AlphaGo

Дмитрий Пыркин

151

23 января 2018 г.

Table of Contents

Что за AlphaGo?

AlphaGo – первая программа для игры в го, которая выиграла матч без форы у профессионального игрока на стандартной доске 19x19.
Разработана Google DeepMind в 2015 году.

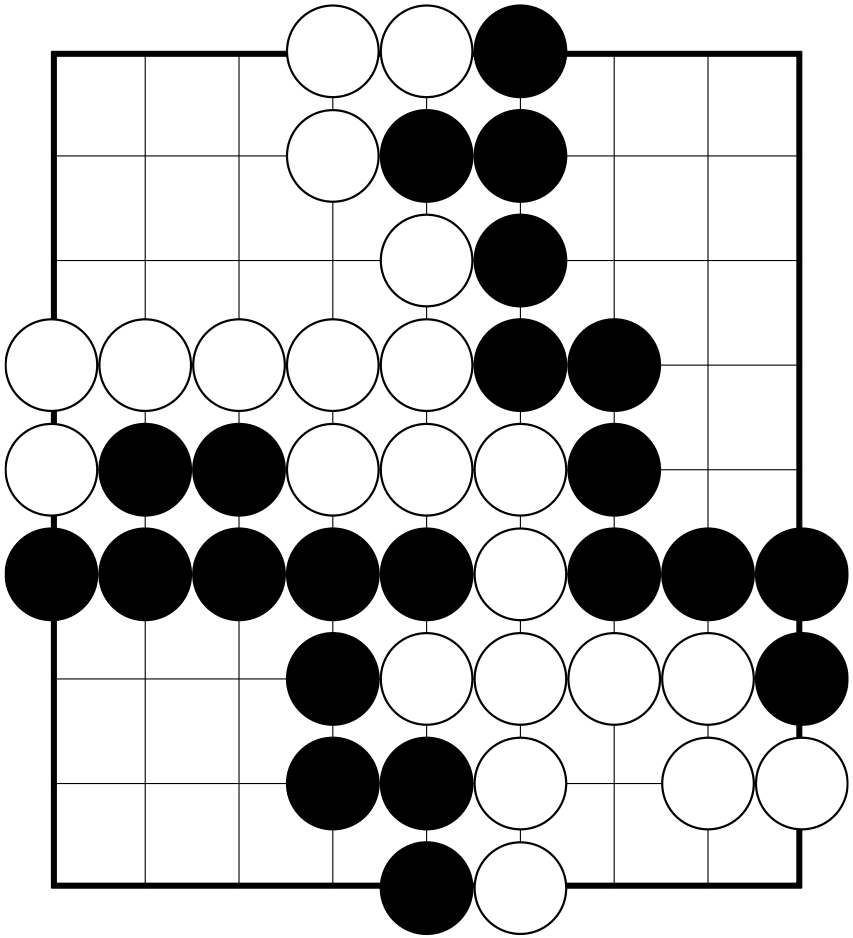
В чём инновационность?

- В го количество возможных позиций намного больше чем в шахматах, поэтому традиционные методы, как например, альфа-бета отсечение давали довольно слабые результаты.
- AlphaGo представляет собой комбинацию Supervised learning (SL) и Reinforcement learning (RL)

