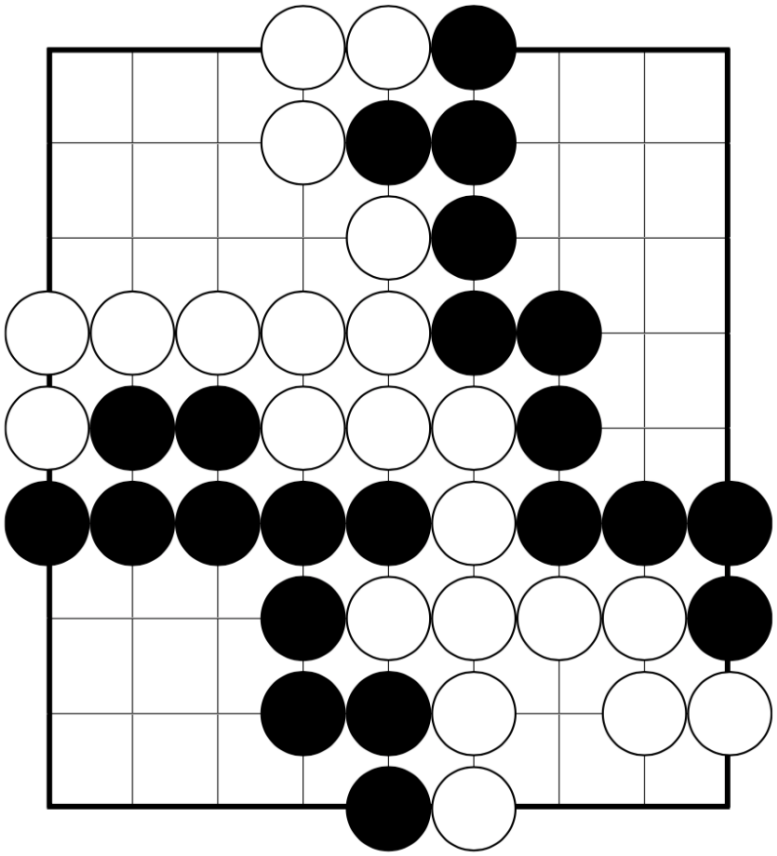
# Reinforcement Learning for GO

Трофимова Юлия

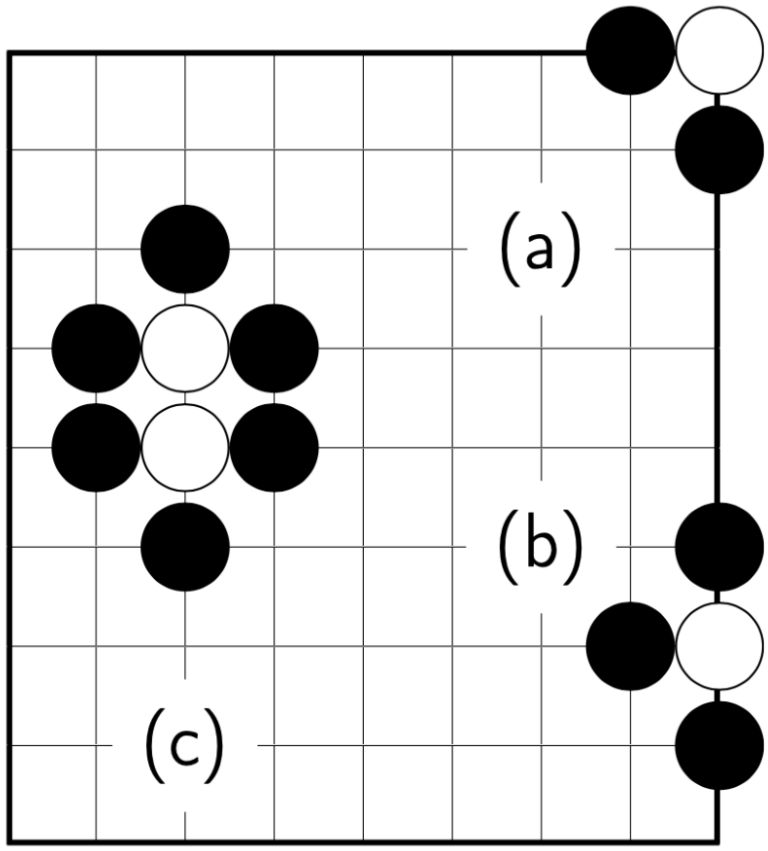
25 января 2019

# Правила игры GO



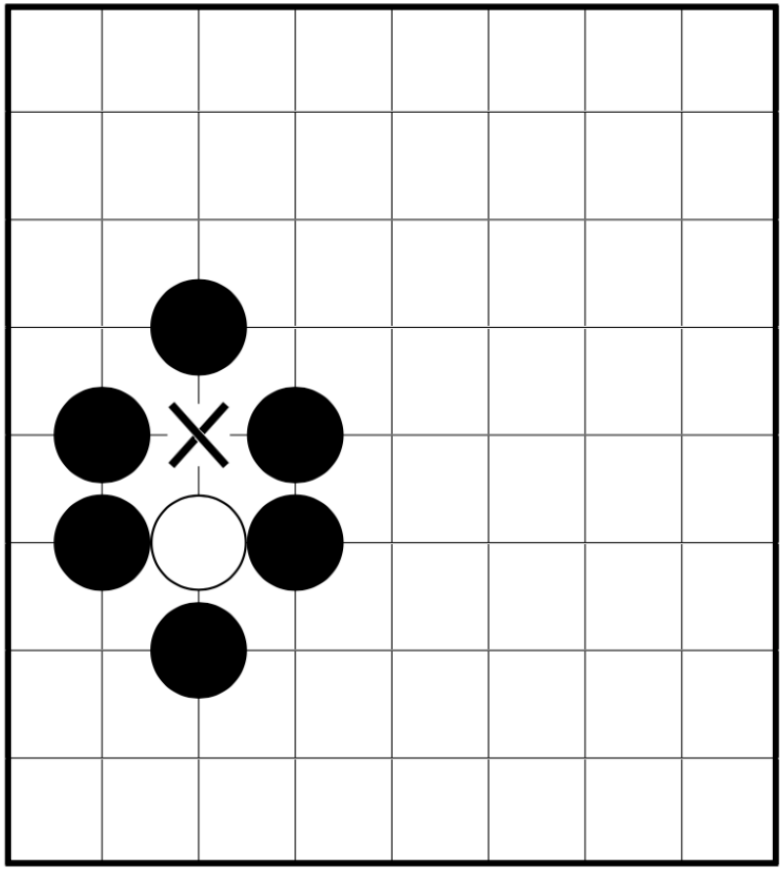
- Игроки по очереди ставят на доску 19x19 камни черного и белого цвета
