AlphaGo

Игра в го

Цель игры - **захват территории**

Каждый камень должен иметь хотя бы один соседний по вертикали или горизонтали (но не по диагонали!) незанятый пункт. **Захватить камень противника** - лишить точек свободы.

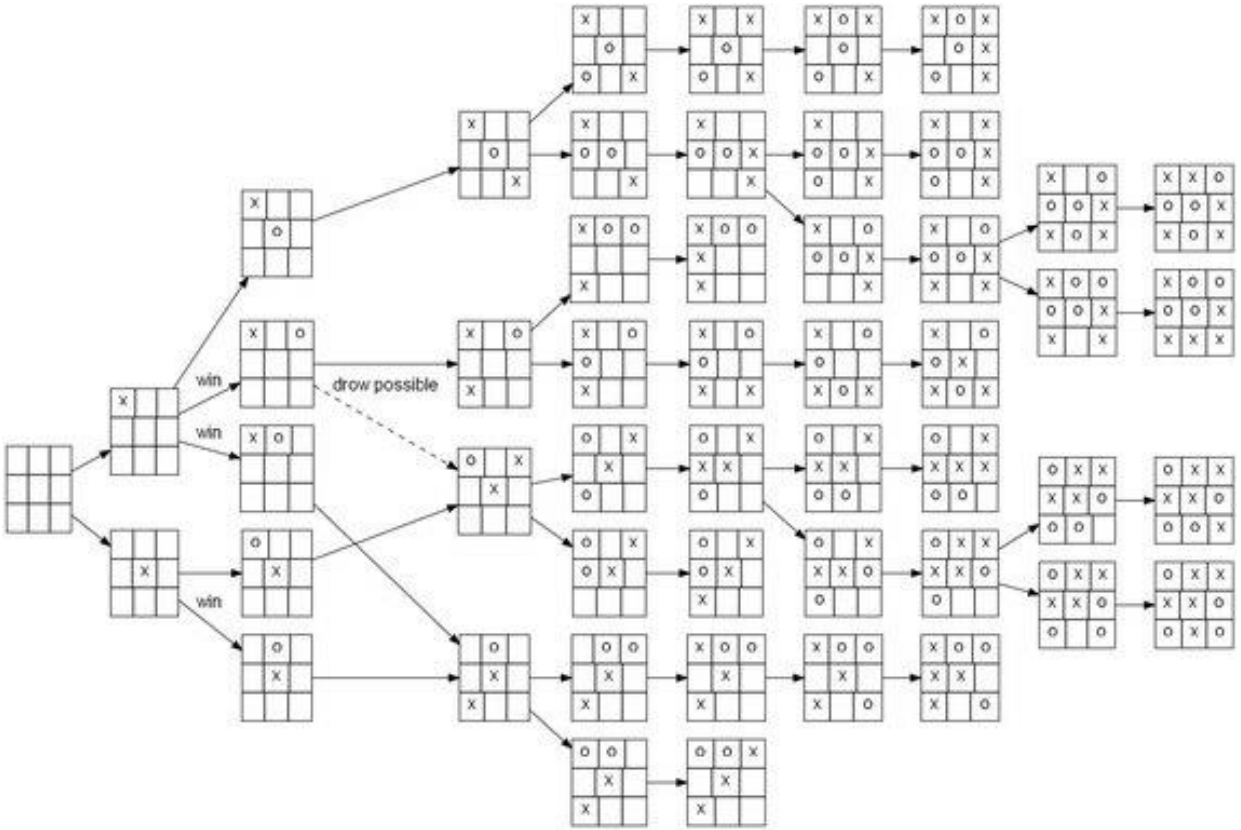
Есть **запрещенные ходы**: например, “убийство” собственного камня.

Когда оба игрока пасуют подряд, игра **заканчивается**.