Основные правила

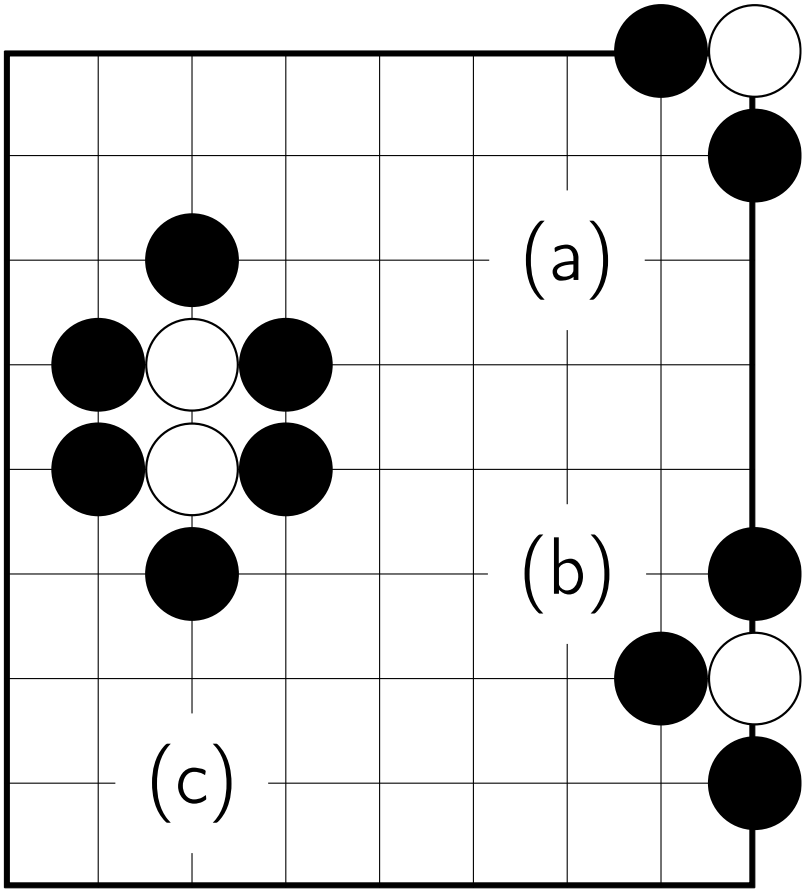
1. Игра начинается с пустой доски. Игроки по очереди ставят на доску камни своего цвета: чёрные и белые.
2. Побеждает тот кто окружит больше территории.
3. Окружённые камни снимаются с доски.
4. Есть запрещённые ходы.
5. Запрещается повторять позицию: правило ко.

Подсчёт очков



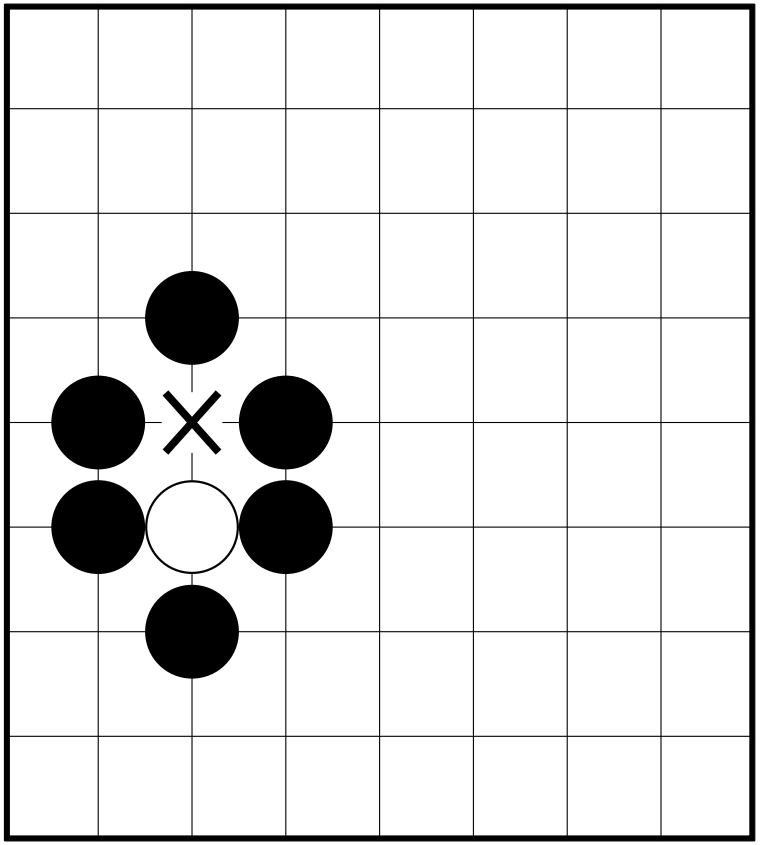
Здесь
у черных территория
в двух местах -
справа вверху и слева
внизу. Территория
белых вверху
слева и внизу справа.

