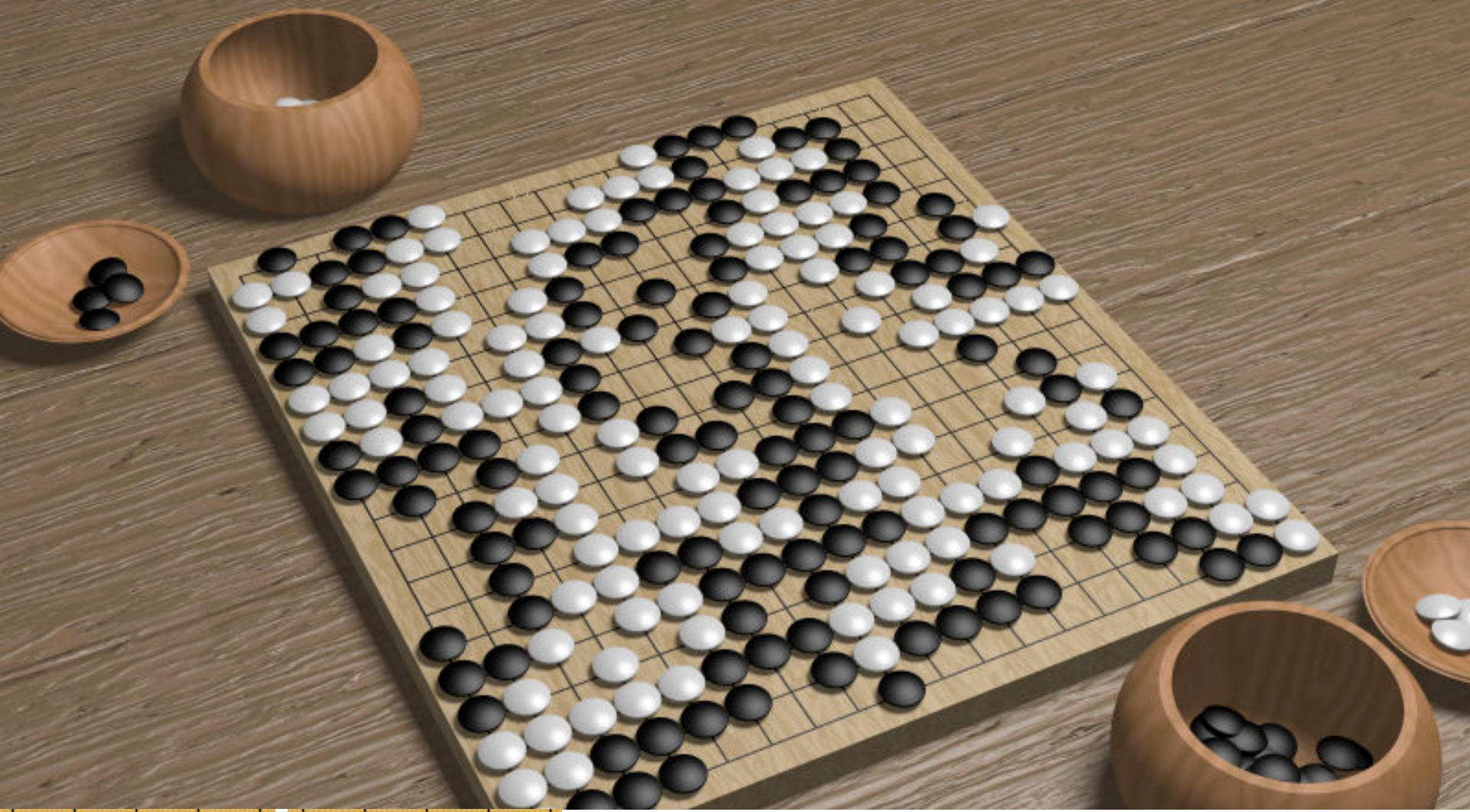
Как компьютер научили играть в го

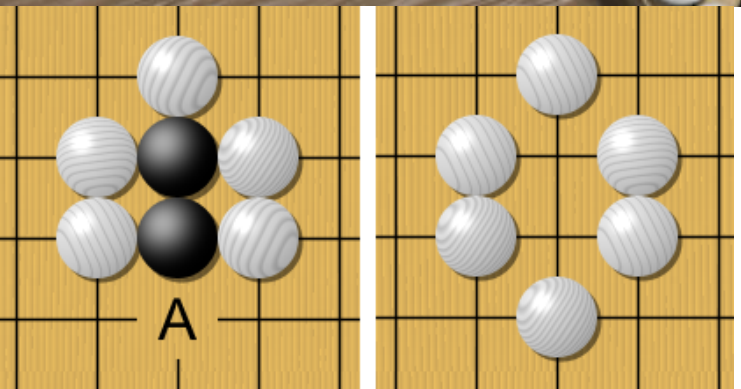
А.С. Трушечкин

Коллоквиум МИАН

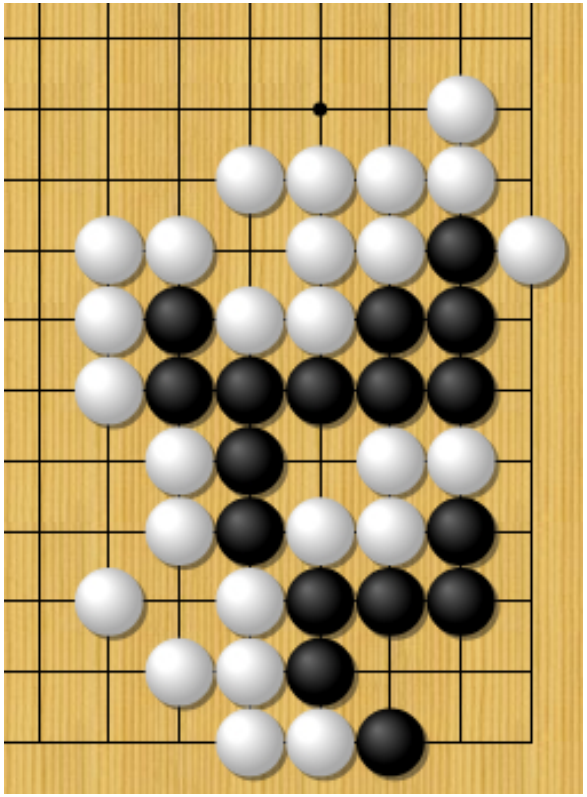
11 мая 2017 г.

