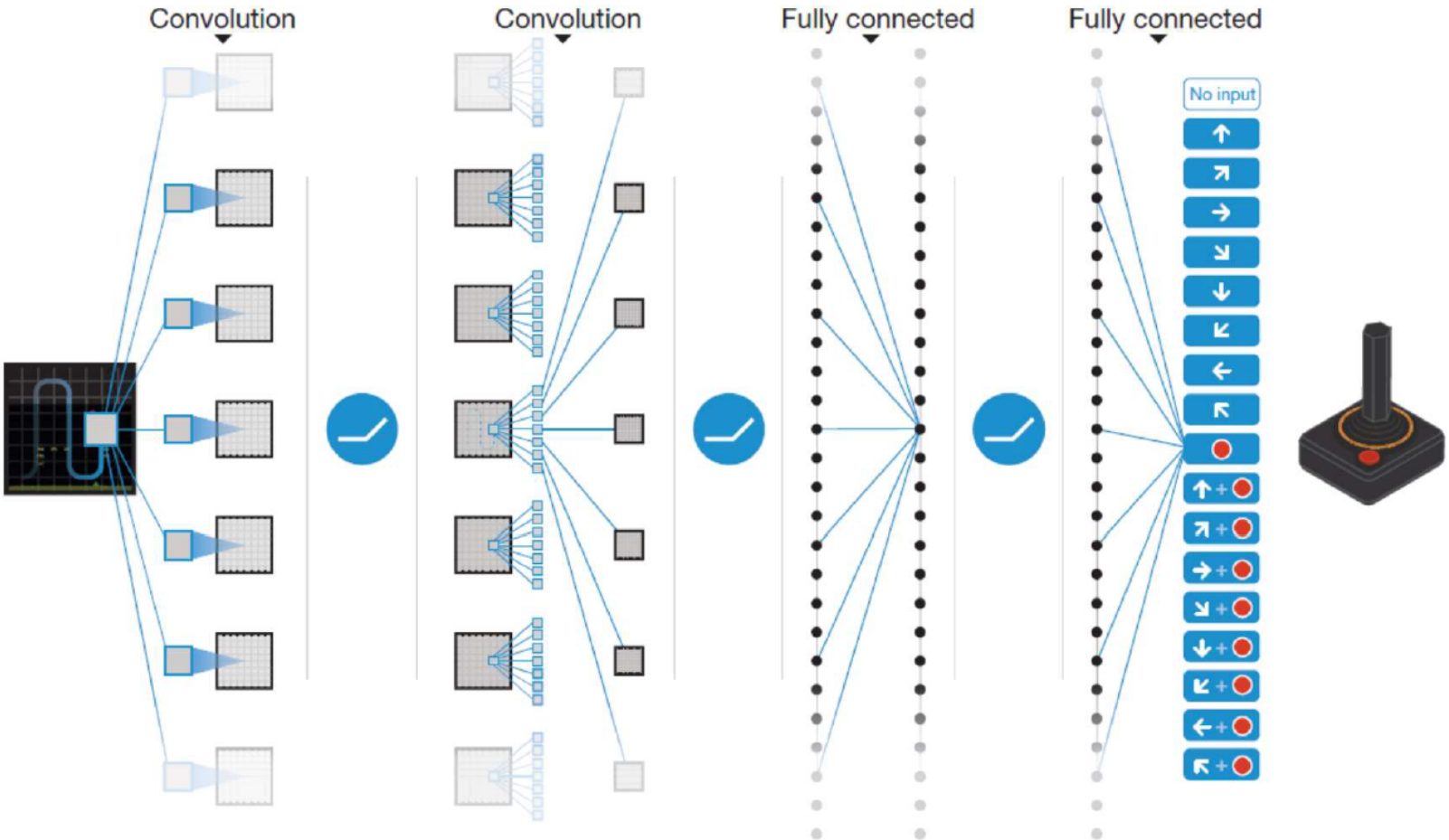
initialize weights w at random in [-1,1]
observe current state s
repeat forever:
select and carry out an action a
observe reward r and state s'
Add (s, a, r, s') to experience buffer
Sample mini-batch of experiences from buffer
For each experience $(\hat{s}, \hat{a}, \hat{r}, \hat{s}')$ in mini-batch

$$\frac{\partial Loss(\mathrm{w})}{\partial w} = -(\hat{r} + \gamma \max_{\hat{a}'} Q(\hat{s}', \hat{a}', \bar{\mathrm{w}}) - Q(\hat{s}, \hat{a}, \mathrm{w}))\frac{\partial Q(\hat{s},\hat{a},\mathrm{w})}{\partial w}$$

$$\mathrm{w} \leftarrow \mathrm{w} - \alpha \frac{\partial Loss(\mathrm{w})}{\partial w}$$

