AlphaGo介绍

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

2016.10.20

开源代码： https://github.com/Rochester-NRT/RocAlphaGo

论文

1. David Silver et al., Mastering the Game of Go with Deep Neural Networks and Tree Search, Nature, 2015.

2. David Silver et al., Mastering the Game of Go without Human Knowledge,Nature, 2017.

围棋规则 （中国规则）

1. 无气自提

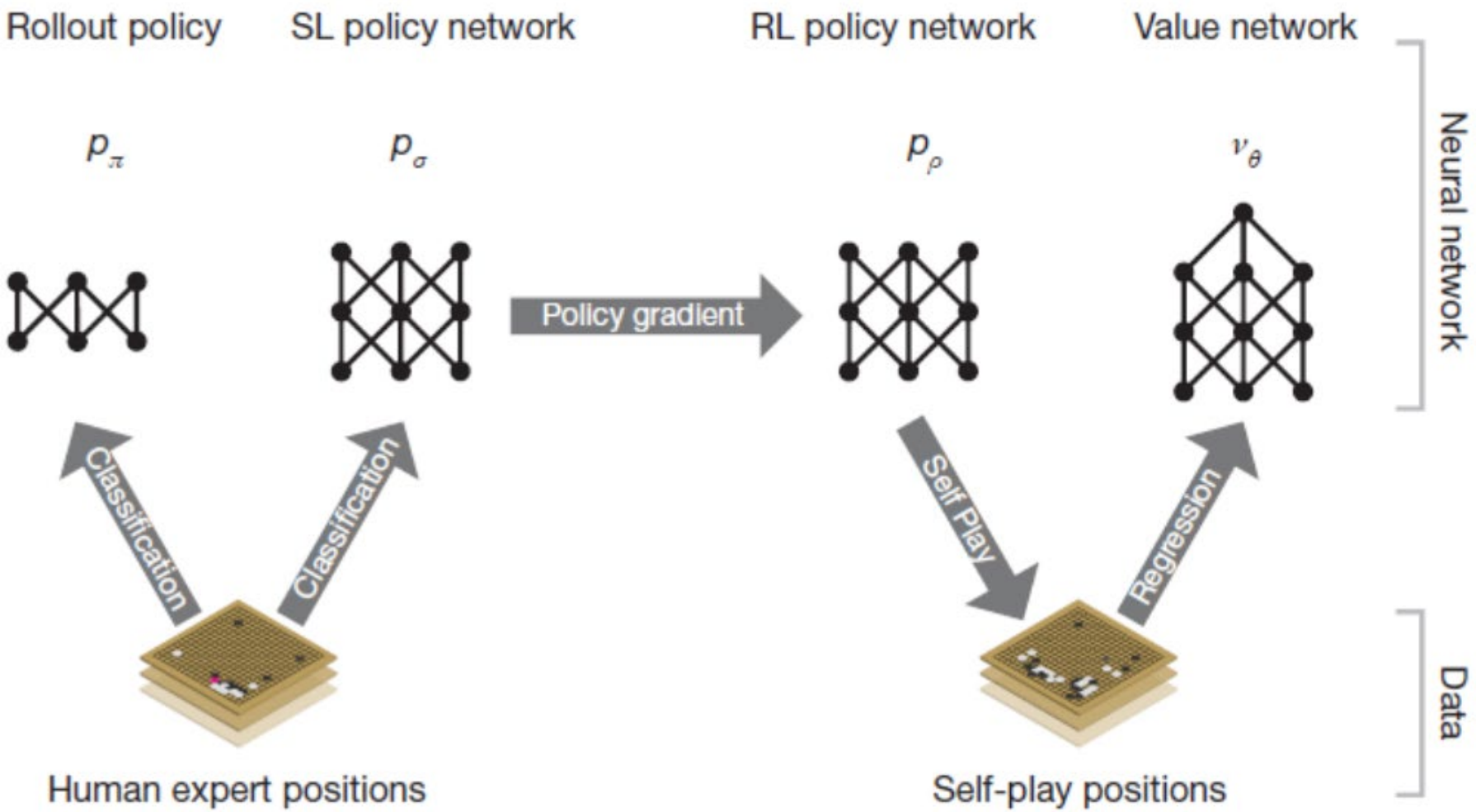
2. 禁止全局同形

-- 每走一步，棋盘状态（即黑子和白子的位置）都必须与以前所有步的棋盘状态不一样。

3. 地大者胜

围棋有必胜策略！

三个深度策略网络 (Policy Networks),一个深度估值网络 (Value Network)