Оценивание:

- Одно очко за каждый из пунктов доски, окружённых камнями только его цвета
- По одному очку за каждый захваченный камень противника, либо за каждый собственный камень, который остался на доске к концу игры

Дерево исходов



Дерево содержит приблизительно $\mathbf{b^d}$ возможных последовательностей шагов

Шахматы: b = 35, d = 80

Го: b = 250, d = 150

В **1997** **Deep Blue**, шахматный суперкомпьютер**,** выиграл матч из 6 партий у чемпиона мира по шахматам Гарри Каспарова

В **2015** году **AlphaGo** выйграла матч у трёхкратного чемпиона Европы Фань Хуэя.

В **2016** году **AlphaGo** выйграла матч у Ли Седоля. Некоторые источники ставят Ли Седоля четвёртым в мире игроком на время матча

Задача: ограничить дерево

Для решения применяются различные методы сокращения пространства поиска

Альфа-бета-отсечение (*alpha-beta pruning*)

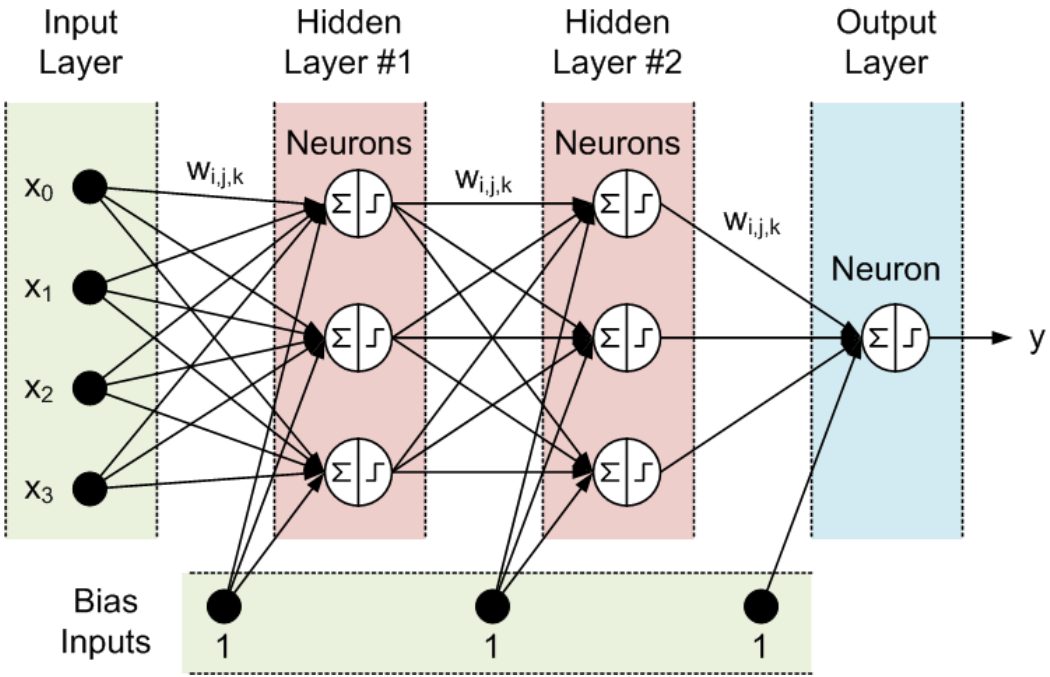
Оценивание ветви дерева поиска может быть досрочно прекращено, если было найдено, что для этой ветви значение оценивающей функции в любом случае хуже, чем вычисленное для предыдущей ветви

Monte Carlo tree search

Анализ наиболее многообещающих ходов

и др!