$s \leftarrow s'$
Every c steps, $\bar{\mathrm{w}} \leftarrow \mathrm{w}$

Deep Q-Network for Atari

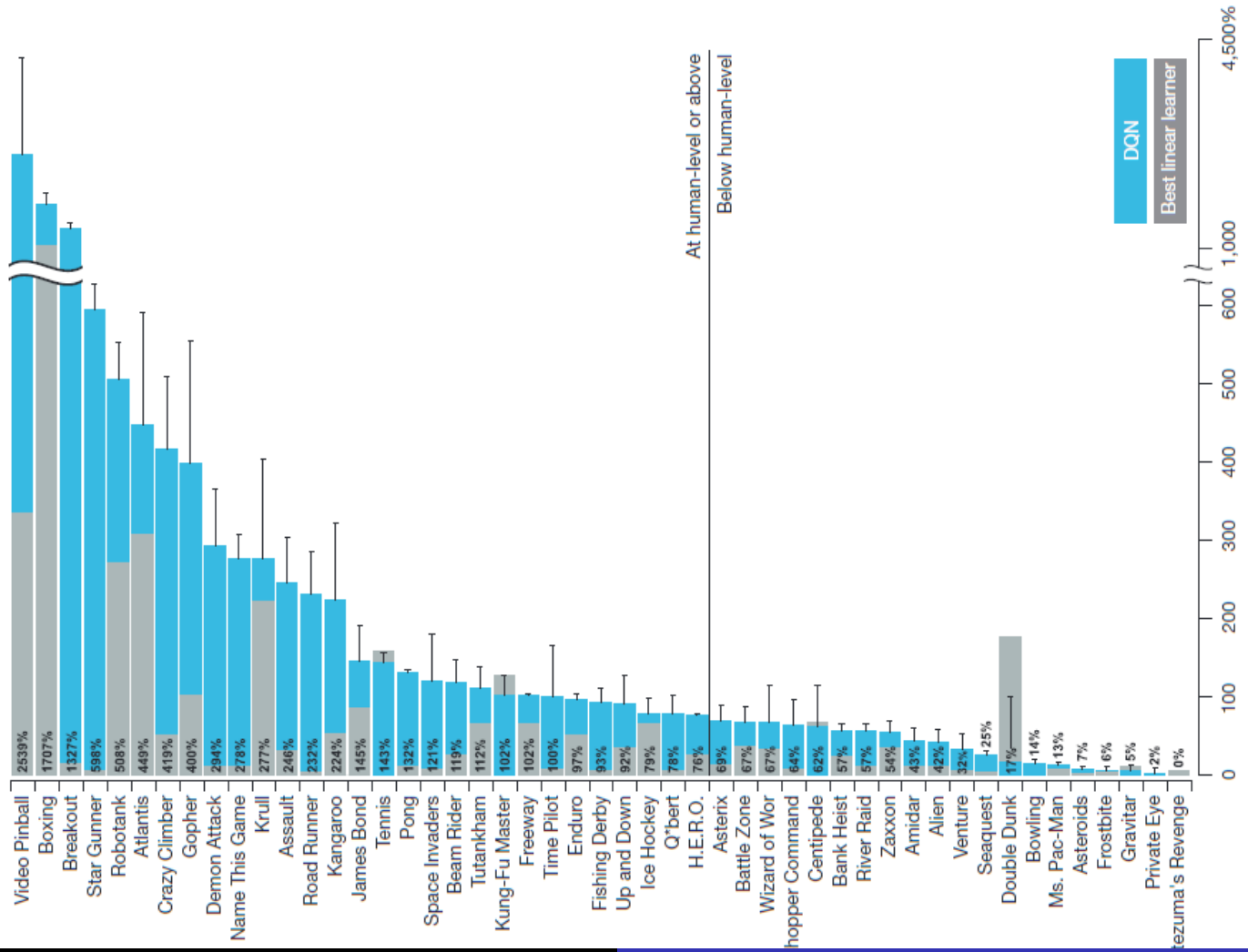


Deep Q-Network for Atari

- 输入由一个84 x 84 x 4的图像组成，该图像是由预处理映射ϕ 生成的。
- 接下来是三个卷积层和两个全连接层。
- 每个隐藏层后面都有一个整流线性单元。
- 对于每个有效的动作，有一个单独的输出。
- 在考虑的游戏中，有效动作的数量在4到18之间。