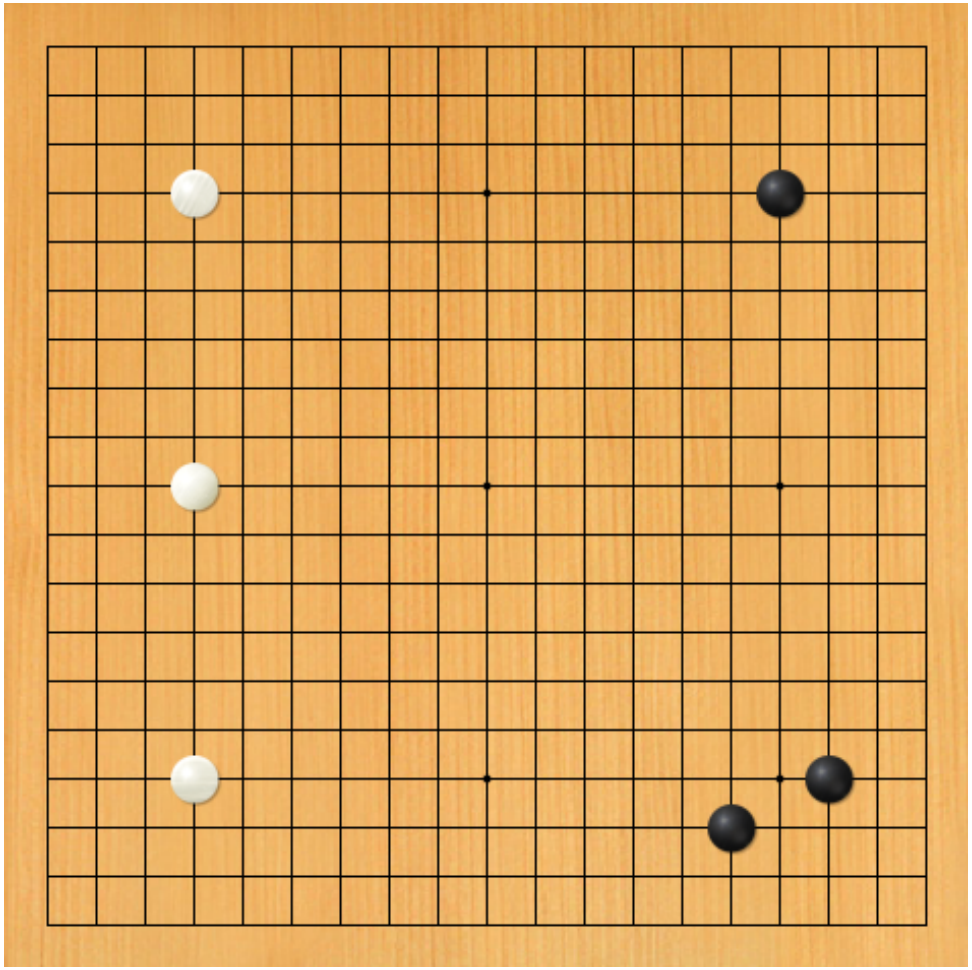
- Март 2016: победа программы AlphaGo (Google DeepMind, Лондон) в матче по го против одного из сильнейших гоистов мира Ли Седоля (4:1)
  - Октябрь 2015: победа AlphaGo в матче против трёхкратного чемпиона Европы Фань Хуэя (5:0)
  - Январь 2017: «таинственный игрок» Master на онлайн-площадках: 60 побед из 60 против сильнейших игроков мира
- 1997: победа компьютера DeepBlue (IBM) в матче по шахматам против Гарри Каспарова (3,5:2,5)




A

Расчёт vs Интуиция





Александр Динерштейн, 7-кратный чемпион Европы по го:

«Для меня это остаётся большим вопросом – как будет действовать программа, если с первых же ходов свернуть с дебютных справочников. На пустой доске вариантов столько, что никаким методом Монте-Карло их не просчитать. В этом го выгодно отличается от шахмат. В шахматах всё давно изучено на глубину 20-30 ходов, а в го, при желании, уже первым ходом можно создать позицию, которая не встречалась в истории профессионального го. Программе придётся играть самостоятельно, а не вытаскивать варианты из базы знаний. Посмотрим, сможет ли она это сделать. Я в этом сильно сомневаюсь и ставлю на Ли Седоля».

Стоя на плечах гигантов…

- Первая программа игры в го –
  1960
- Непосредственно перед AlphaGo: программы на уровне мастеров-любителей
- Но был большой разрыв в силе игры с профессионалами
- А. Динерштейн: «Я был уверен, что у нас есть хотя бы 10 лет в запасе. Ещё пару месяцев назад мы играли с программами на форе в четыре камня – это примерно как фора в ладью в шахматах. И тут – бац! – и сразу Ли Седоль повержен».

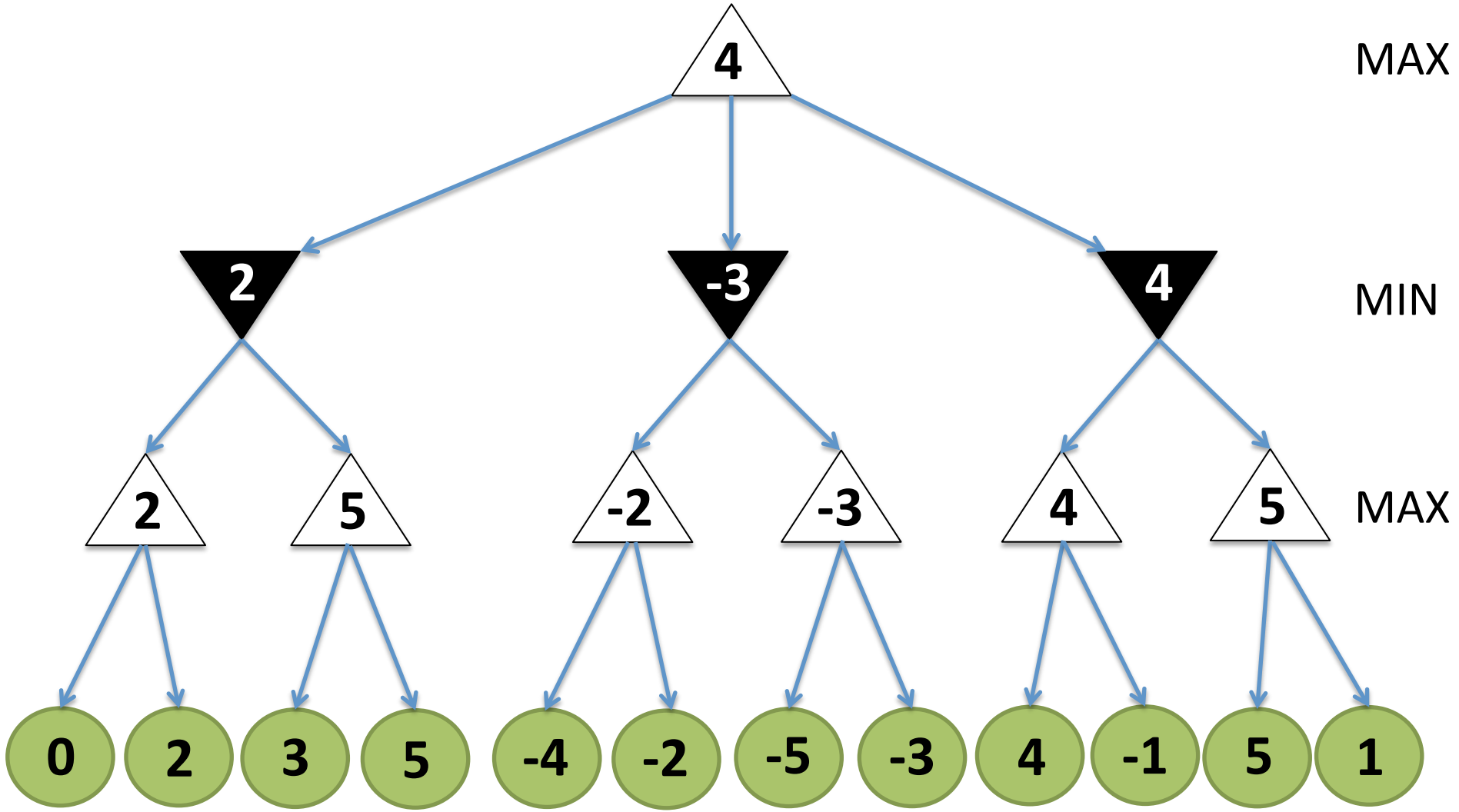
План

- Метод альфа-бета-отсечения
- Метод Монте-Карло для поиска в дереве
- Машинное обучение, свёрточные нейронные сети как подзадачи в методе Монте-Карло