В AlphaGo используется MCTS

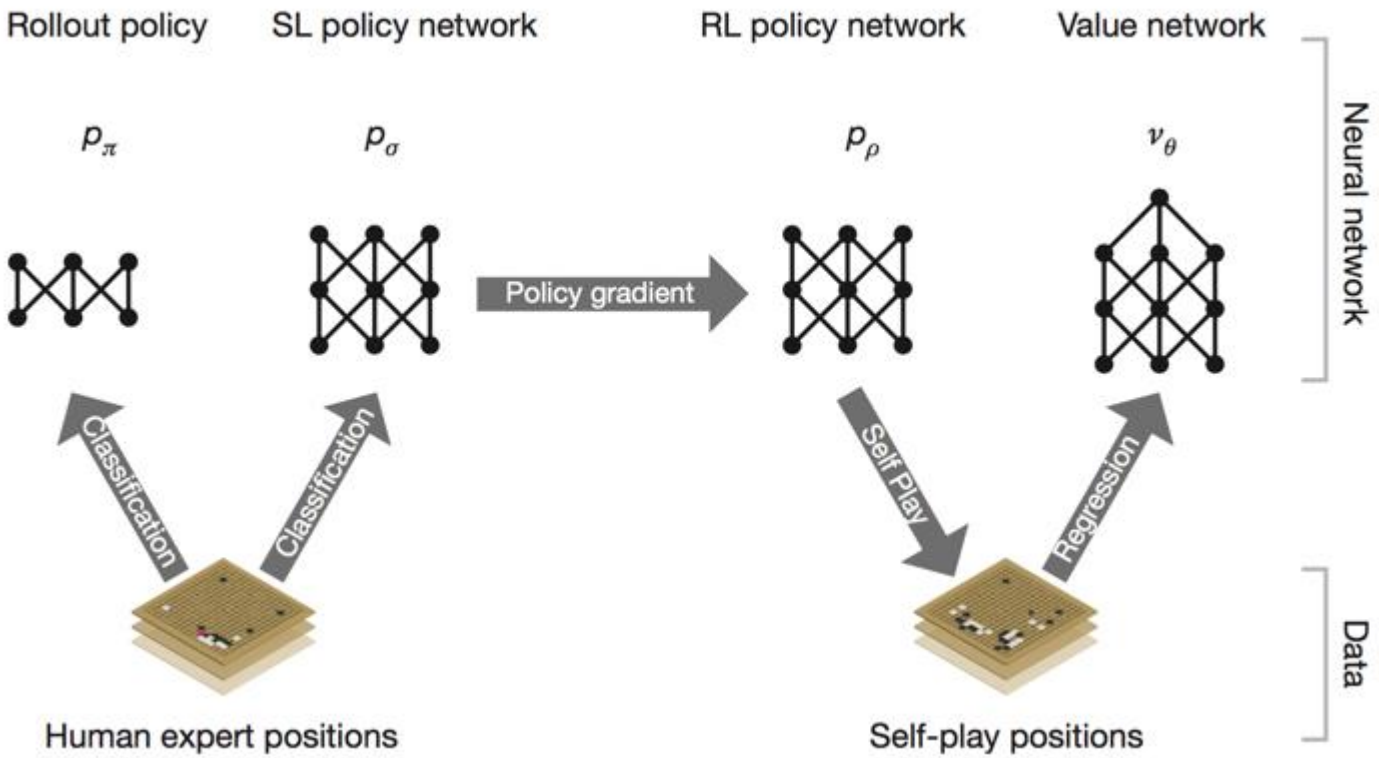


Примеры функций активации:

- Сигмоида $\sigma(x) = \frac{1}{1+e^{-x}}$
- ReLU (rectified linear unit)