深度策略网络 p_σ：(Supervised Learning Policy Network)

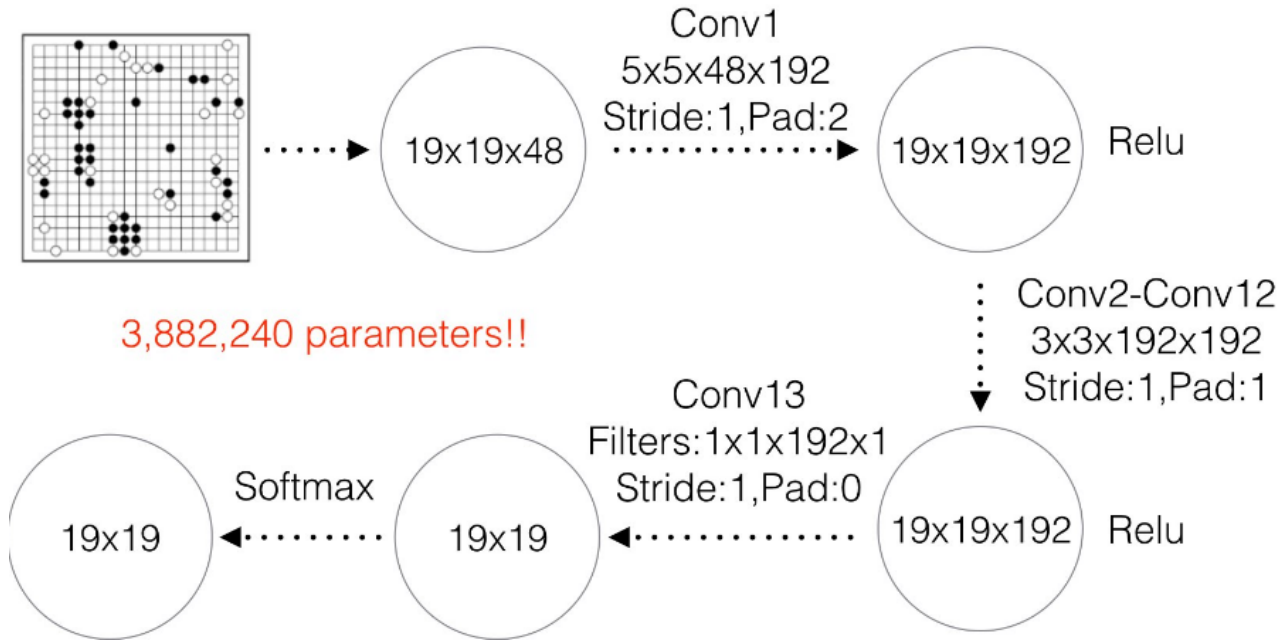
输入：当前棋盘状态。

输出：下一步的走法。

训练数据： KGS Go SERVER上的 三亿个样本。

网络设置： 13层深度网络。

输入的特征：



57%正确率,

3ms一步

深度策略网络 p_σ：(Supervised Learning Policy Network)

- Input：棋盘特征（19x19x48)
- Output：每个位置的选择概率
- Cost：Cross-Entropy Loss
- Training：异步SGD 50GPU 3周
- Training Step：340,000,000
- Batch Size:16
- Learning Rate: 初始0.003　80,000,000步后0.0015

深度策略网络 p_σ：(Supervised Learning Policy Network)，

优化分析：

目标：最大化概率值(让正确动作概率趋于1)

==

最大化对数似然函数log likelihood（使趋近于0)

==

最小化：$L(\sigma) = -\sum_k a^k log\left(p_\sigma\left(a^k \mid s^k\right)\right)$ Cross-Entropy Loss

One Hot Encoding

梯度：$\Delta\sigma = \frac{\alpha}{m}\sum_{k=1}^{m}\frac{\partial \log p_\sigma(a^k|s^k)}{\partial\sigma}$