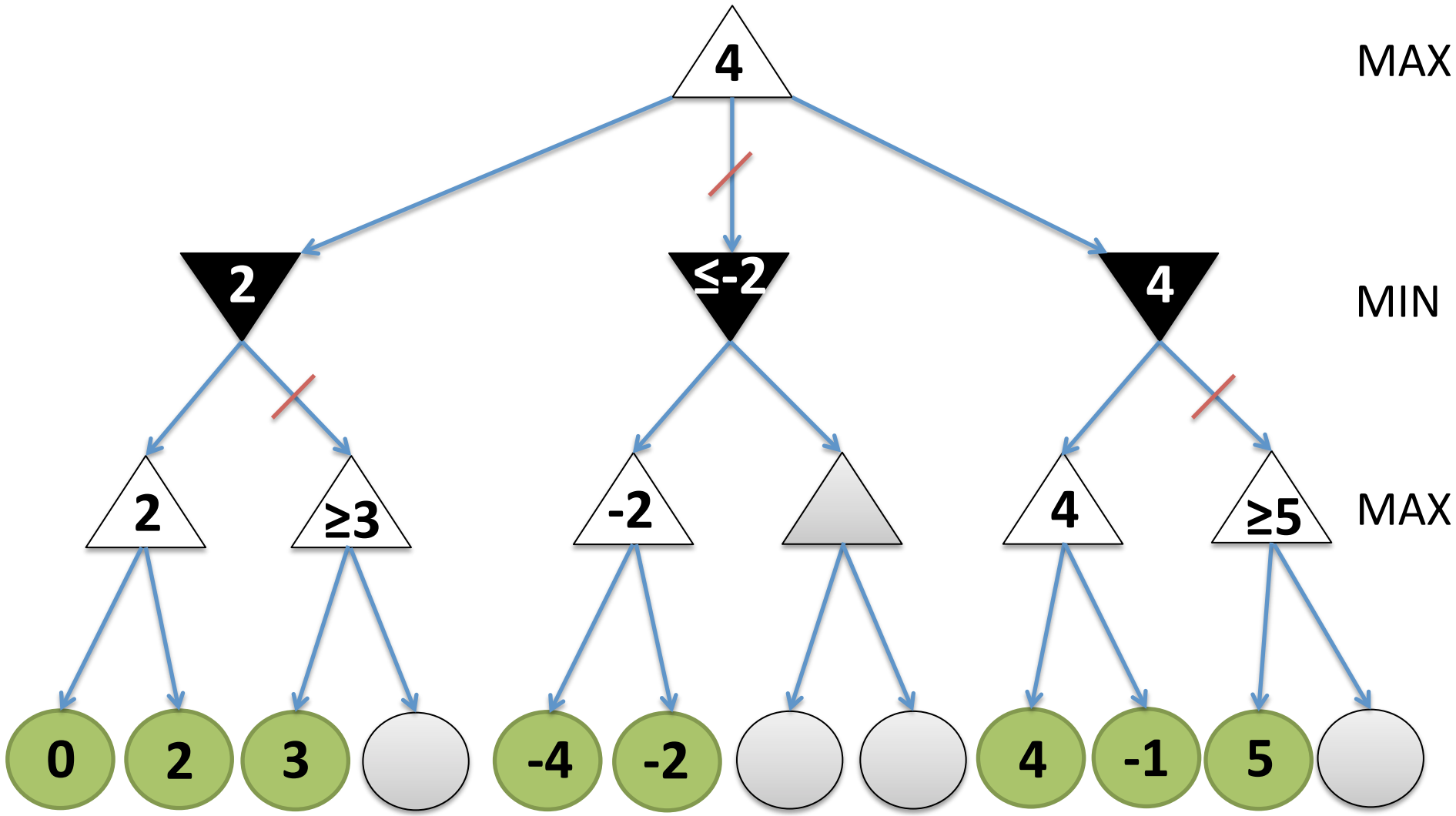
Шашки, шахматы, го – это игры со следующими свойствами:

- С полной информацией
- Антагонистические (с нулевой суммой)
- Конечные

Дерево игры



Альфа-бета-отсечение



Вариативность шахмат и го

- Число всевозможных партий ≈ b^d
  - b – средний коэффициент ветвления (число возможных ходов в фиксированной позиции)
  - d – средняя глубина дерева (длина игры)
- Шахматы: b≈35, d≈80, $b^d \approx 10^{124}$
- Го: b≈250, d≈150, $b^d \approx 10^{544}$
- Число атомов во Вселенной: 10^{80}

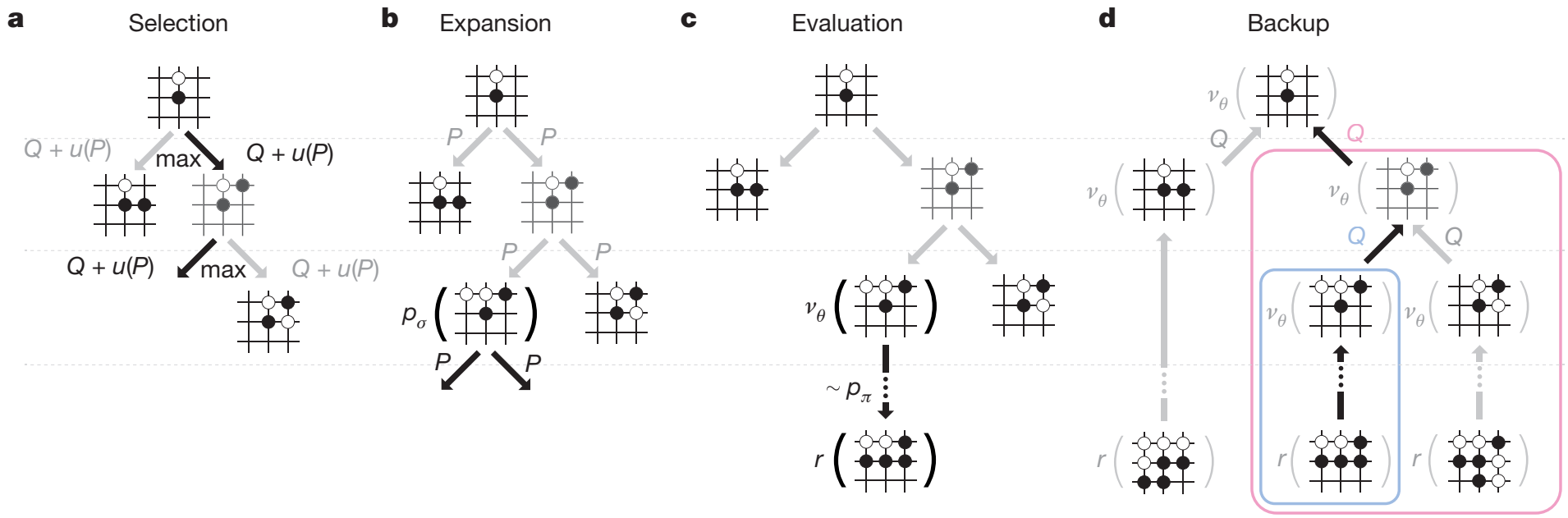
Применение альфа-бета-отсечения

- Итак, каждая вершина (состояние игры) s имеет свою ценность $v^*(s)$ – выигрыш одного из игроков при оптимальной игре обоих
- Вместо истинного значения – оценка $v(s)$
  - Пример в шахматах: оценка материала (пешка – 1 очко, конь и слон – по 3 очка и т.д.)
  - В DeepBlue – сложная экспертная формула оценки позиции (8000 параметров)
- Итеративное углубление
  - В DeepBlue: углубление на от 6 до 16, в отдельных случаях – до 40 уровней
- **Проблема:** в го, помимо большего объёма перебора, сложно формализовать качество позиции