DQN 与线性近似的比较

DQN的归一化性能计算公式为：
(DQN得分- 随机游戏得分) / (人类得分- 随机游戏得分)



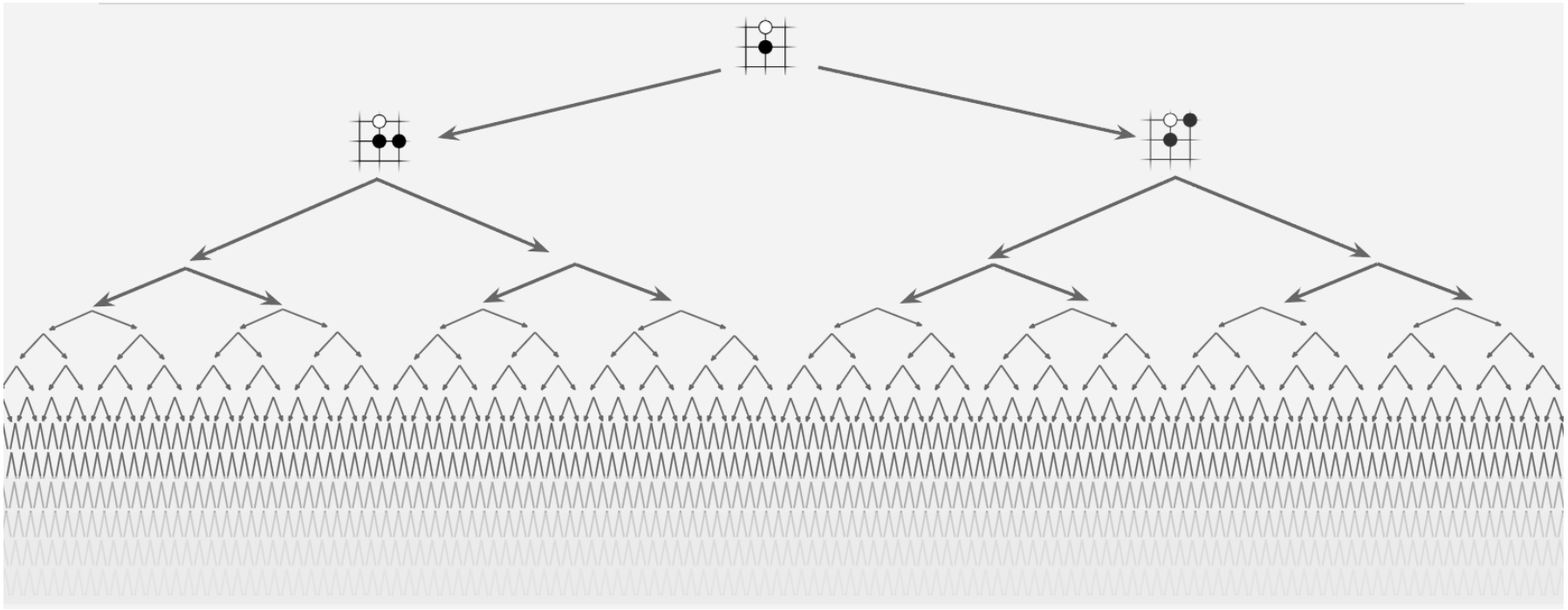
AlphaGo

- 计算机围棋非常困难，原因在于围棋庞大的搜索空间以及对棋盘位置和棋步进行评估的困难性。
- 一种新的计算机围棋方法：采用价值网络来评估棋盘位置，以及策略网络来选择棋步。
- 这些深度神经网络通过结合人类专家对弈的监督学习和自我对弈的强化学习来进行训练。
- 一种新的搜索算法：将蒙特卡洛模拟与价值网络和策略网络相结合。