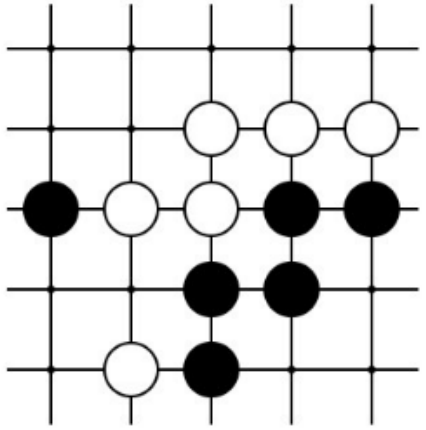
深度策略网络 p_σ：(Supervised Learning Policy Network)，

棋盘特征：

特征	平面数量	说明
棋子颜色	3	自己的棋子、对手棋子、空白位置
1	1	全1平面
轮次	8	每个落子过后经过的轮次
气	8	每个落子气的数量（邻近空的点）
打吃	8	对手被打吃的数目
被打吃	8	自己被打吃的数目
落子后的气	8	每个落子刚落之后气的数量
征子有利	1	落子是否征子有利
征子逃脱	1	落子是否征子逃脱
合法性	1	落子是否合法并且没有填自己的眼
0	1	全0平面
颜色	1	是否当前是执黑

深度策略网络 p_σ：(Supervised Learning Policy Network)，

落子颜色（Stone Color）：



```
[0 0 0 0 0]
[0 0 1 1 1]
[0 1 1 0 0]
[0 0 0 0 0]
[0 1 0 0 0]
```

自己棋子
(这里是白棋)

```
[0 0 0 0 0]
[0 0 0 0 0]
[1 0 0 1 1]
[0 0 1 1 0]
[0 0 1 0 0]
```

对方棋子
(这里是黑棋)

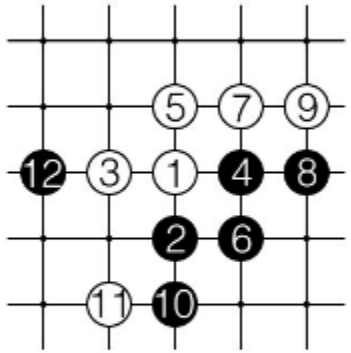
```
[1 1 1 1 1]
[1 1 0 0 0]
[0 0 0 0 0]
[1 1 0 0 1]
[1 0 0 1 1]
```

空白位置

深度策略网络 p_σ：(Supervised Learning Policy Network)，

轮次 （Turn Since）：

One Hot encoding 二值化



P1

```
[0 0 0 0 0]
[0 0 0 0 0]
[1 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
```

P2

```
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
[0 1 0 0 0]
```

P3

```
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 1 0 0]
```

P4

```
[0 0 0 0 0]
[0 0 0 0 1]
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
```

P5

```
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 1]
[0 0 0 0 0]
[0 0 0 0 0]
```

P6

```
[0 0 0 0 0]
[0 0 0 1 0]
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
```

P7

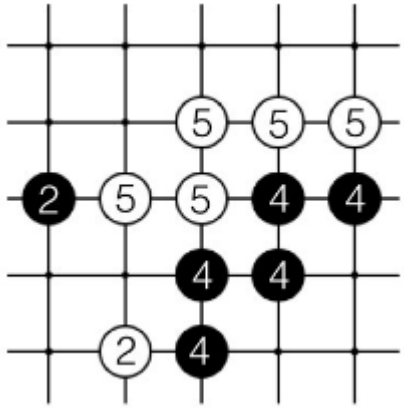
```
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 1 0]
[0 0 0 0 0]
```

P8

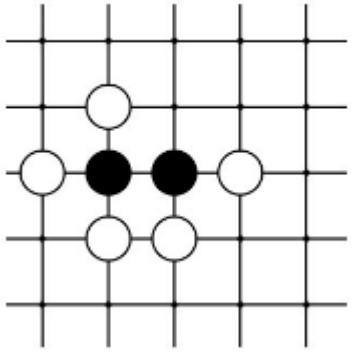
```
[0 0 0 0 0]
[0 0 1 0 0]
[0 1 1 1 0]
[0 0 1 0 0]
[0 0 0 0 0]
```