В центре AlphaGo – метод Монте-Карло для поиска в дереве

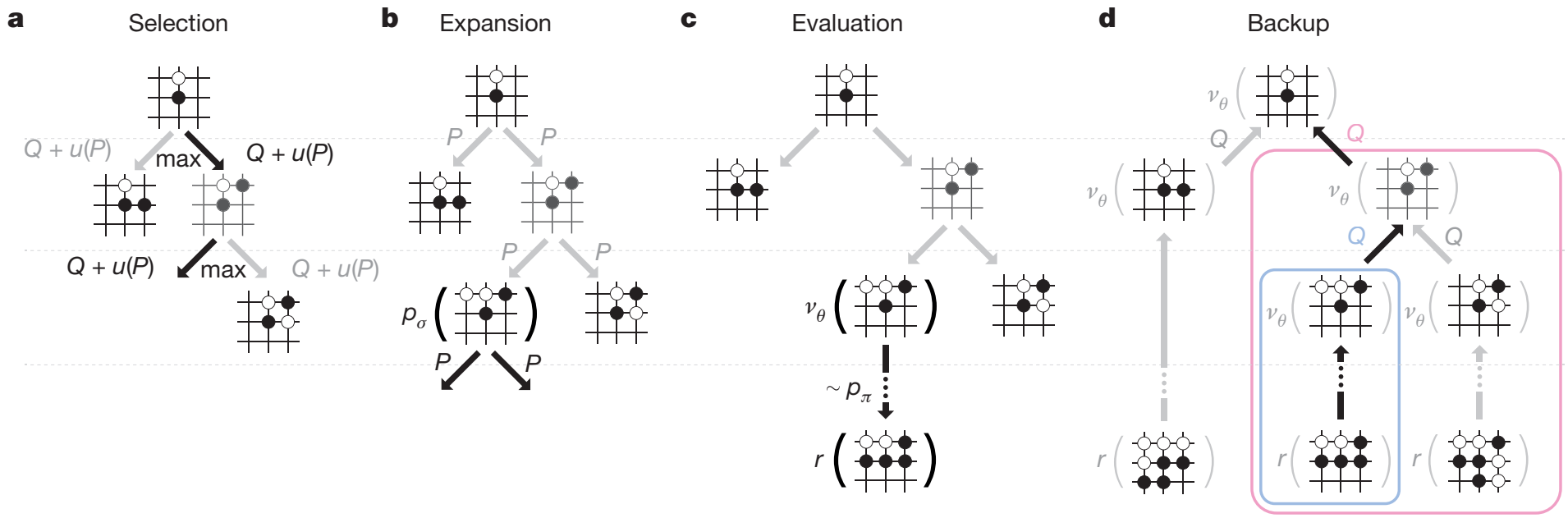
- Проблемы:
  - сложно формализовать качество позиции
  - большой объёма перебора
- Для оценки позиции – **метод Монте-Карло**: сыграть большое количество случайных партий, начиная с позиции *s*. Если в большинстве случае побеждают, например, белые, то позиция более выигрышна для белых.
- Для снижения объёма перебора – не просто метод Монте-Карло для оценки позиций, а эвристика для эффективного поиска – **метод Монте-Карло для поиска в дереве**

Метод Монте-Карло для поиска в дереве



- Дерево формируется итеративно
- Начальное дерево – текущая позиция и её непосредственные потомки (один ход)
- На каждой *i*-й итерации дерево проходится от корня до некоторого листа s_L^i, для которого вычисляется оценка (случайная игра) $V(s_L^i)$

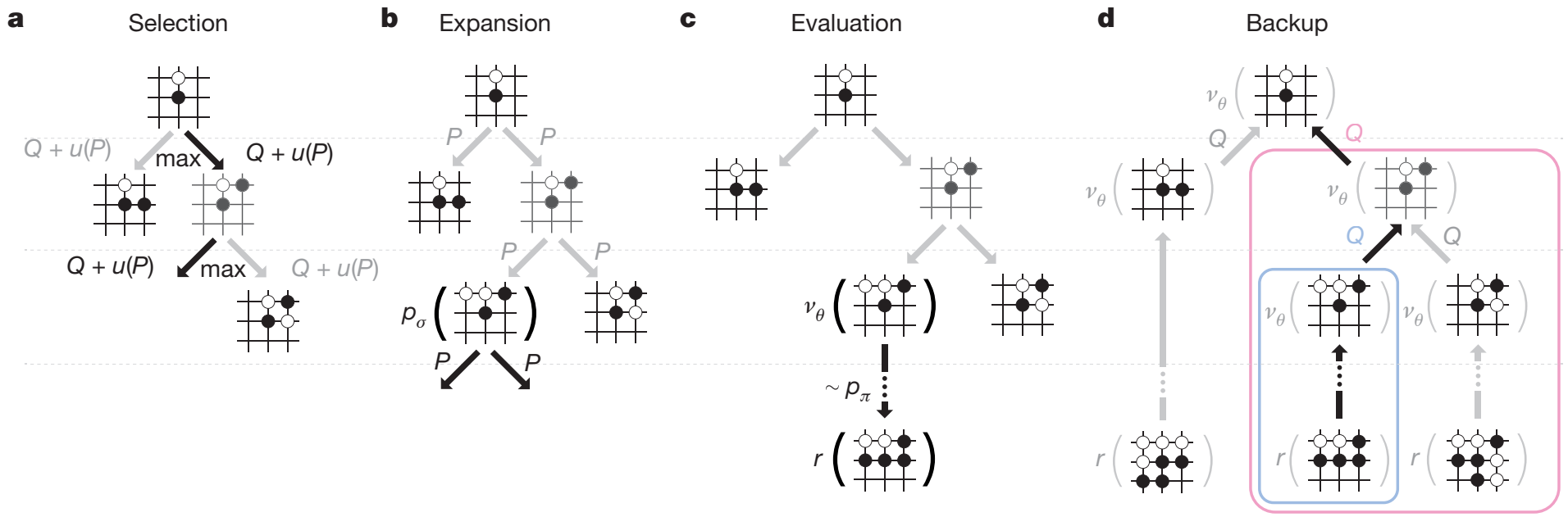
Метод Монте-Карло для поиска в дереве



- Пусть сделано *n* итераций. В результате сформировано некоторое дерево. Каждое ребро (*s*,*a*) хранит переменные:
  - Априорная вероятность *P*(*s*,*a*) **(запомним!)**
  - Счётчик посещений $N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$
  - Среднее качество $Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$

1(*s*,*a*,*i*) – прошла ли *i*-я итерация через ребро (*s*,*a*)

Метод Монте-Карло для поиска в дереве



- Правило прохода по дереву: следующий ход из позиции s_t выбирается по правилу

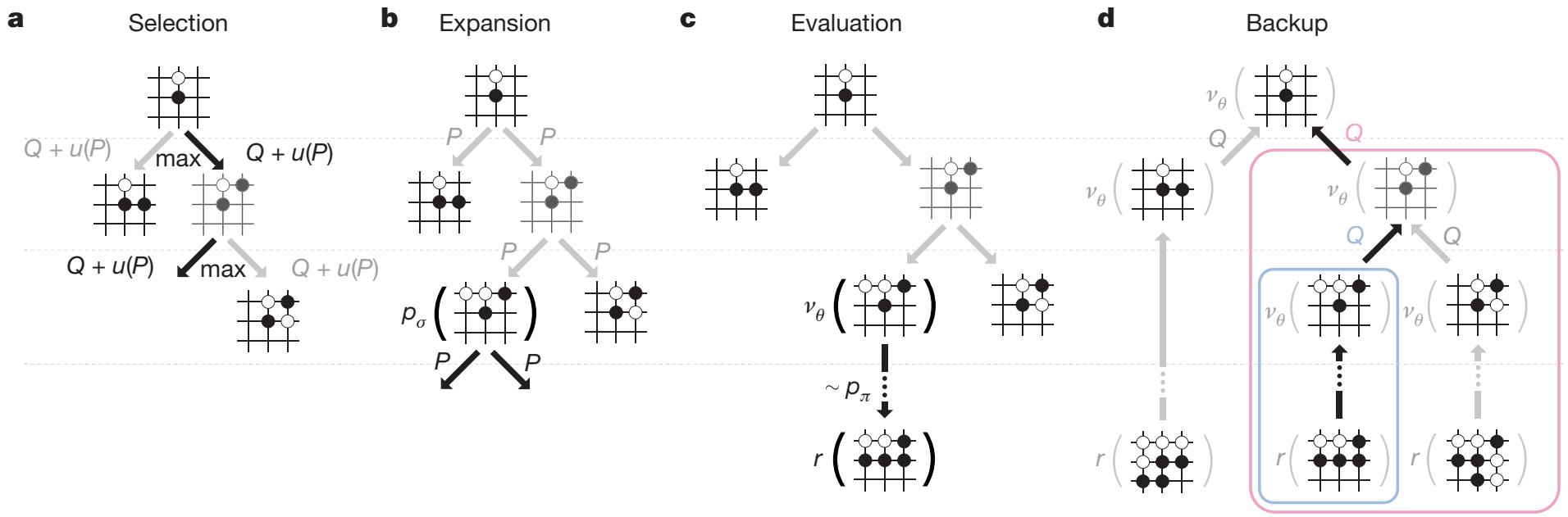
$$a_t = \arg\max_a (Q(s_t,a) + u(s_t,a)), \quad u(s_t,a) \sim \frac{P(s_t,a)}{1+N(s_t,a)}$$