- Побеждает игрок, окруживший большее количество территории

# Правила игры GO



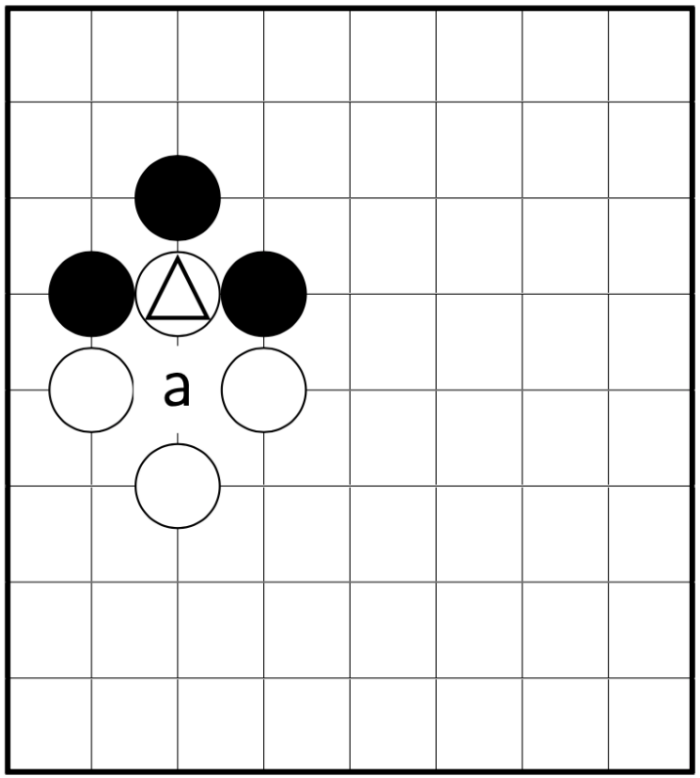

- Во всех трех случаях белые камни удаляются с доски, потому что оказываются захвачены черными.