深度策略网络 p_σ：(Supervised Learning Policy Network)，

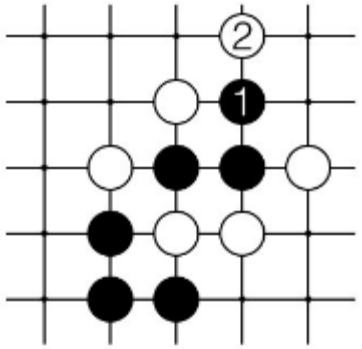
气（Liberty）：



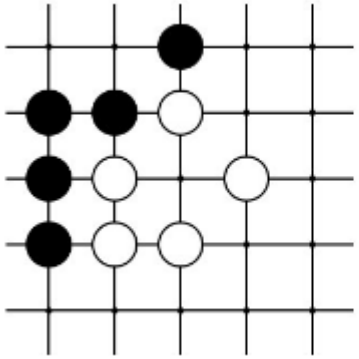
打吃数量：



征子（Ladder）：



合法性（Sensibility）：



```
[1 1 0 1 1]
[0 0 0 1 1]
[0 0 0 0 1]
[0 0 0 1 1]
[1 1 1 1 1]
```

深度策略网络 p_ρ：(Reinforcement Learning Policy Network)

1. 网络结构、输入输出与 p_σ 完全一样。

2. 一开始初始化网络参数 $\rho = \sigma$

3. 参数更新策略，自己和自己下棋，不断下下去直到分出胜负。

$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial\rho} z_t$$ （Policy Gradient）

上式中，$p_\rho(a_t|s_t)$ 为在第t步走下一步 a_t 的概率，当胜利时，z_t 等于1，否则 z_t 等于0。

强化学习训练策略：

- Step 1：将监督学习的网络复制作为增强学习的初始网络
- Step 2：将当前版本的网络与之前的某个随机的版本对局，得到棋局和棋局结果（输赢）
- Step 3：根据棋局和棋局结果利用REINFORCE 算法更新参数最大化期望结果（赢）
- Step 4：每500次迭代就复制当前网络参数到对手池中用于Step 2

深度策略网络 p_π：(Rollout Policy Network)