$$f(x) = \max(0, x)$$

Сети, используемые в AlphaGo



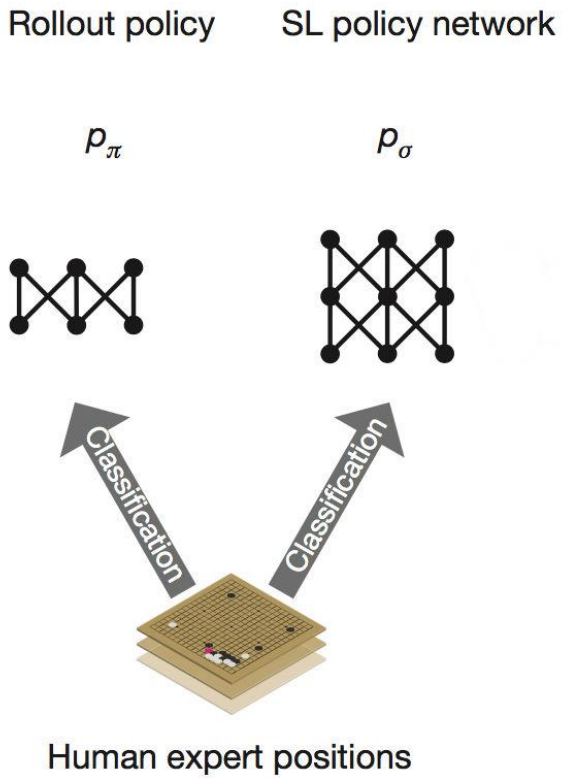
Rollout network, SL policy network

Обучение с учителем

Обучались на KGS, базе ходов профессиональных игроков в го:

- 29.4 миллионов позиций из 160,000 игр
- 6 - 9 профессиональный дан (уровень мастерства, 9 - максимальный)

Оценивают распределение вероятностей доступных ходов, $p_{pi}(s)$ и $p_{sigma}(s)$



Rollout network: признаки и результаты

Feature	# of patterns	Description
Response	1	Whether move matches one or more response 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

1. Очень быстро дает ответ
2. Невысокая точность: 24.2%

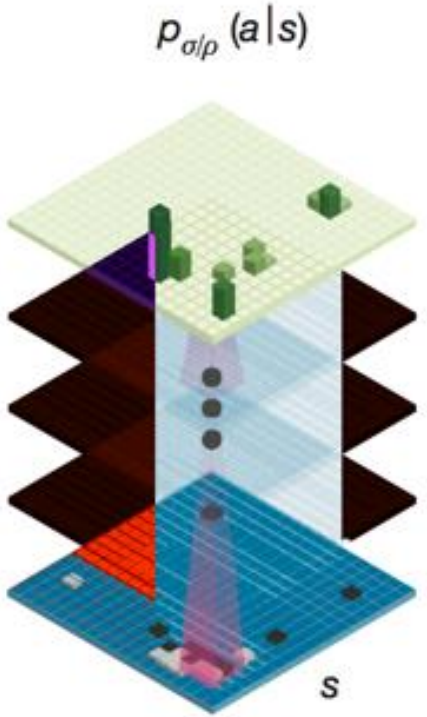
SL policy network: архитектура

сверточная нейронная сеть с 13 скрытыми слоями

вход: 19 x 19 x 48 (48 представлений доски размером 19 x 19)

ReLU nonlinearity

Softmax на выходные значения



SL policy network: features

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

SL policy network

Обучающая выборка ~ 28.4 миллионов позиций

На каждом шаге:

mini-batch из m случайно выбранных объектов, пар $\{s_k, a_k\}$, m = 16

асинхронный стохастический градиентный спуск для максимизации функции правдоподобия

размер шага α=0.03, каждые 80 миллионов шагов уменьшается вдвое

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

SL policy network: результаты

1. Большая точность: 57.0 % на тестовой выборке (~ 1 миллион позиций)
2. Выиграла 11% игр у Pachi
3. Значительно медленнее, чем rollout network

RL-network - обучение с подкреплением

Инициализируется с весами SL-network $\rho = \sigma$

Каждая итерация состоит из n матчей, играемых параллельно, между policy network p_ρ и противником, случайно выбранным из пула значений весов на предыдущих итерациях, $\rho = \rho^-$

Каждые 500 итераций текущее значение ρ добавляется в пул противников

Каждый матч играется до завершения на шаге T^i, $z_t = \pm r(s_t) = \pm 1$

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

RL policy

10,000 mini-batch по 128 игр

Обучение заняло один день

Результаты:

Против SL-policy: выйграно 80% матчей

Против Pachi, самой сильной open-source программы: 85% матчей

Reinforcement learning of value networks

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

z_t - награда (+1 за победу, -1 за проигрыш)

s_t - состояние доски в момент времени t

p – стратегия для ходов $a_{t \dots T}$

Задача: найти оптимальную value function.

Будем использовать лучшую стратегию: RL policy network p_ρ.