Среднее качество

Бонус

- Правило разветвления. Пусть ребро (s,a) ведёт к листу s_L. При превышении N(s,a) определённого порога (40), лист s_L разветвляется

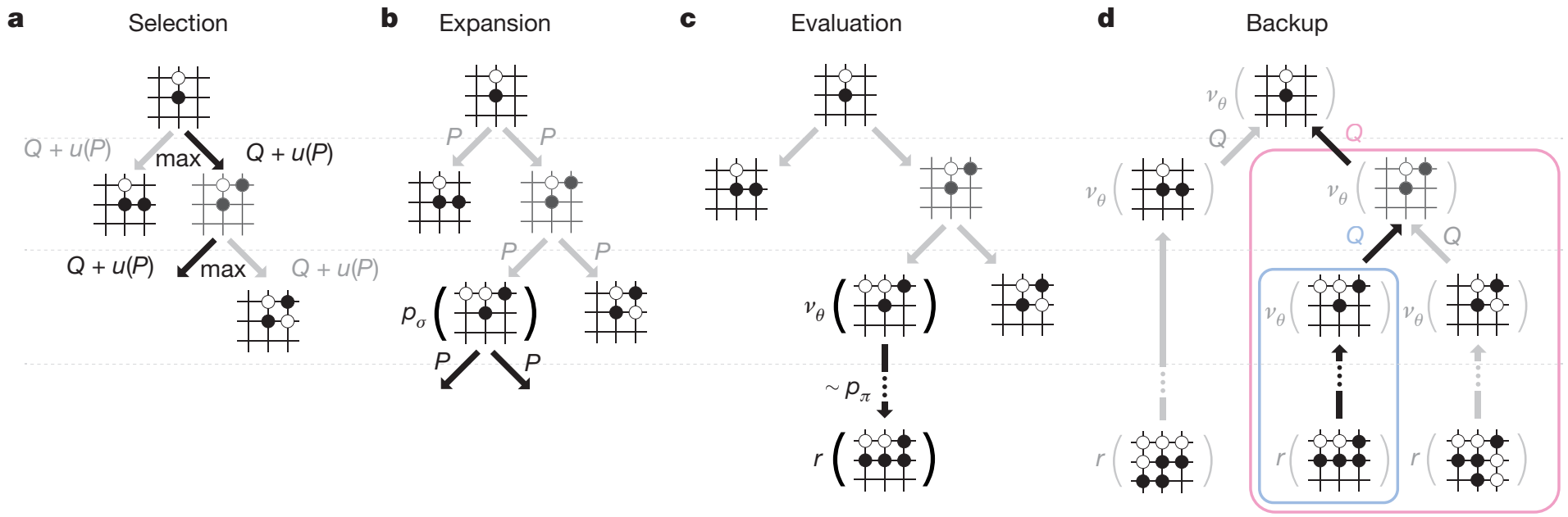
Метод Монте-Карло для поиска в дереве



1) Выбор (пути до листовой вершины)
2) Разветвление (если необходимо)
3) Оценивание (случайная игра)
4) Обратное распространение (обновление переменных у рёбер)

После всех итераций выбирается ход a с max $N(S,a)$

Метод Монте-Карло для поиска в дереве



Для хорошей работы метода необходимо:

- Хорошая функция априорной вероятности $P(s,a)$ для начального отбора ходов-кандидатов
- Хорошая стратегия для случайной игры для адекватной оценки позиции

Для обеих задач используются нейронные сети

Применения метода Монте-Карло для поиска в дереве

- Игры
- Планирование
- Составление расписаний
- Задача удовлетворения ограничений (выполнимость булевых функций, раскраска графа, раскраска карты)

Машинное обучение

«Область исследований, изучающая возможность компьютеров обучаться без непосредственного программирования» (Артур Сэмюэль)

Пример: задача классификации

- *X* – множество описаний объектов
- *Y* – конечное множество классов
- Существует неизвестная функция $f : X \to Y$, значения которой известны только на объектах обучающей выборки $\{(x_1, y_1), \ldots, (x_n, y_n)\}$
- Задача: восстановить функцию *f*
- Неявное предположение: *f* зависит от параметров, число которых много меньше размерности *X*
- Выбранный метод (например, нейронная сеть определённой архитектуры) фиксирует параметрическое семейство функций.
- Тогда задача обучения – подобрать параметры, чтобы функция аппроксимировала обучающую выборку наилучшим образом

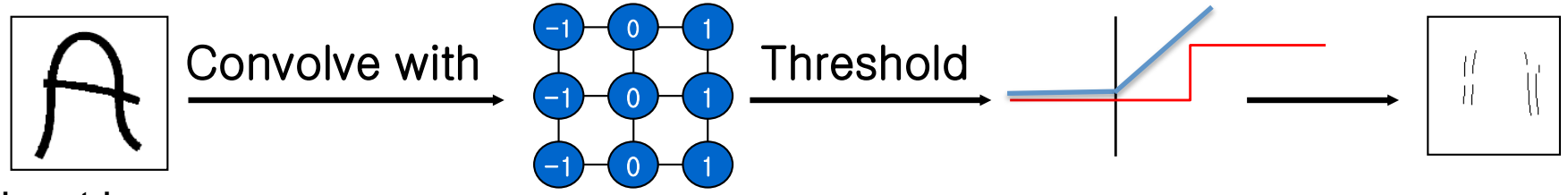
Виды машинного обучения

- С учителем (классификация, регрессия)
- Без учителя (кластеризация, понижение размерности, визуализация)
- С подкреплением (обратная связь от среды, «кнут и пряник»)

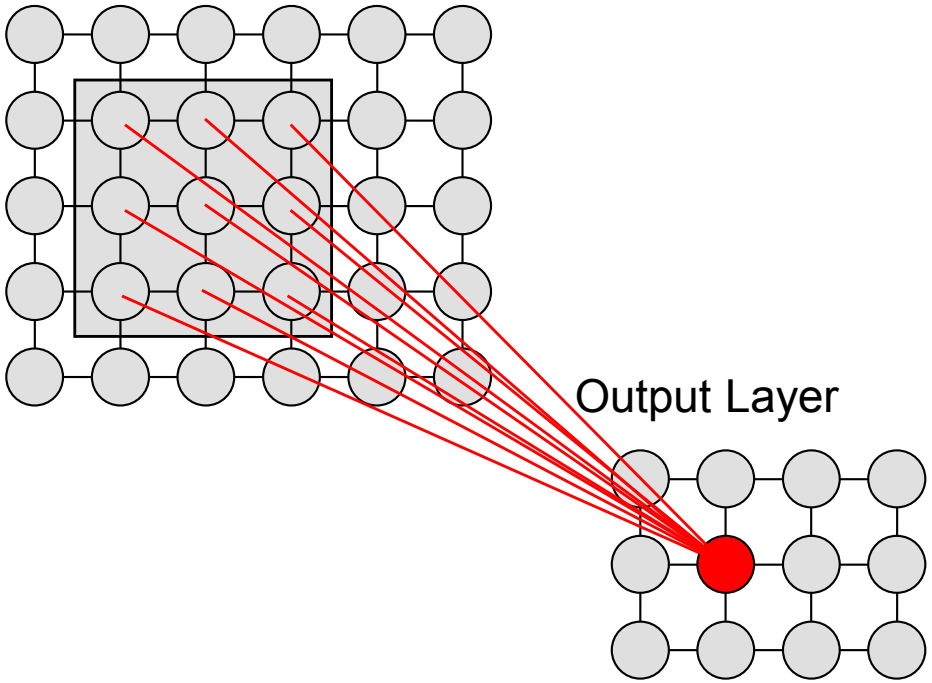
В.Н. Вапник, А.Я. Червоненкис, Ю.И. Журавлёв