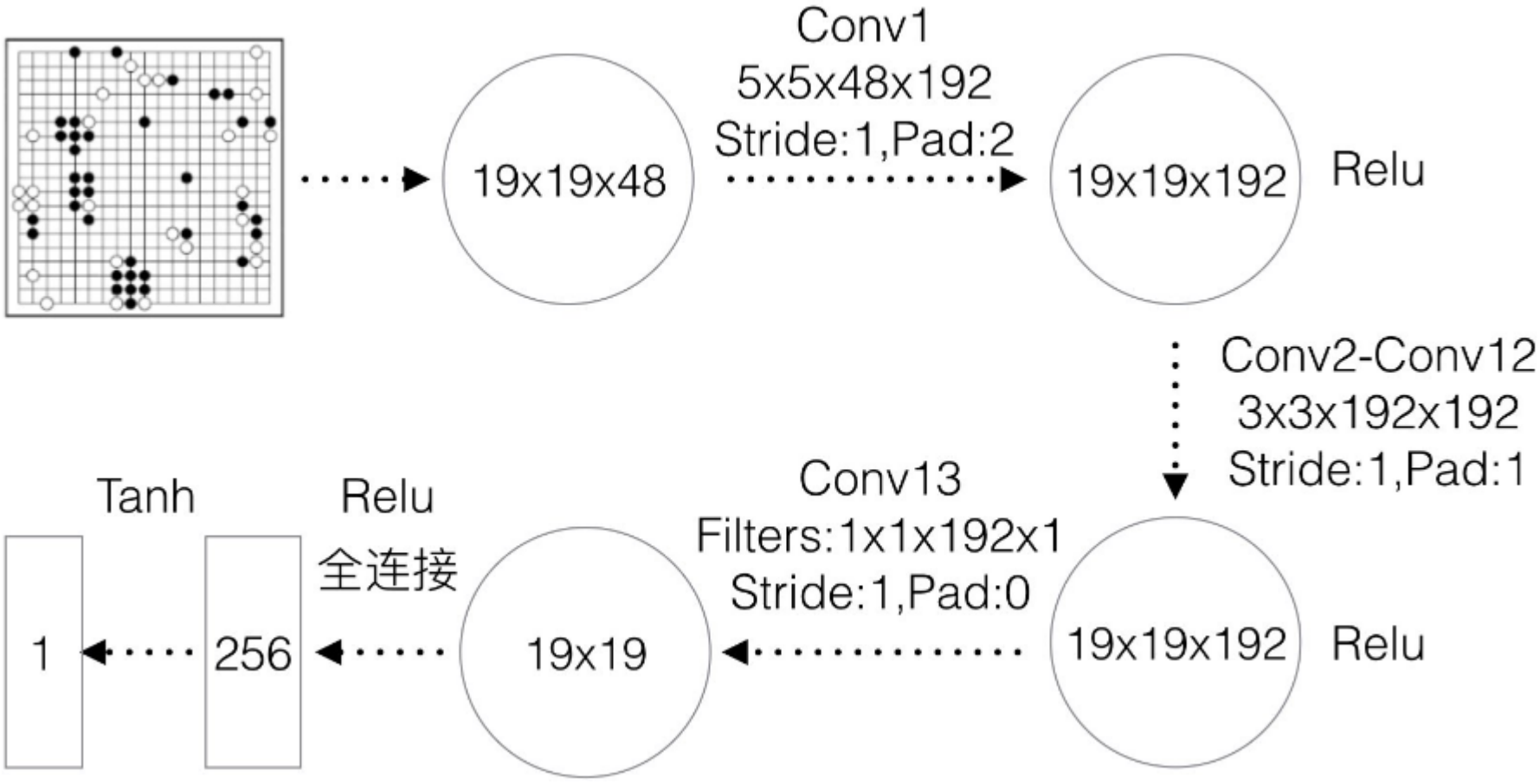
1. 输入特征比 p_σ 和 p_ρ 少。

Feature	# of patterns	Description
Response	1	Whether move matches one or more response pattern features
Save atari	1	Move saves stone(s) from capture
Neighbour	8	Move is 8-connected to previous move
Nakade	8192	Move matches a *nakade* pattern at captured stone
Response pattern	32207	Move matches 12-point diamond pattern near previous move
Non-response pattern	69338	Move matches 3 × 3 pattern around move
Self-atari	1	Move allows stones to be captured
Last move distance	34	Manhattan distance to previous two moves
Non-response pattern	32207	Move matches 12-point diamond pattern centred around move

2. 网络结构更简单。

换句话说，这个网络以牺牲准确率换取速度。24.2%正确率，2um一步。

深度估值网络 v_θ ：(Rollout Policy Network)



深度估值网络 v_θ ：(Rollout Policy Network)

1. 输入：当前棋盘状态 （与 p_σ 输入一样），以及执黑或执白。

2. 输出： 获胜的概率（一个0到1的数）

3. 参数更新策略：

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

用 p_π来走很多轮来预测 真实值 z 。

进一步深化 p_ρ 的训练

- 使用minibatch 进行参数更新：

$$\Delta\rho = \frac{\alpha}{n}\sum_{i=1}^{n}\sum_{t=1}^{T^i}\frac{\partial \log p_\rho(a_t^i|s_t^i)}{\partial\rho}(z_t^i - v(s_t^i))$$