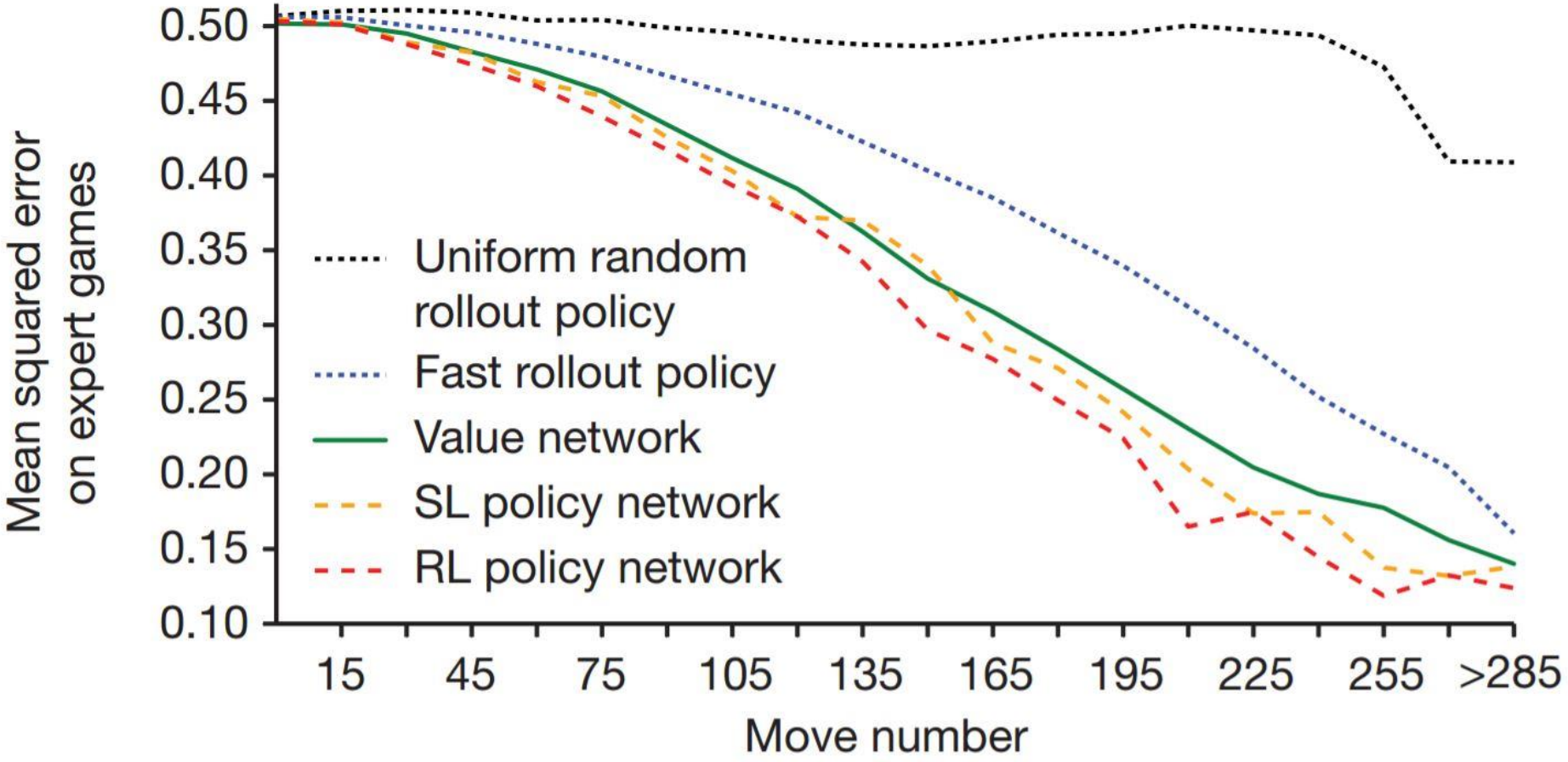
Reinforcement learning of value networks

Будем приближать value function $v_\theta(s)$ с помощью нейронной сети (с весами θ) с такой же структурой, как у SL policy network, только ответом будет одно предсказание, вместо распределения вероятностей.