Удаление камней с доски



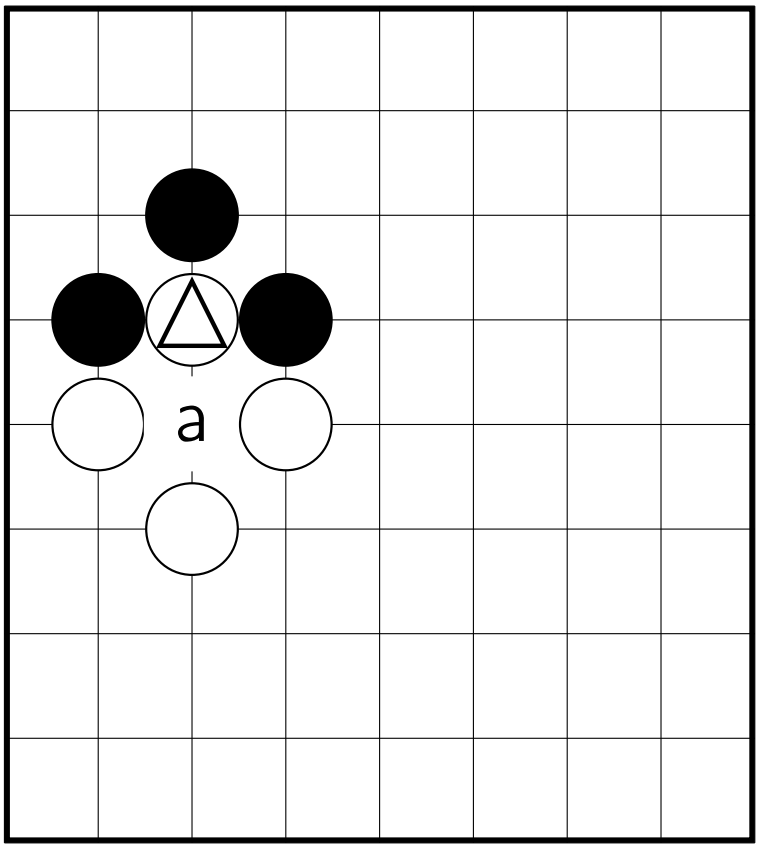
В данном примере во всех трёх случаях белые камни оказывается захваченными и снимаются с доски.

Самоубийственные ходы



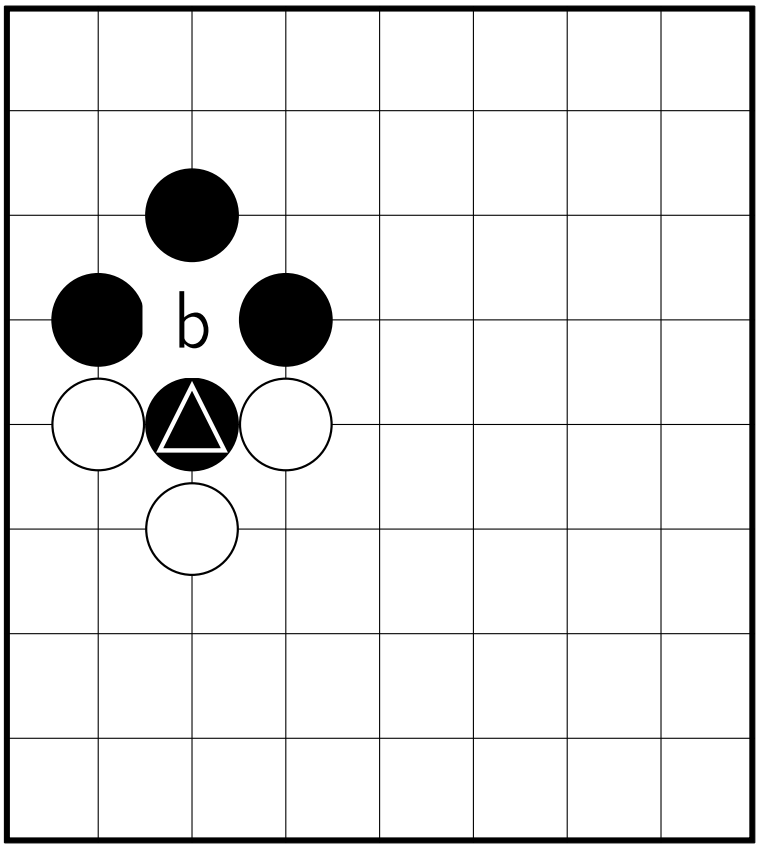
В данном случае белые не могут пойти в пункт, отмеченный крестиком, т.к. это самоубийственный ход.