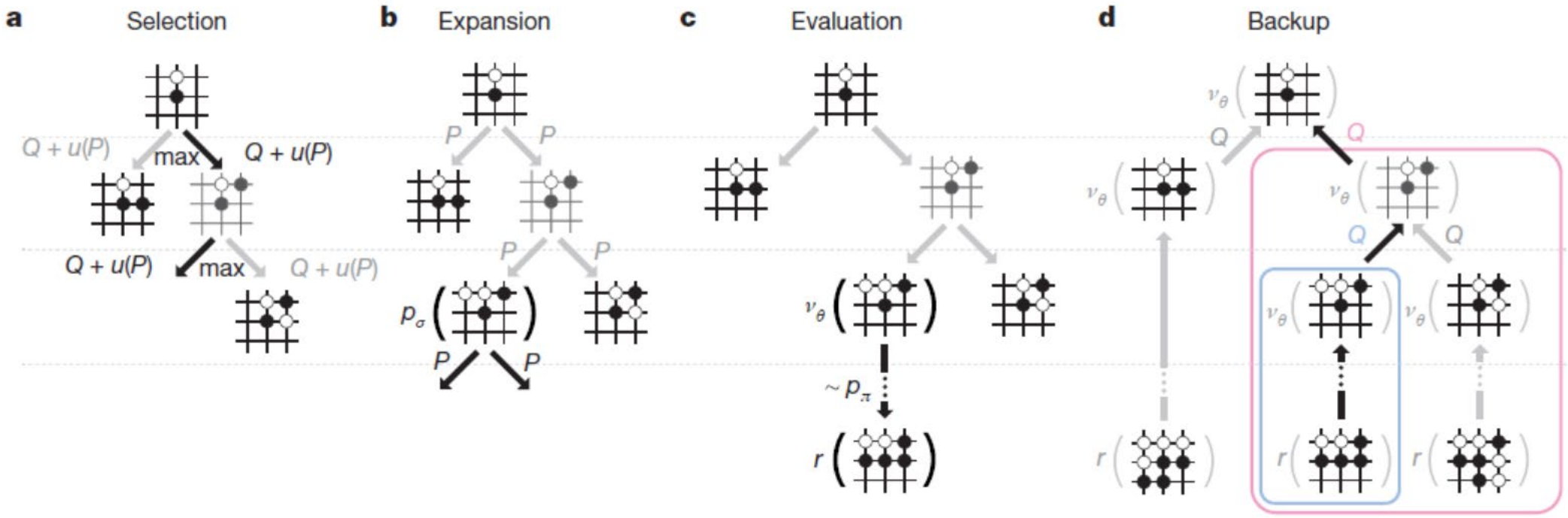
（**Actor-Critic**）

- 对 SL Policy Network达到80%的胜率，自学习质的飞跃！

AlphaGo 原理

下棋方法 -- 蒙特卡洛树搜索 （Monte Carlo Tree Search）：

多次模拟未来棋局，然后选择在模拟中选择次数最多的走法



蒙特卡洛树搜索 （Monte Carlo Tree Search）最终确定走棋。

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

$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)} \qquad P(s,a) = p_\rho(a|s)$$

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

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

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

下棋方法 -- 蒙特卡洛树搜索 （Monte Carlo Tree Search）

- Step 1:基于SL Policy Network来预测未来的下一步走法，直到L步，到一个叶节点。如果该节点访问次数大于一个阈值，那么就使用Tree Policy拓展下一个节点（Tree Policy比Rollout Policy多一些特征）

下棋方法 -- 蒙特卡洛树搜索 （Monte Carlo Tree Search）

- Step 2：结合两种方式来对未来到L的走势进行评估，一个是使用Value Network进行评估，判断赢面，一个是使用Rollout Network做进一步的预测直到比赛结束得到模拟的结果。综合两者（两者权重各0.5）对预测到未来L步走法进行评估。