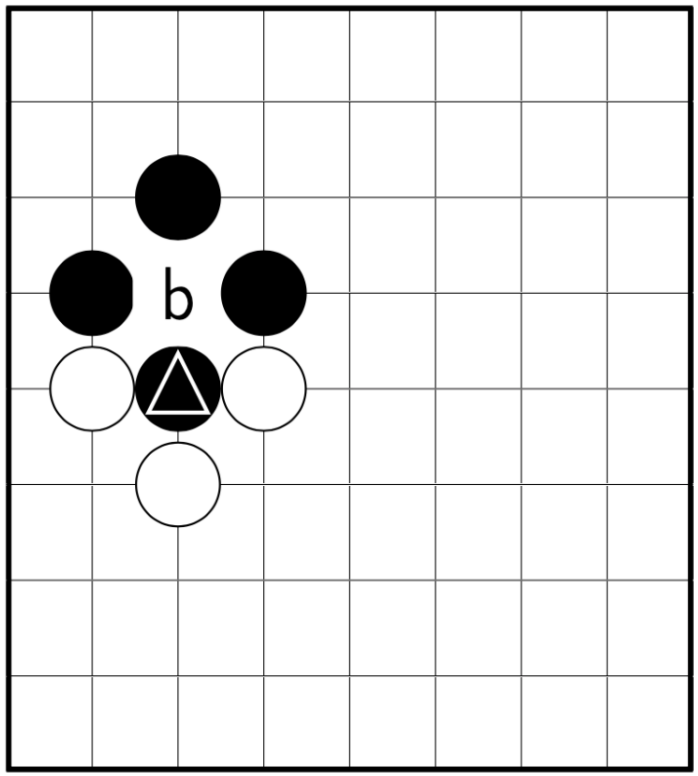
# Правила игры GO



- Нельзя делать ходы, которые явно приведут к исчезновению собственных камней

# Правила игры GO




Черные могут захватить белый камень, если пойдут в точку a.

Но белые не смогут сразу пойти в точку b, но смогут через ход.

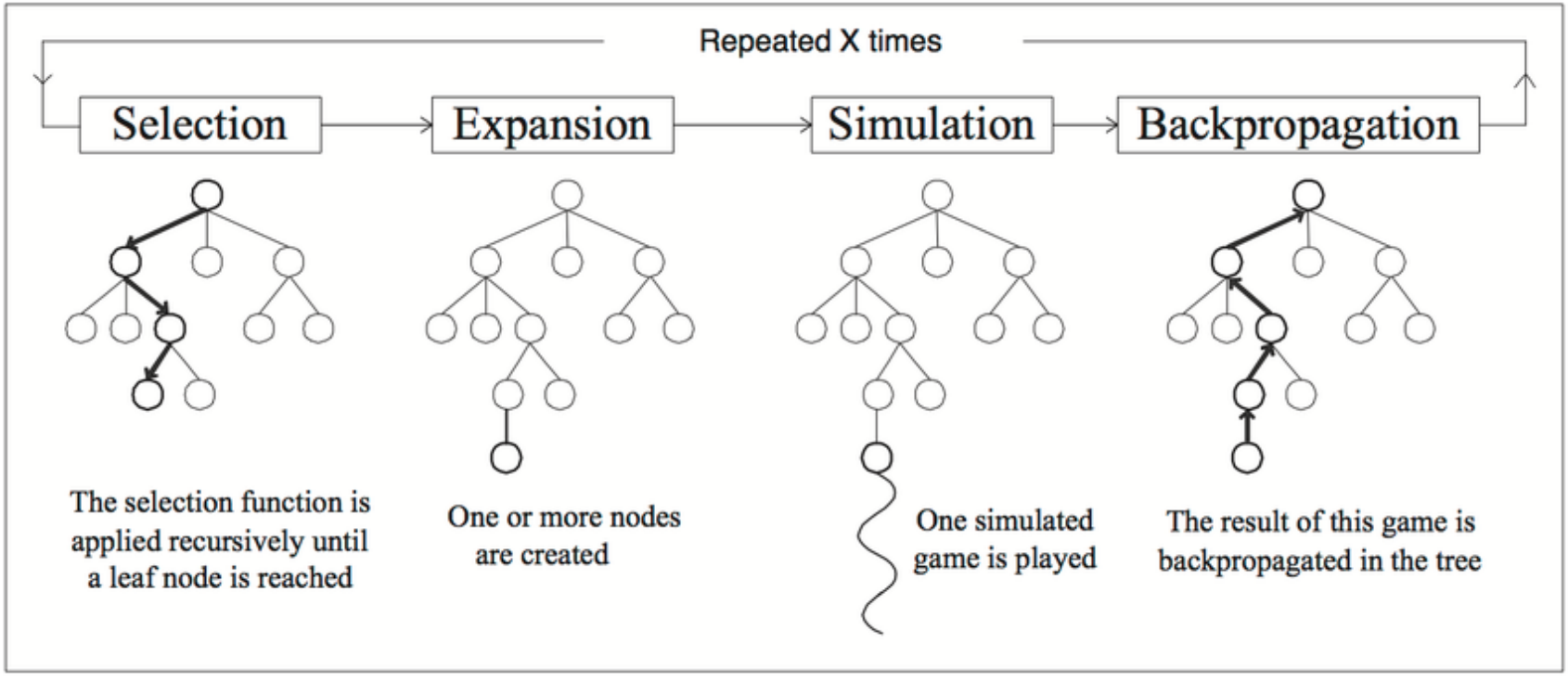
# И почему это сложно?