Свёрточные нейронные сети

Convolutional Neural Network



Input Layer

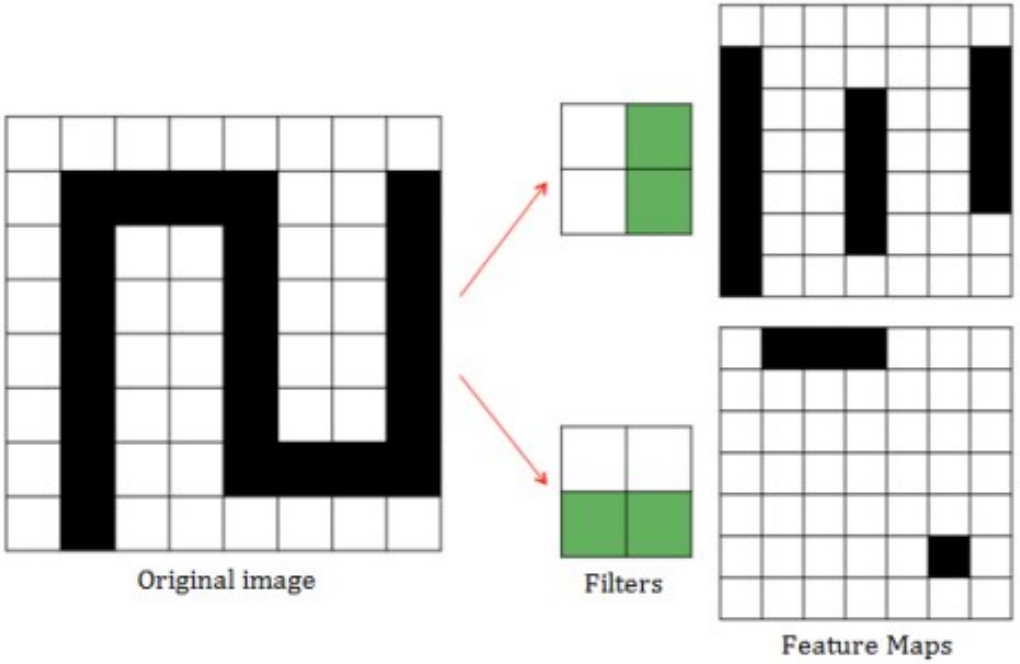


$$y_{ij} = \theta\left(\sum_{k,l=1}^{3} w_{kl} x_{i+k-1,j+l-1} - b \right)$$

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

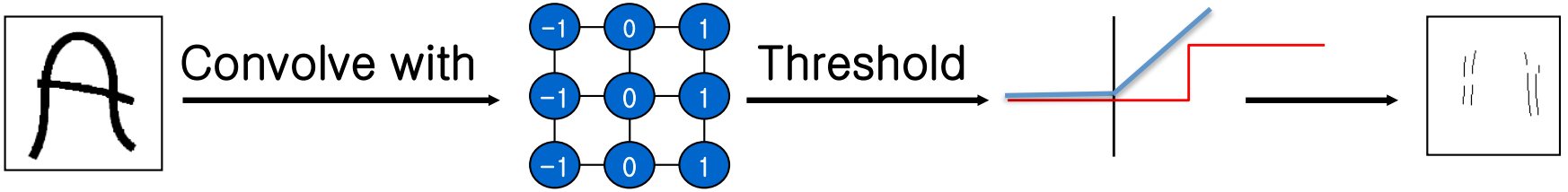
slide by Abi-Roozgard

Фильтр: вычленение элементов

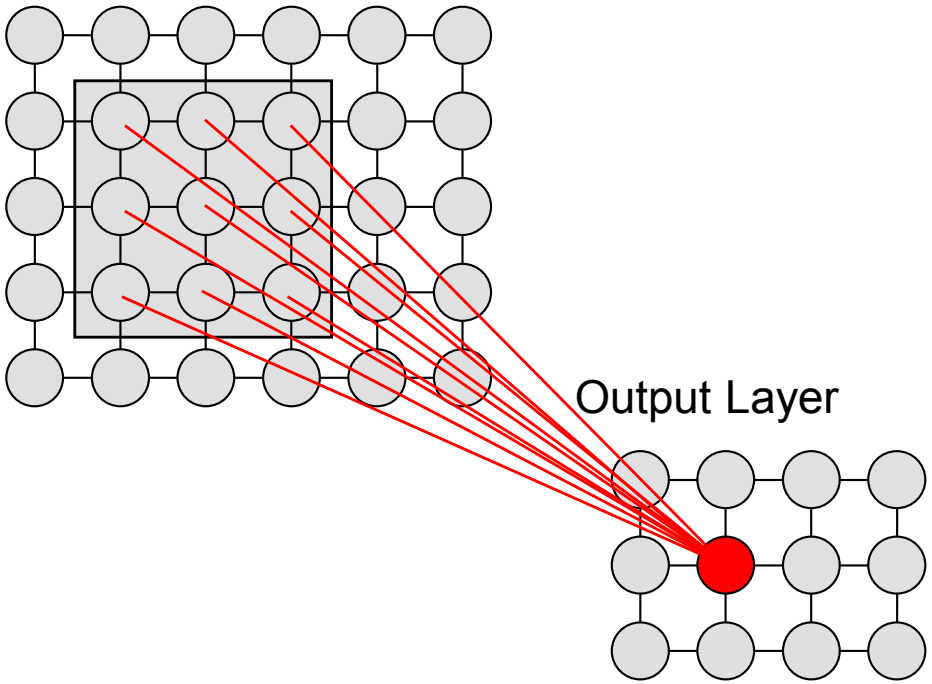


Свёрточные нейронные сети

Convolutional Neural Network



Input Layer



$$y_{ij} = \theta\left(\sum_{k,l=1}^{3} w_{kl} x_{i+k-1,j+l-1} - b \right)$$

Многослойная сеть:
выход одного слоя
– вход следующего

Обучение – подбор
Коэффициентов w_{kl} и b
для каждого фильтра и слоя

slide by Abi-Roozgard

Стратегическая нейронная сеть

- Задача классификации: (позиция, ход мастера)
- Вход: 48 двоичных матриц (карт признаков) размерности 19x19 (доска го) – цвет камня, как давно он поставлен, количество свободных соседних пунктов и т.п.
- Выход: матрица 19x19 – распределение вероятностей следующего хода
- **13 слоёв**, 192 фильтра размера 5x5 (на первом слое) и 3x3 (на последующих слоях)
- На последнем слое – 1 фильтр 1x1; вероятность хода *a* при входной позиции *s* (σ – веса сети):

$$p_\sigma(a \mid s) = \frac{\exp(y_a / T)}{\sum_b \exp(y_b / T)}, \qquad T = 0{,}67$$