下棋方法 -- 蒙特卡洛树搜索 （Monte Carlo Tree Search）

- Step 3: 评估完，将评估结果作为当前棋局下的下一步走法的Q值。即给一开始给出的下一步走法根据未来的走向进行评估，Q值越大，之后模拟选择的次数就会越多。

下棋方法 -- 蒙特卡洛树搜索 （Monte Carlo Tree Search）

- Step 4：结合下一步走法的Q值和SL Policy Network进行再一次的模拟，如果出现同样的走法，则对走法的Q值取平均（蒙特卡洛的思想在这里）

下棋方法 -- 蒙特卡洛树搜索 （Monte Carlo Tree Search）

- Step 5：反复循环上面的步骤到n次，然后选择选择次数最多的走法作为下一步。

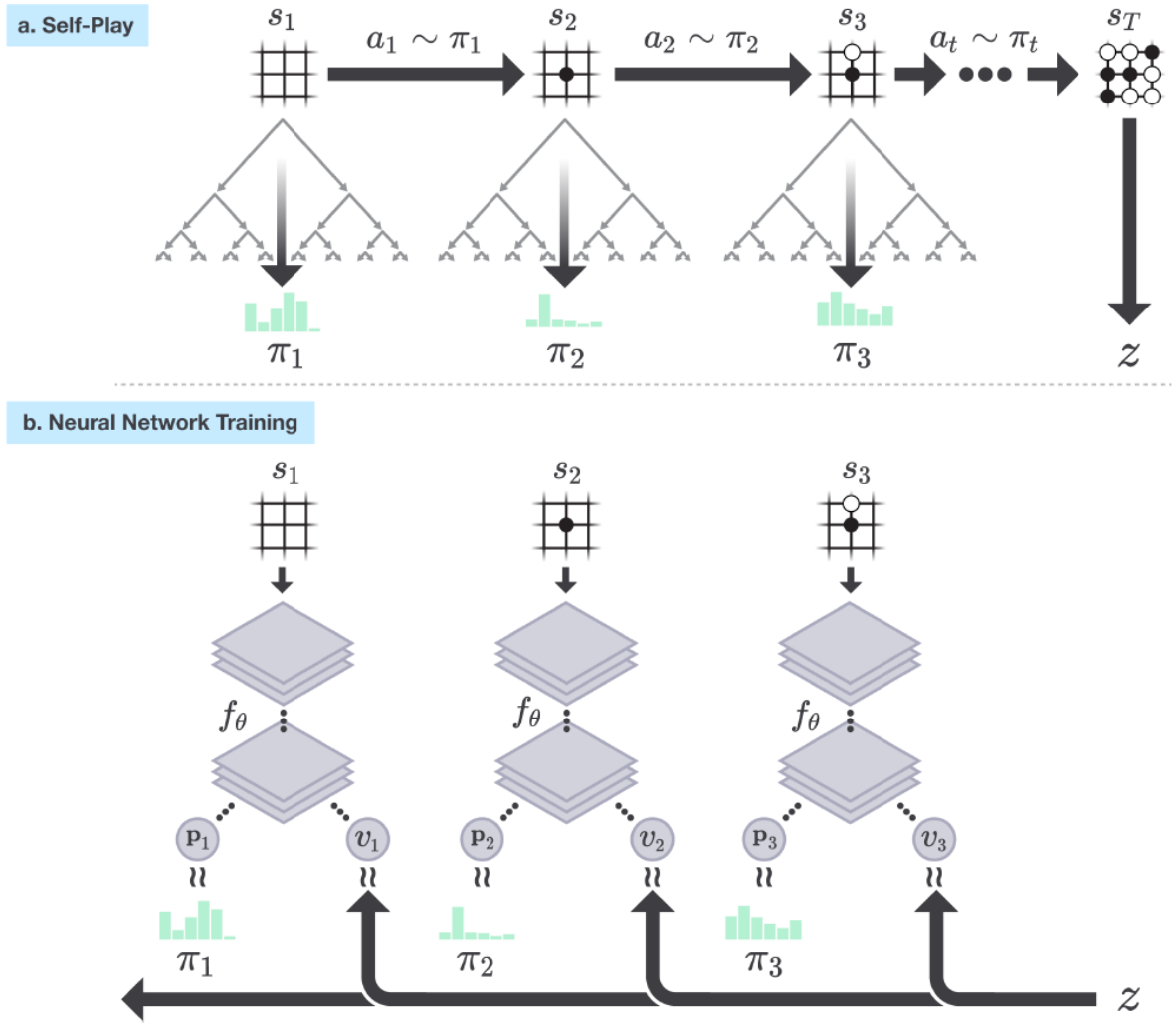
AlphaGo Zero 的改进

（1）完全不需要人类棋谱，采用自己和自己下棋的方式学习。

（2）将走棋网络和估值网络合并为一个网络：

$$(\mathrm{p}, v) = f_\theta(s)$$

AlphaGo Zero 的改进



自学习过程和神经网络训练过程

AlphaGo Zero 的改进

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

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

$$N(s,a) = \sum_{i=1}^{n} l = (z-v)^2 - \boldsymbol{\pi}^\top \log \mathbf{p} + c||\theta||^2$$