- 19 x 19 = 361 ходов (из них более 200 допустимы)
- более 100 ходов за игру

Дерево всех состояний получается огромное

# Первые попытки обучения

## Monte Carlo Search Tree



# Выигрышная стратегия

В 2015 году Google DeepMind разработали программу AlphaGo, которая выиграла матч у 3х кратного чемпиона Европы Fan Hui.

Из чего она состоит?

- SL policy network (медленная нейросеть)
- Fast rollout policy
- RL policy network
- Value network

# SL policy network

## Требования

- Нейросеть по состоянию доски предсказывает ход человека
- Точность предсказаний около 57%
- Предсказание занимает 3 ms

# SL policy network

## Структура

- Для каждой клетки на вход подается 48 признаков
- 13 сверточных слоев с нелинейностью и softmax на выходном слое
- Для обучения использовалось 30 миллионов позиций с сервера KGS GO

# SL policy network

## Обучение

- Используем стохастический градиентный спуск
- Обучаемся на парах (s, a) - (состояние доски, действие человека)

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

# Fast rollout policy

- Принцип работы как у SL policy
- Точность предсказаний до 24.2%
- Предсказание занимает 2 микросекунды
- На вход подается очень много признаков

# RL policy

## Структура

- Сыграем несколько раундов между текущей сетью и рандомной сетью из предыдущих итераций

# RL policy

## Обучение

- На каждой итерации сыграем несколько раундов между текущей сетью и рандомной сетью с предыдущих итераций.
- $z_t$ задает направление градиента: +1 если выиграли, -1 если проиграли
- обновляем веса всех ребер, которые входили в игру

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

# Value network

## Структура

- Эта нейросеть по состоянию доски оценивает value function $v^p(s)$ - вероятность выигрыша (с учетом того, что оба игрока используют политику p)