Ко



Чёрные могут взять отмеченный белый камень, если сделают ход в пункт a.

Ko



Белые
не могут сходить
в пункт b сейчас,
но смогут через ход.

Из чего состоит AlphaGo?

1. SL policy network
2. Fast rollout policy
3. RL policy network
4. Value network
5. Monte-Carlo tree search

Что требуется от SL policy network?

- Предсказание хода, сделанного человеком.
- Требуется большая точность. В релизе **57%**.
- Каждое предсказание занимает приблизительно 3ms.

Структура SL policy network

- Для каждой клетки на вход подаётся 48 признаков.
- 13 свёрточных слоев с softmax на выходном слое.
- Для обучения было использовано 30 миллионов позиций.

Как это обучать?

Обучение SL policy network

Используем стохастический градиентный спуск.

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

σ – веса сети.

s – текущая позиция на доске.

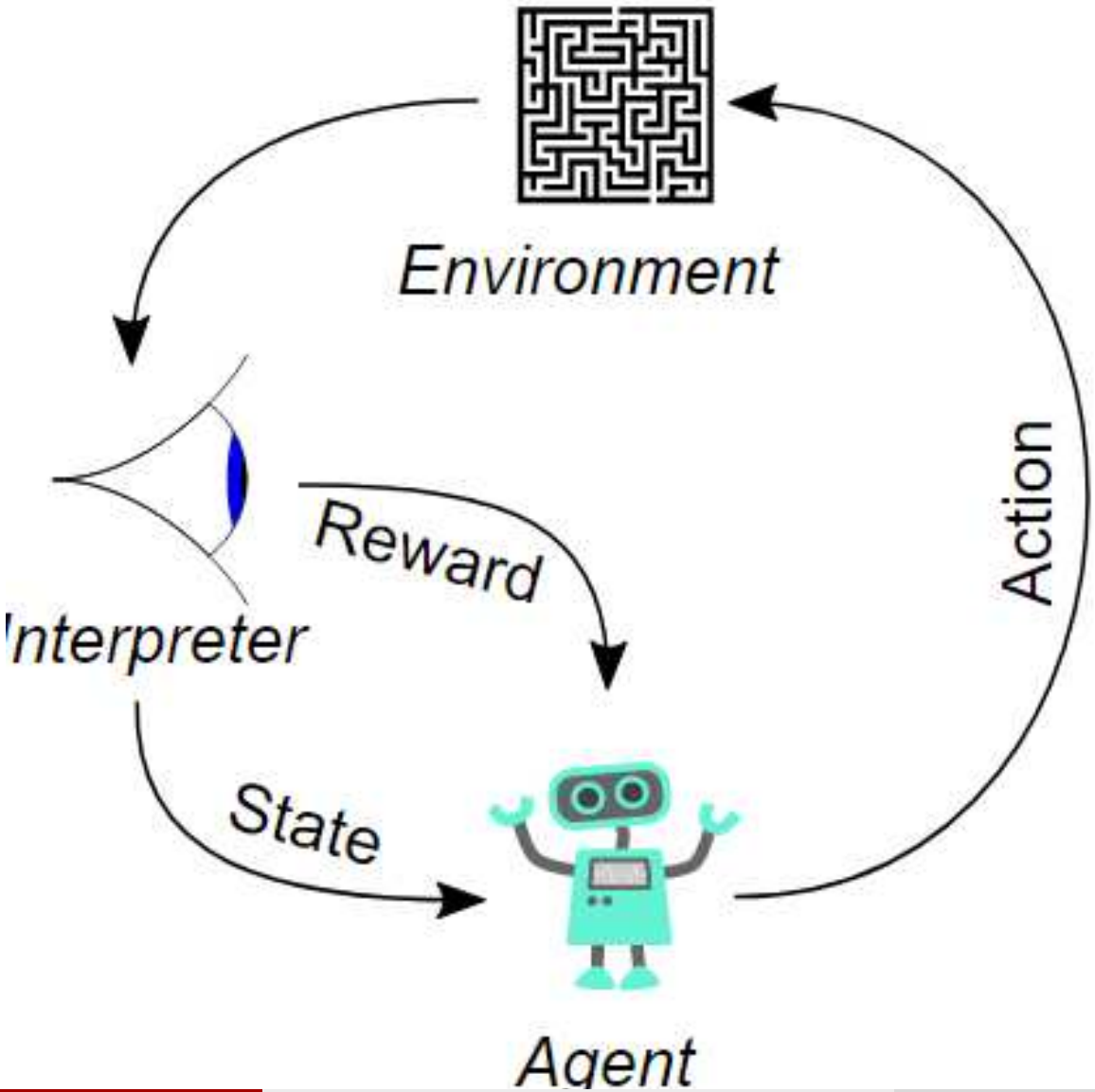
a – сделанный ход.