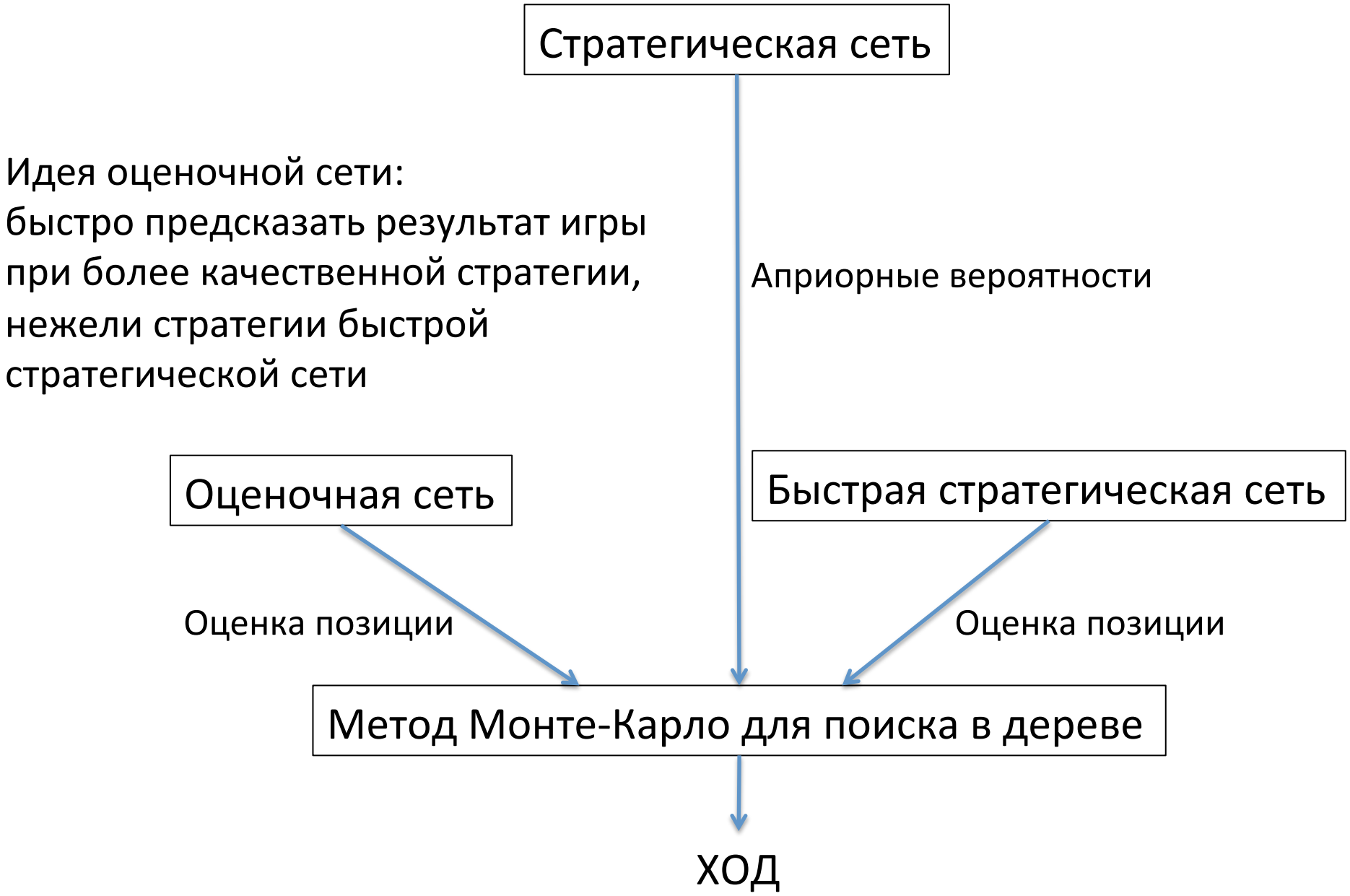
- Корректировка весов (a^k – правильные ответы):

$$\Delta\sigma = \frac{\alpha}{m} \sum_{k=1}^{m} \frac{\partial \log p_\sigma(a^k \mid s^k)}{\partial\sigma}, \quad m = 16, \quad \alpha \le 0{,}003$$

Результаты обучения

- Выборка: 30 млн. позиций с сервера KGS
  - 29 млн. – обучающая выборка
  - 1 млн. – тестовая выборка
- Обучение: 50 GPU (графических процессоров), 3 недели, 340 млн. обучающих шагов
- Точность классификации – 57,0% за 3 мс
  - В программах до этого – 44,4%
  - «Малое улучшение точности привело к большому улучшению силы игры»
- **Быстрая стратегическая сеть** p_π – точность 24,2% за 2 мкс

Схема взаимодействия компонентов



Оценочная нейронная сеть

- Обучающая выборка:
  (*s*,*z*) = (позиция из игры, результат игры)
- Выход сети $v_\theta(s) \in [-1,1]$ (θ – веса сети)
- Архитектура сети почти совпадает с архитектурой стратегической сети
- Обучение – минимизация среднеквадратичной ошибки:

$$E(\theta) = \sum_{(s,z)} (z - v_\theta(s))^2, \quad \Delta\theta \sim -\frac{\partial E(\theta)}{\partial \theta} \sim \frac{\partial v_\theta(s)}{\partial \theta}(z - v_\theta(s))$$

- Обучение с учителем, задача регрессии

Обучающая выборка?

- Если обучающая выборка – игры с сервера, то система «переобучается» (не обобщает, а запоминает исход конкретной партии)
  - Точность по обучающей выборке: 37%
  - Точность по тестовой выборке: 19%
- Причина: последовательные позиции в игре сильно коррелированы, отличаясь на один камень, тогда как результат игры общий.
- Выход: из каждой партии брать только одну позицию в случайный момент. Но тогда объём выборки будет слишком мал
- Следовательно, случайные позиции надо генерировать и затем доигрывать
- Доигрывать по какому алгоритму?

Обучение оценочной сети: доигрывать позиции по какому алгоритму?

- Можно играть по стратегической сети
- Но в AlphaGo создали **улучшенную стратегическую сеть**

Улучшенная стратегическая сеть

- Нейросеть играет сама с собой (точнее, со своими предыдущими версиями), стремится улучшить качество игры
- $p_\rho(a|s)$ – вероятность следующего хода a в позиции s
- Начальные веса сети: $\rho = \sigma$ (от исходной стратегической сети)

$$\Delta\rho \sim \sum_{t=1}^{T} \frac{\partial \log p_\rho(a^t \mid s^t)}{\partial \rho} z$$

(z = +1 в случае выигрыша и –1 в случае проигрыша)

Обучение с подкреплением

Результаты обучения улучшенной стратегической сети

- Обучение: сутки, 50 GPU, 1,28 млн. игр
- Выигрыш в 85% игр против программы Pachi (сильнейшей с открытым кодом, 2-й любительский дан на сервере KGS)
- Для сравнения: выигрыш исходной стратегической сети против Pachi – в 11% игр.