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

$$v_\theta(s) \approx v^{p_\rho}(s) \approx v^*(s)$$

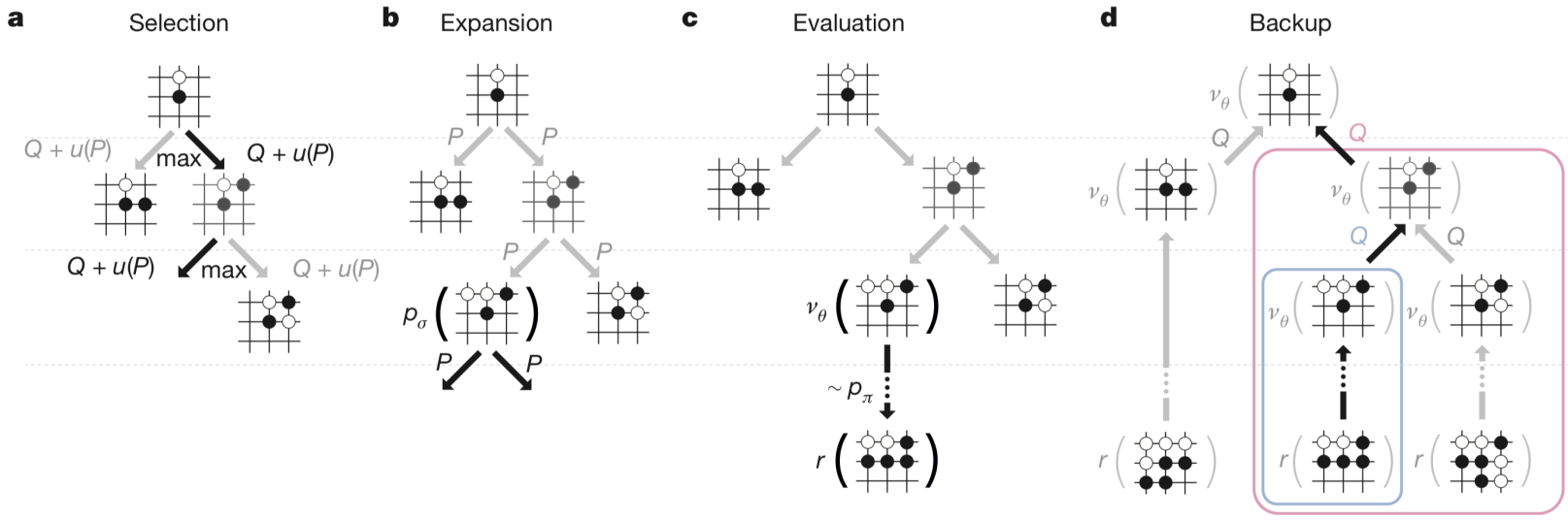
# Value network

## Обучение

- Архитектура схожа с policy network
- Обучается на парах (s, z) - (состояние доски, исход игры)
- Обучает SGD с минимизацией MSE.

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

# Процесс игры



# Selection

На каждом шаге t выбираем ребро следующим образом

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

$Q(s_t, a)$ - action value
$u(s_t, a)$ - добавка, отвечающая за exploration

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

# Expansion

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

- С помощью SL policy считаем вероятности всех легальных ходов $P(s,a) = p_\sigma(a|s)$

# Evaluation

## Каждый лист оцениваем двумя способами

- С помощью value network $v_\theta(s_L)$
- С помощью игры fast rollout policy $z_L$

## Итоговая оценка листа

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

# Backup

Обновляем количество посещений и action value для всех посещенных ребер

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

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

# Как AlphaGo начал побеждать

| Версии | Аппаратное обеспечение[25] | Рейтинг Эло* | Матчи |
|---|---|---|---|
| **AlphaGo Fan** | 176 GPU,[26] распределенные вычисления | 3144[27] | 5:0 Матч AlphaGo — Фань Хуэй |
| **AlphaGo Lee** | 48 TPU,[26] распределенные вычисления | 3739[27] | 4:1 Матч AlphaGo — Ли Седоль |
| **AlphaGo Master** | 4 TPU[26] v2, одна машина | 4858[27] | 60:0 против профессиональных игроков го;<br>Саммит «Будущее Го»;<br>3:0 Матч AlphaGo — Кэ Цзе |
| **AlphaGo Zero** | 4 TPU[26] v2, одна машина | 5185[27] | 100:0 против AlphaGo Lee<br>89:11 против AlphaGo Master |
| AlphaZero | 4 TPU v2, одна машина | N/A | 60:40 против AlphaGo Zero |

Для сравнения рейтинг Эло лучшего человека игрока в го Кэ Цзе на октябрь 2017 года составлял 3670 пунктов

# AlphaGo Zero

Если компьютеры уже победили, зачем AlphaGo Zero?

# AlphaGo Zero

## Если компьютеры уже победили, зачем AlphaGo Zero?

У AI-сообщества были претензии

- Для обучения AlphaGo использовались игры людей
- Очень много вручную подобранных признаков
- Нужен большой кластер, чтобы все это запустить

# AlphaGo Zero

## Если компьютеры уже победили, зачем AlphaGo Zero?

У AI-сообщества были претензии

- Для обучения AlphaGo использовались игры людей
- Очень много заинженеренных фич
- Нужен большой кластер, чтобы все это запустить

Поэтому DeepMind в конце 2017 представили версию AlphaGo Zero, которая обучалась с нуля, исключительно на играх с самой собой, использовала случайные веса в качестве стартовых и в качестве фич использовала только положение камней на доске. А еще было сильно проще по требованиям к железу.