Fast rollout policy

- Делает то же, что и SL policy network.
- Точность 24.2%.
- Время работы $2\mu s$.
- На вход подаётся очень большое количество заранее заготовленных признаков.

Как улучшить точность SL policy network?

Пришло время reinforcement learning



А как использовать RL для нейронных сетей?

Дообучение SL policy network

- RL policy network имеет такую же структуру, как и SL.
- Изначально её веса инициализируются весами SL. $\rho = \sigma$.

Reinforce algorithm

- На каждой итерации играем одновременно n игр между текущей сетью p_ρ и случайной сетью с предыдущих итераций p_{ρ^-}.
- Каждая игра i из пула T^i играется до конца.

У нас есть состояния и действия, что насчёт награды?

$$r(s) = \begin{cases} 0, \text{ for non terminal state}, & t < T \\ 1, \text{ for terminal state}, & t = T \end{cases}$$

У нас есть состояния и действия, что насчёт награды?

$$r(s) = \begin{cases} 0, \text{ for non terminal state, } & t < T \\ 1, \text{ for terminal state, } & t = T \end{cases}$$

$$z_t = \pm r(s_T)$$

Outcome с точки зрения текущего игрока.

Обновление весов

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

α – step parameter.

$v(s_t^i)$ – baseline. Используется для уменьшения дисперсии.

Value-network

Оценивает позицию s при условии, что оба игрока используют стратегию p.

$$v^p(s) = \mathbb{E}[z_t | s_t = s, a_{t...T} \sim p]$$

Value-network

Оценивает позицию s при условии, что оба игрока используют стратегию p.