完美信息博弈

- 具有一个最优值函数$v^*(s)$，从每个棋盘位置或状态s开始，在所有玩家完美对弈下。
- 这类游戏可以通过在搜索树中递归计算最优值函数来求解。
  - 包含$\approx b^d$ 个可能的动作序列，
  - 其中b 是游戏的宽度（每个位置上合法动作的数量）
  - d 是游戏的深度（游戏长度）

穷举搜索



完美信息博弈

- 在大规模博弈中，穷举搜索是不可行的。
  - 国际象棋($b \approx 35$, $d \approx 80$) and 围棋($b \approx 250, d \approx 150$)
- 然而，可以通过两个一般原则来减少有效搜索空间：

减少搜索深度

- 通过位置评估: 通过一个近似值函数$v(s) \approx v^*(s)$ 来截断状态s 处的搜索树
- 这种方法已经在国际象棋、跳棋和黑白棋等游戏中实现了超人类的表现。
- 然而，由于围棋的复杂性，人们认为这种方法在围棋中是难以应用的。

减少搜索的广度

- 通过从策略$p(a|s)$ 中采样动作，其中$p(a|s)$ 是在位置s上可能移动a 的概率分布。
- 例如，蒙特卡洛模拟可以通过从策略p 中为两个玩家采样长序列的动作，不进行任何分支而搜索到最大深度。
- 通过对这些模拟结果求均值，可以提供有效的位置评估，并在backgammon和Scrabble等游戏中实现超人类的表现。

蒙特卡洛树搜索(MCTS)

- 使用蒙特卡洛模拟来估计搜索树中每个状态的值。
- 随着执行更多的模拟，搜索树变得更大，相关的值也变得更准确。
- 在搜索过程中用于选择动作的策略也会随着时间的推移而改进，从而选择具有较高值的子节点。
- 这个策略收敛到最优游戏策略，并且状态评价收敛到最优值函数。