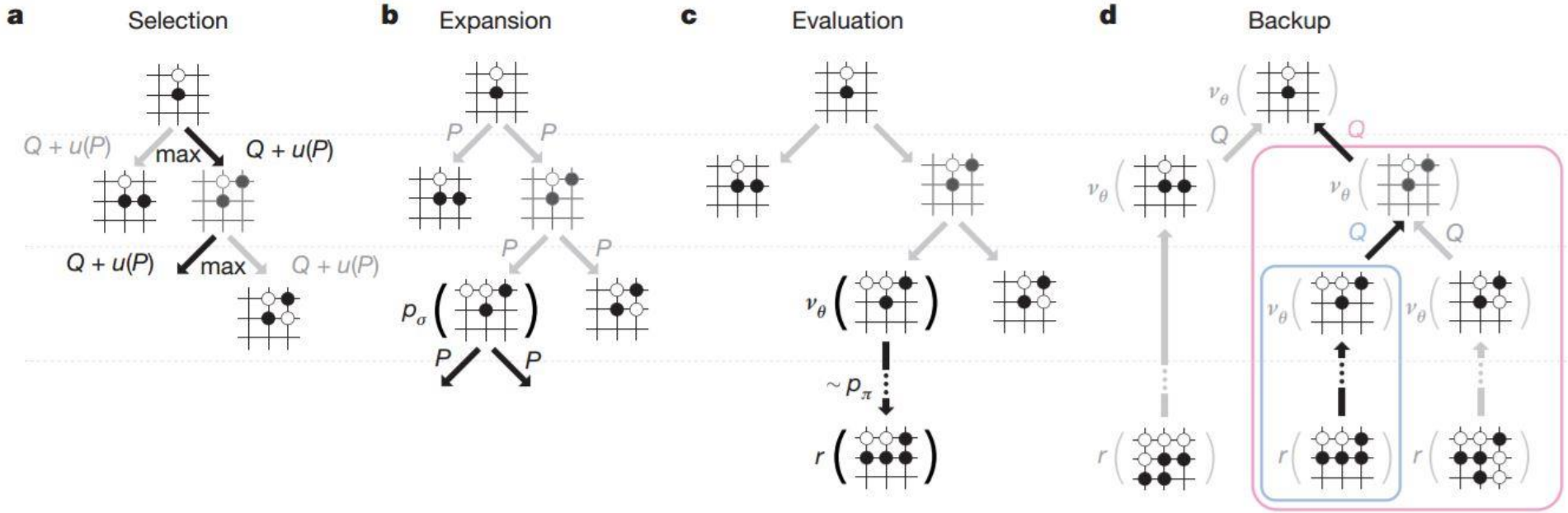
Настроим веса θ, используя стохастический градиентный спуск для минимизации MSE:

$$\min_\theta (z - v_\theta(s))^2$$

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


Mean squared error on expert games
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
Uniform random rollout policy
Fast rollout policy
Value network
SL policy network
RL policy network
15
45
75
105
135
165
195
225
255
>285
Move number

Monte Carlo Tree Search (MCTS)



Monte Carlo Tree Search (MCTS)

Каждый узел (s, a) – (состояние доски, ход) хранит:

- action value $Q(s, a)$
- количество посещений $N(s, a)$
- априорная вероятность $P(s, a)$

Selection: в каждом узле выбираем ход

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

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

Monte Carlo Tree Search (MCTS)

Expansion: при достижении листа применяем SL policy network для предсказания узла s_L

Evaluation:

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

1. value network $v_\theta(s_L)$
2. fast rollout policy p_π

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

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

$1(s,a,i)$ - индикатор того, что посетили узел (s, a) на i-ой симуляции

После окончания поиска алгоритм выбирает ход с наибольшим $N(s,a)$

Список литературы

- Mastering the game of Go with deep neural networks and tree search (http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html) (можно открыть с помощью http://sci-hub.io/)
- BETTER COMPUTER GO PLAYER WITH NEURAL NETWORK AND LONG-TERM PREDICTION (https://arxiv.org/pdf/1511.06410v3.pdf)