$$v^p(s) = \mathbb{E}[z_t | s_t = s, a_{t...T} \sim p]$$

$$v_\theta(s) \approx v^p(s) \approx v^*(s)$$

Обучение Value-network

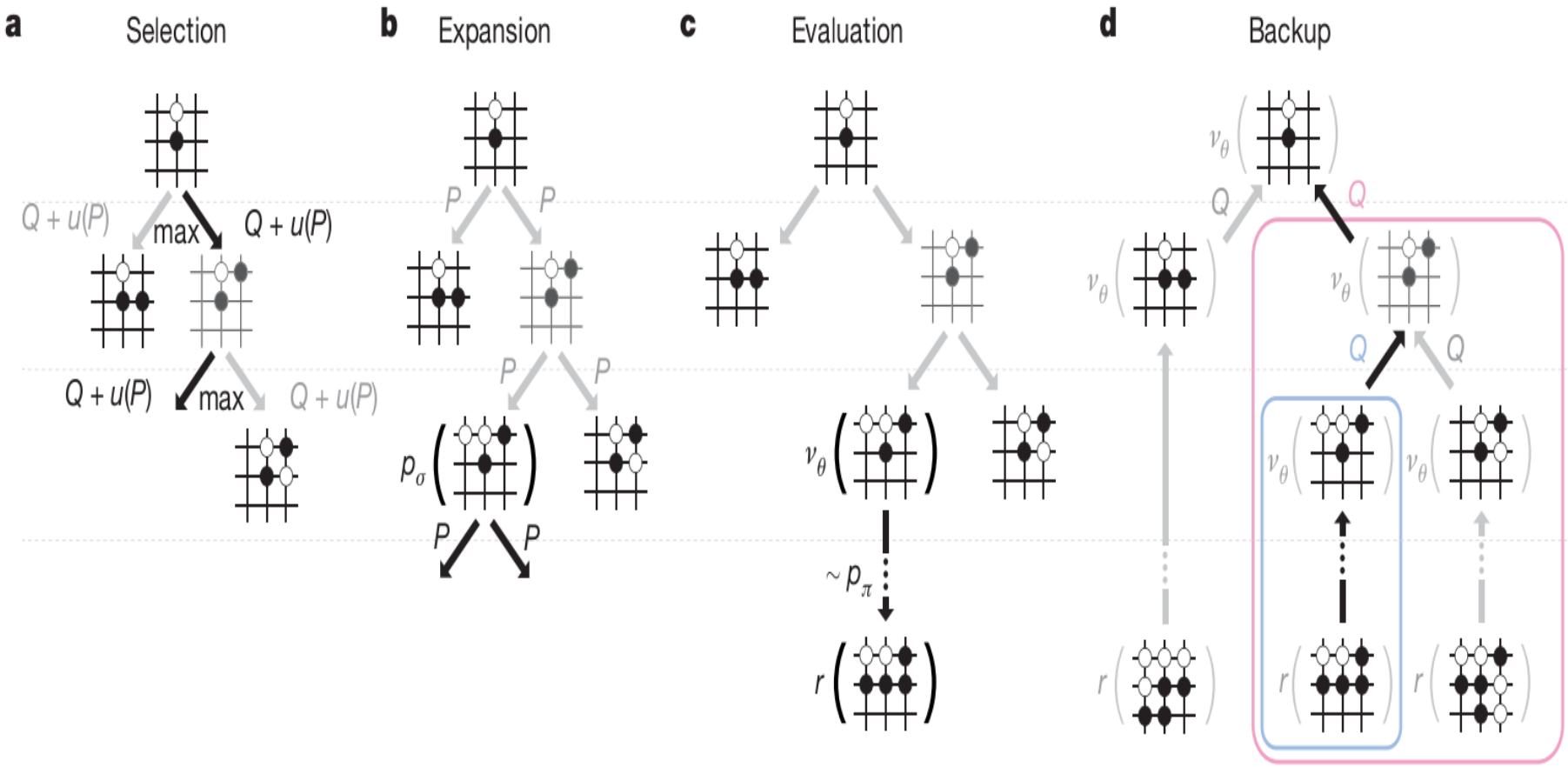
- Имеет схожую архитектуру с policy network.
- Обучается на парах (s, z).
- Используется SGD для минимизации MSE.
- $\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial\theta}(z - v_\theta(s))$

Как бороться с переобучением?

Борьба с переобучением

1. Делается N ходов с помощью SL policy network.
2. Делается случайный легальный ход.
3. Игра доигрывается с помощью RL policy network.
4. Для обучения используется только ход $N + 2$.

Ну а как теперь играть?


a
Selection
Q + u(P)
max
Q + u(P)
Q + u(P)
max
Q + u(P)
b
Expansion
P
P
P
P
P
P
c
Evaluation
~ p_π
d
Backup
Q
Q
Q
Q
Q

Selection

На каждом шаге t выбираем ребро следующим образом:

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

$Q(s, a)$ – action value.
$u(s, a)$ – бонус, поощряющий посещение плохо изученных вершин.

Selection

На каждом шаге t выбираем ребро следующим образом:

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

$Q(s, a)$ – action value.
$u(s, a)$ – бонус, поощряющий посещение плохо изученных вершин.

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

Expansion

- Рассматриваем лист дерева с позицией s.
- Для каждого легального хода a считаем его вероятность с помощью SL policy network $P(s, a) = p_\sigma(a|s)$.

Evaluation

Каждый лист оценивается двумя способами.

1. С помощью value network.
2. С помощью outcome игры, сыгранной с помощью rollout policy.

Evaluation

Каждый лист оценивается двумя способами.

1. С помощью value network.
2. С помощью outcome игры, сыгранной с помощью rollout policy.

Итоговая оценка:

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

Backup

Теперь необходимо обновить action value и количество посещений всех посещённых рёбер.

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

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

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).

Extended Data Table 4 | Input features for rollout and tree policy

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

Features used by the rollout policy (first set) and tree policy (first and second set). Patterns are based on stone colour (black/white/empty) and liberties (1, 2, $\geq$3) at each intersection of the pattern.

Литература

- https://en.wikipedia.org/wiki/Go_(game)
- Mastering the Game of Go with Deep Neural Networks and Tree Search, Google DeepMind
- https://habrahabr.ru/post/279071/