# AlphaGo Zero

## Структура

- Тренируем нейросеть, объединяющую policy network и value network
- На вход поступает текущее состояние доски и несколько предыдущих, а на выходе она дает вероятность следующих ходов и value function
- Сотоит из множества остаточных блоков сверточных слоев и batch normalization
- Обучается на играх с самой собой

# AlphaGo Zero

## Обучение

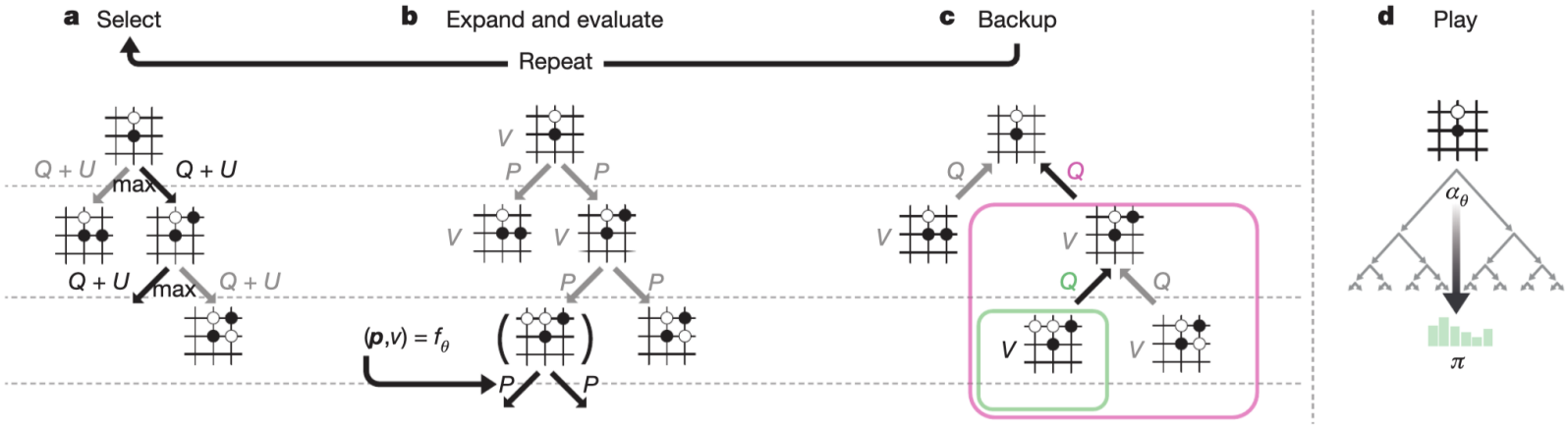
В каждой ноде дерева состояний хранится четыре значения — N (сколько раз мы ходили по этой ноде), V (value этой ноды), Q (усреднённое value всех дочерних нод этой ноды) и P (вероятность, что из всех допустимых на данном ходу нод мы выберем именно эту)

# AlphaGo Zero

## Обучение

- Спускаемся по дереву, выбрая ребро с наибольшим Q + U (как и ранее)
- Когда доходим до неисследованной ноды, подаем ее на вход нейросети.
- Полученное value записываем в ноду
- Создаёт дочерние ноды с P согласно полученным вероятностям и нулевыми N, V и Q.
- Обновляет все ноды выше текущей, которые были выбраны во время симуляции, следующим образом: N := N + 1; V := V + v; Q := V / N.

# AlphaGo Zero



# AlphaGo Zero

## Ход, который сделает сеть

- Если это игра, то выбирается вершина с наибольшим N
- Если обучение, выбираем ход их распределения $\pi_a = N(s,a)^{1/t}$ где t - некоторая температура для контроля exploration

В данном алгоритме мы для каждой ноды всего раз запускаем нейросеть, а rollout вообще не делаем, считая, что предсказанный результат итак достаточно точен.