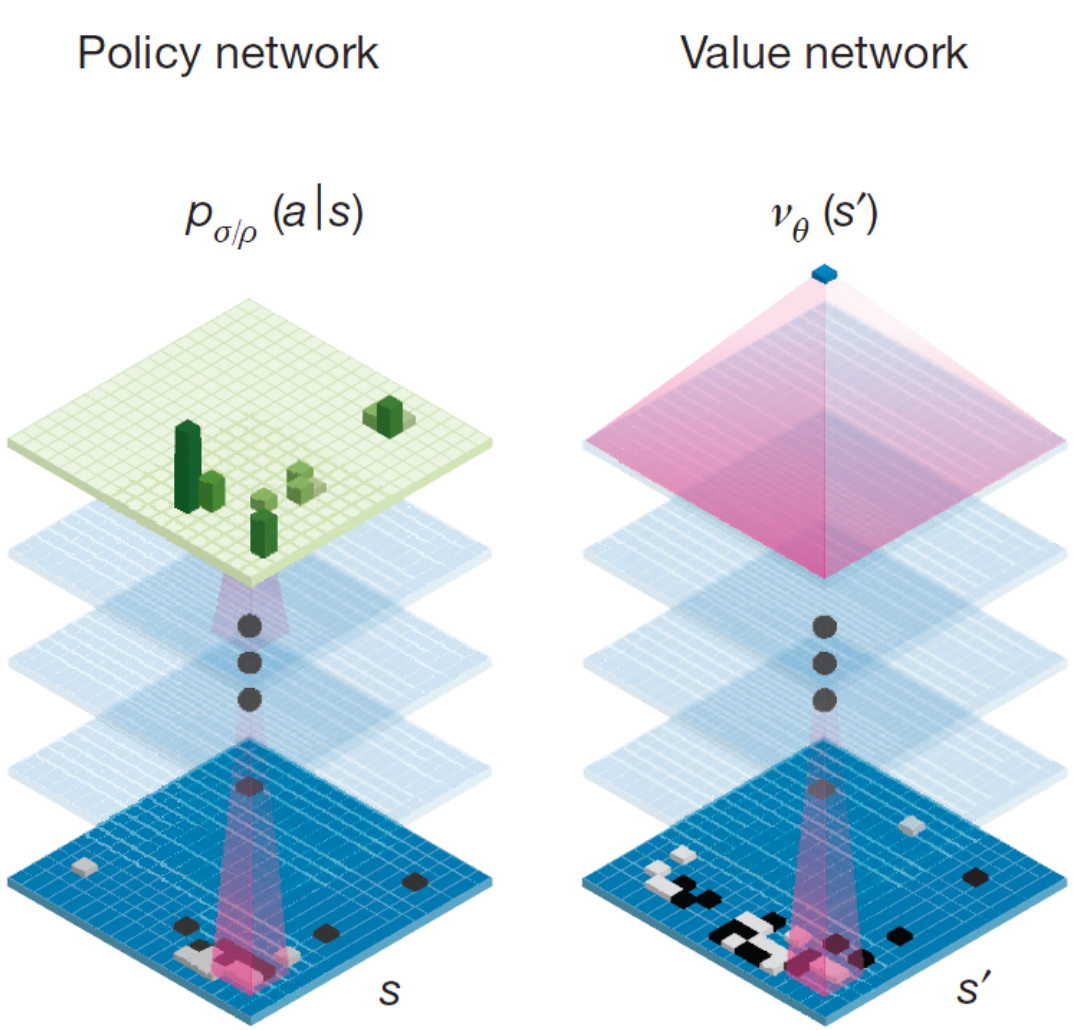
改进MCTS

- MCTS（蒙特卡洛树搜索）可以通过训练用于预测人类专家走法的策略来改进。
- 这些策略被用于将搜索范围缩小到一组高概率动作，并在模拟过程中采样动作。

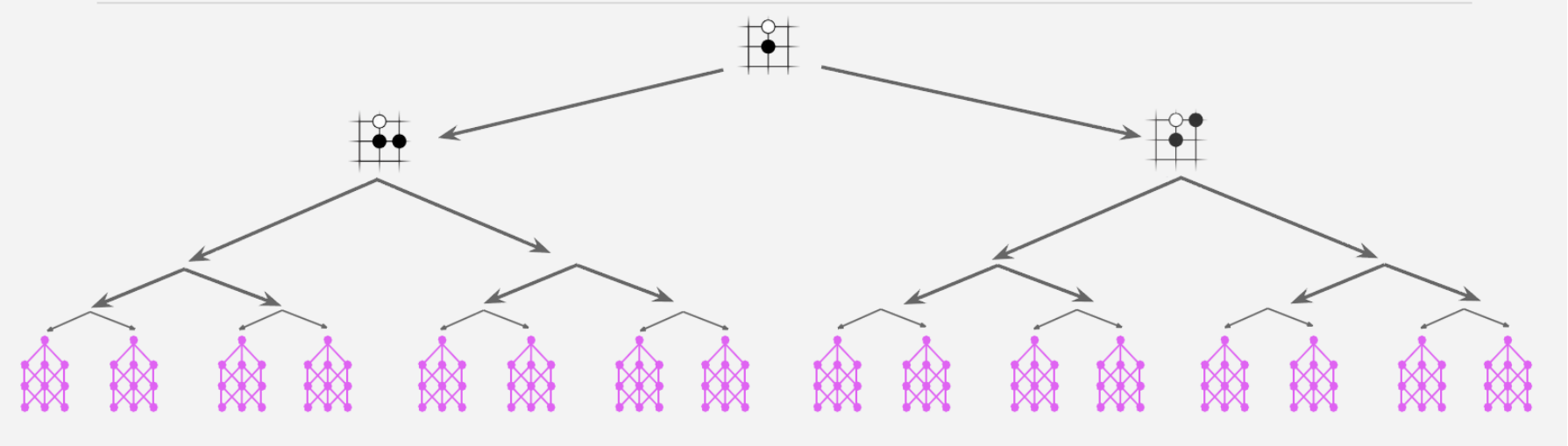
AlphaGo

- 将棋盘位置作为一个19×19的图像传入，并使用卷积层构建位置的表示。
- 利用这些神经网络来减少搜索树的有效深度和广度：使用价值网络评估位置，并使用策略网络采样动作。