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

$$\pi_a \propto N(s,a)^{1/\tau}$$

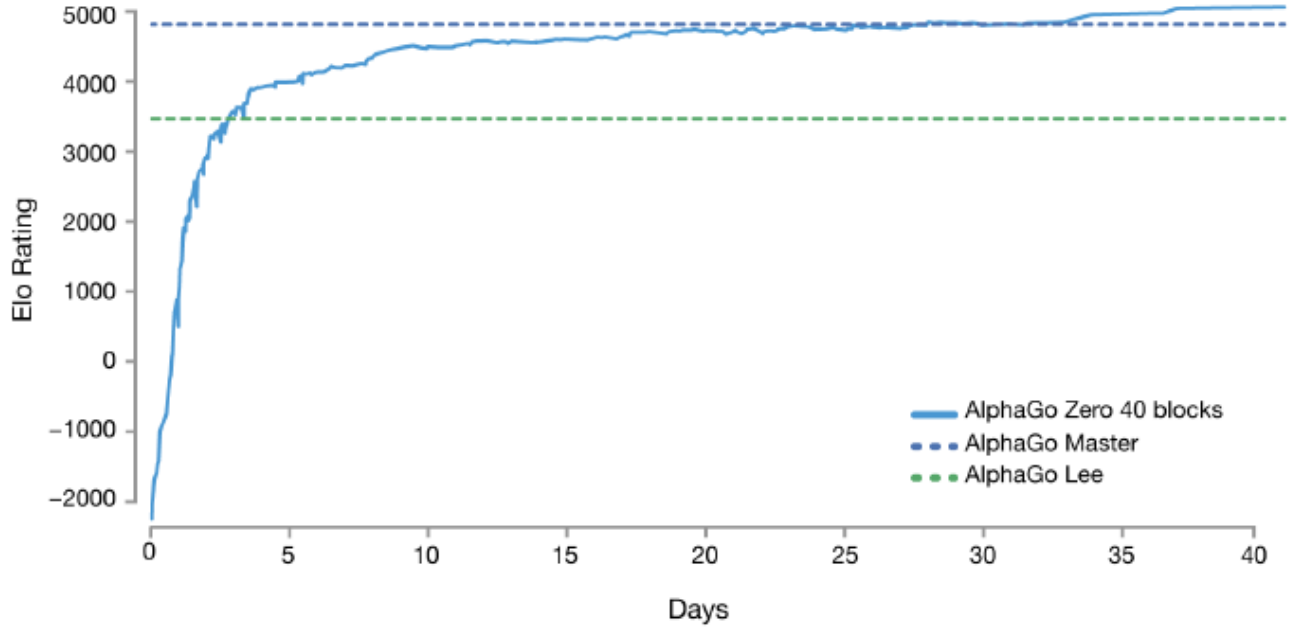
标签 π 的生成

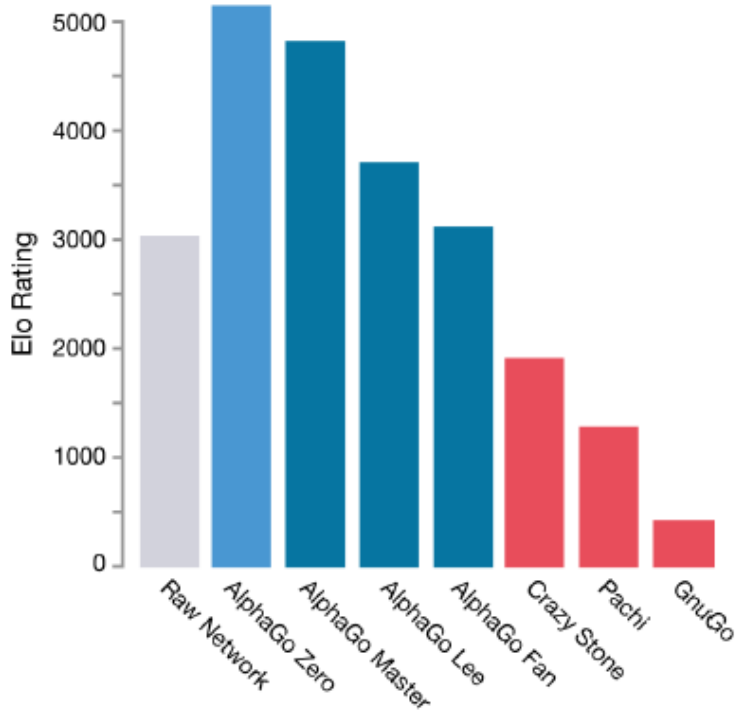
$$(\mathbf{p}, v) = f_\theta(s),$$

$$l = (z - v)^2 - \boldsymbol{\pi}^\top \log \mathbf{p} + c||\theta||^2$$

目标函数

AlphaGo Zero 的改进





实验结果对比