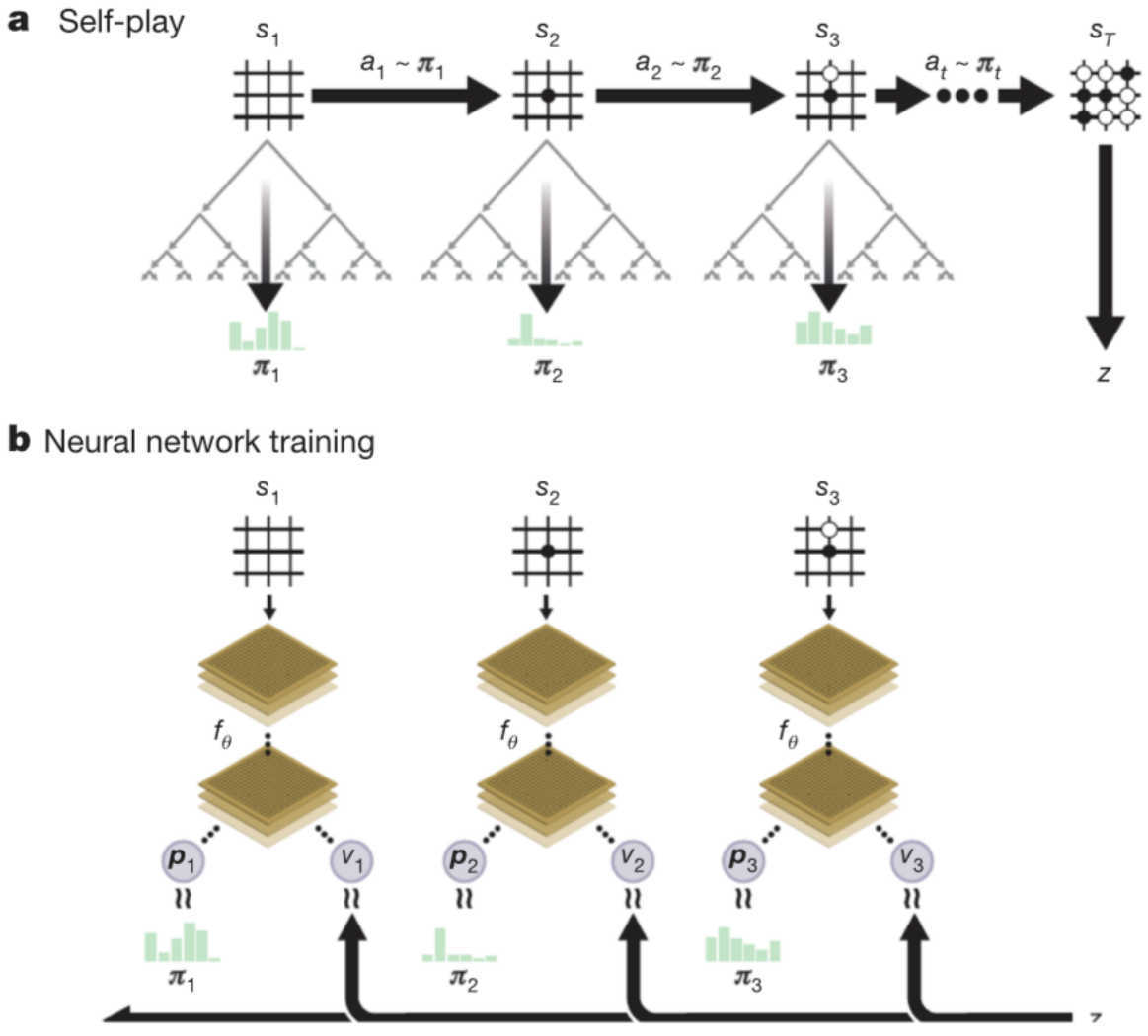
# AlphaGo Zero

## Обучение нейросети

- Обучаемся на пулах $(s_t, \pi_t, z_t)$
- Минимизируем MSE между предсказанием сети и результатом игры.
- Минимизируем разницу между предсказаниями сети p и насемплированными вероятностями ходов
- L2 регуляризация

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

# AlphaGo Zero



# AlphaGo Zero

## Обучение нейросети

- Есть наилучшая сеть с весами A
- Проводим 25000 игр A с самой собой. Сохраняем каждый ход как $(s_t, \pi_t, z_t)$
- Готовим батчи из 2048 случайных позиций из последних 500 000 игр, отдаём 1000 таких батчей на тренировку и получаем некоторую новую сеть с весами B
- Играем 400 раундов A vs B
- Если B выигрывает более чем в 55%, она становится лучшей сетью

# AlphaGo Zero