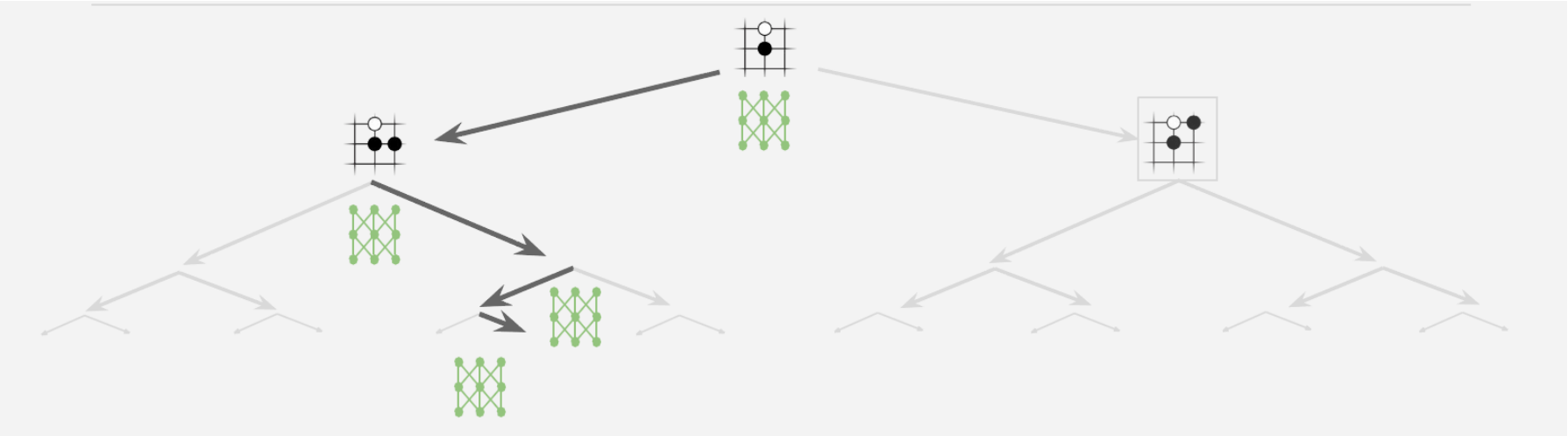
策略和价值网络



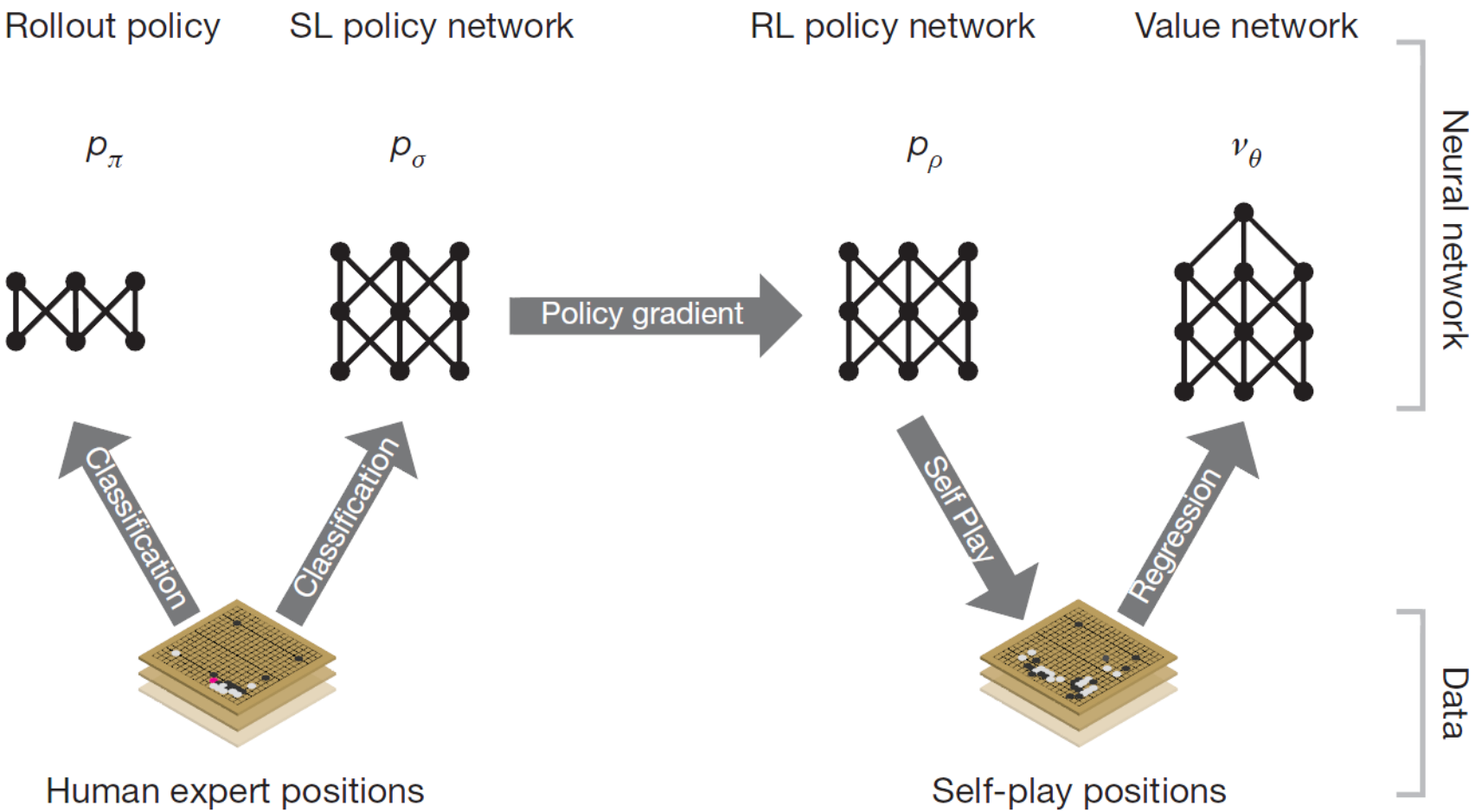
用价值网络减小深度



用策略网络减小宽度



神经网络训练管道


Rollout policy
SL policy network
RL policy network
Value network
p_π
p_σ
p_ρ
v_θ
Policy gradient
Classification
Classification
Self Play
Regression
Neural network
Data
Human expert positions
Self-play positions

神经网络训练管道

- 快速模拟策略p_π和监督学习（SL）策略网络p_σ被训练以预测人类专家在一个位置数据集中的走法。
- 强化学习（RL）策略网络p_ρ被初始化为SL策略网络，然后通过策略梯度学习进行改进。
- 使用RL策略网络进行自我对弈，生成新的数据集。
- 最后，通过回归训练价值网络v_θ，以预测自我对弈数据集中的位置的期望结果（即当前玩家是否获胜）。

Supervised learning of policy networks

Policy network: 12 layer convolutional neural network

Training data: 30M positions from human expert games (KGS 5+ dan)

Training algorithm: maximise likelihood by stochastic gradient descent

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma}$$

Training time: 4 weeks on 50 GPUs using Google Cloud

Results: 57% accuracy on held out test data (state-of-the art was 44%)

随机采样的状态-动作对(s, a)

Reinforcement learning of policy networks

Policy network: 12 layer convolutional neural network

Training data: games of self-play between policy network

Training algorithm: maximise wins z by policy gradient reinforcement learning

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma} z$$

Training time: 1 week on 50 GPUs using Google Cloud

Results: 80% vs supervised learning. Raw network ~3 amateur dan.

结果z是终局奖励：赢棋为+1，输棋为-1。

与SL策略网络对战的胜率超过80%。

Reinforcement learning of value networks

Value network: 12 layer convolutional neural network

Training data: 30 million games of self-play

Training algorithm: minimise MSE by stochastic gradient descent

$$\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial\theta}(z - v_\theta(s))$$

Training time: 1 week on 50 GPUs using Google Cloud

Results: First strong position evaluation function - previously thought impossible