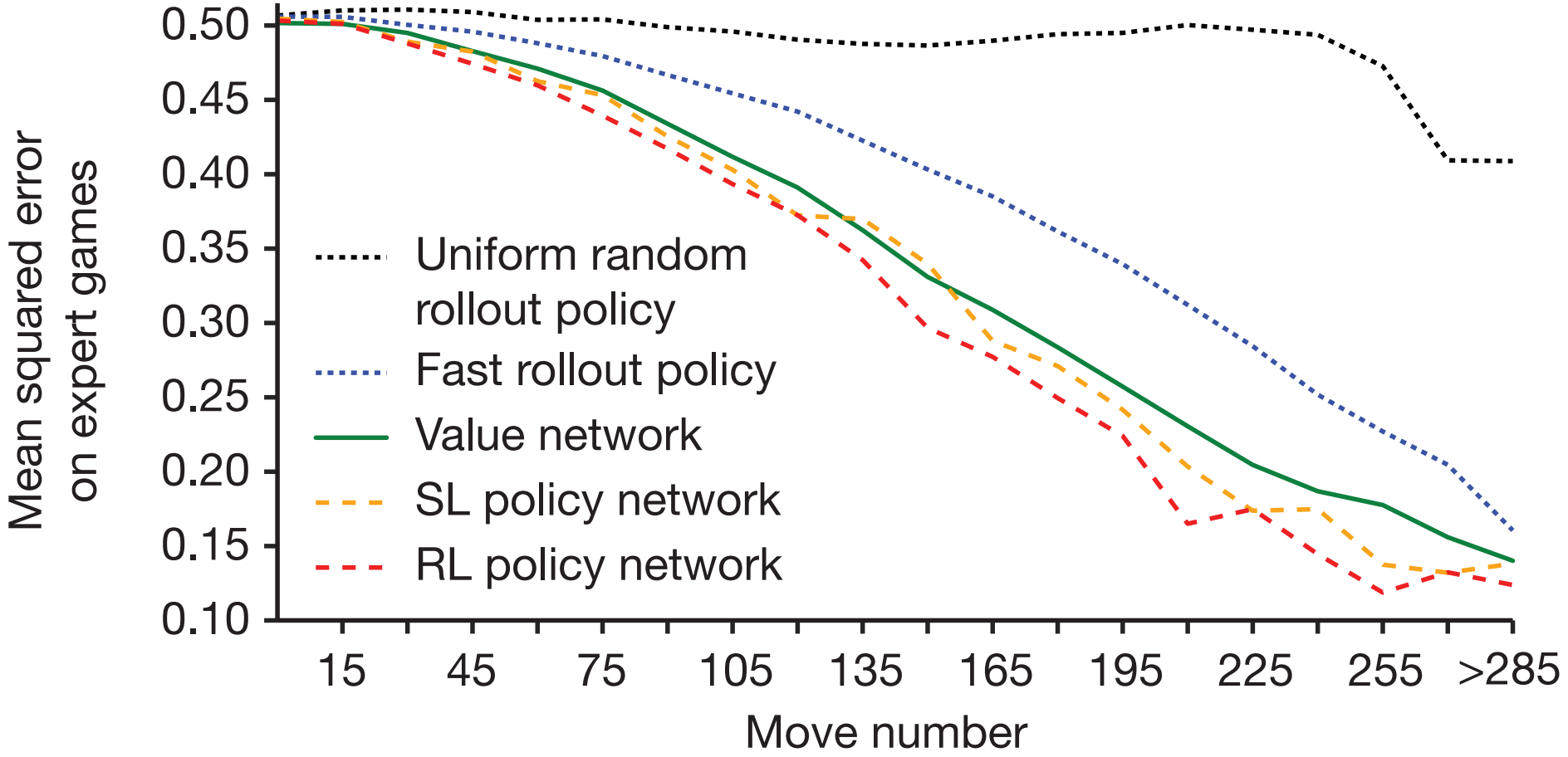
Генерация случайной позиции

- Генерация случайного t от 1 до 450
- Генерация случайной игры по исходной стратегической сети (т.е. имитация игры мастеров) до хода t–1 включительно
- Генерация случайного хода t, равномерно распределённого на множестве допустимых ходов

Обучение оценочной сети

- Выборка: 30 млн. случайных позиций, далее сравнение $v_\theta(s)$ с результатом игры программы с самой собой по стратегии ρ.
- Точность: 23,4%.

Сравнение разных методов предсказаний результата игры



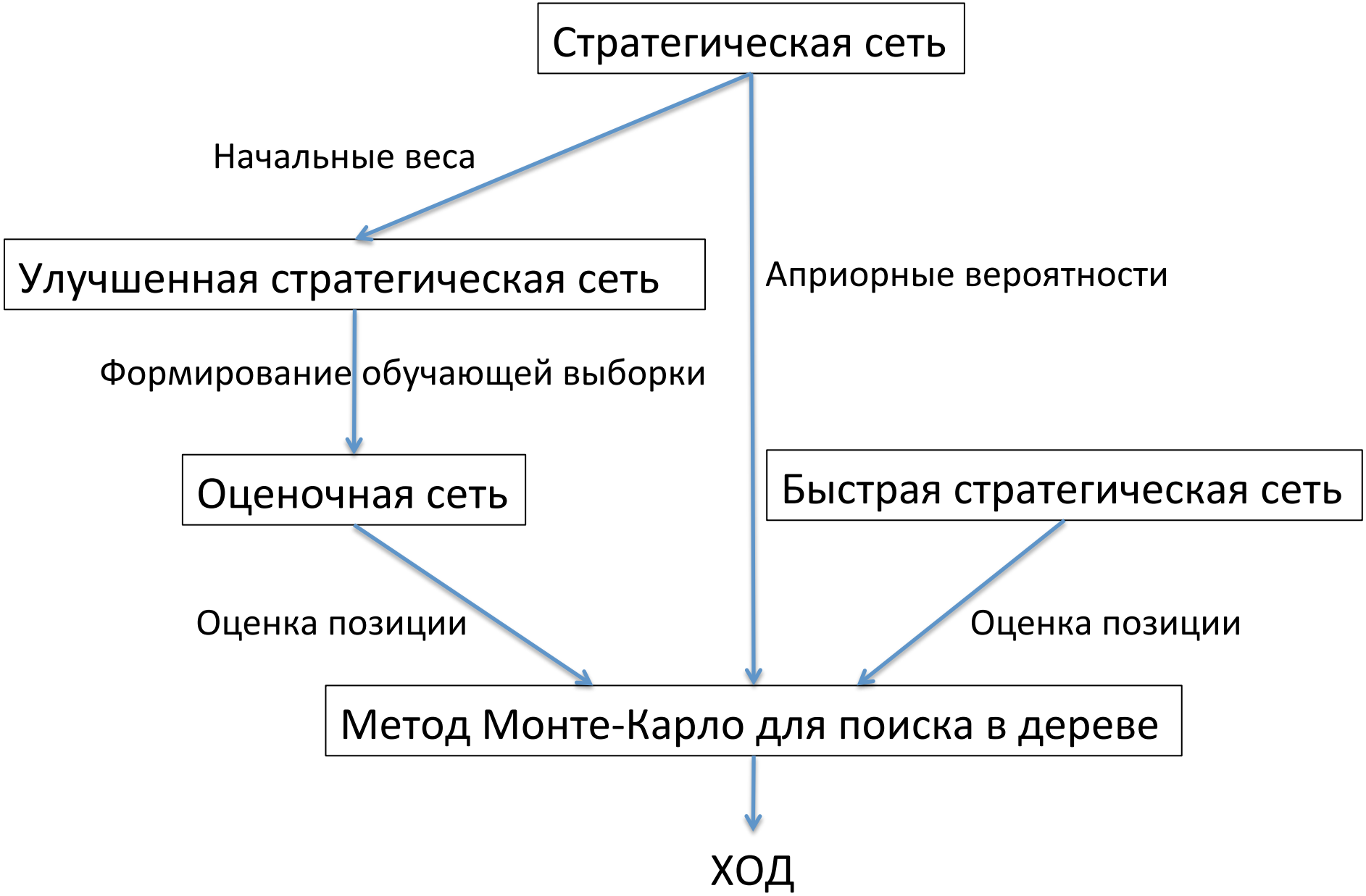
Оценивающая сеть (value network) предсказывает почти так же, как улучшенная стратегическая сеть (RL policy network),
но требует в 15 000 раз меньше вычислений

Оценка позиции s

$$V(s) = \frac{z_\pi + v_\theta(s)}{2}$$

- $v_\theta(s)$ – предсказание оценочной нейронной сети
  - Достоинство: предсказание на основе более качественной стратегии
  - Недостаток: однократное детерминированное вычисление
- z_π – результат случайной игры по быстрой стратегии
  - Недостаток: предсказании на основе менее качественной стратегии
  - Достоинство: многократные вычисления, накопление статистики

Схема взаимодействия компонентов



«В матче против Фань Хуэя AlphaGo анализировала в тысячи раз меньше позиций, чем DeepBlue в матче по шахматам против Каспарова, компенсируя это тем, что выбирала эти позиции более разумно, используя стратегическую сеть, и оценивала их более точно, используя оценочную сеть, – подход, который, вероятно, ближе к тому, как играет человек».

Silver D. et al. Mastering the game Go with deep neural networks and tree search // Nature. 2016. V. 529. P. 484–489.

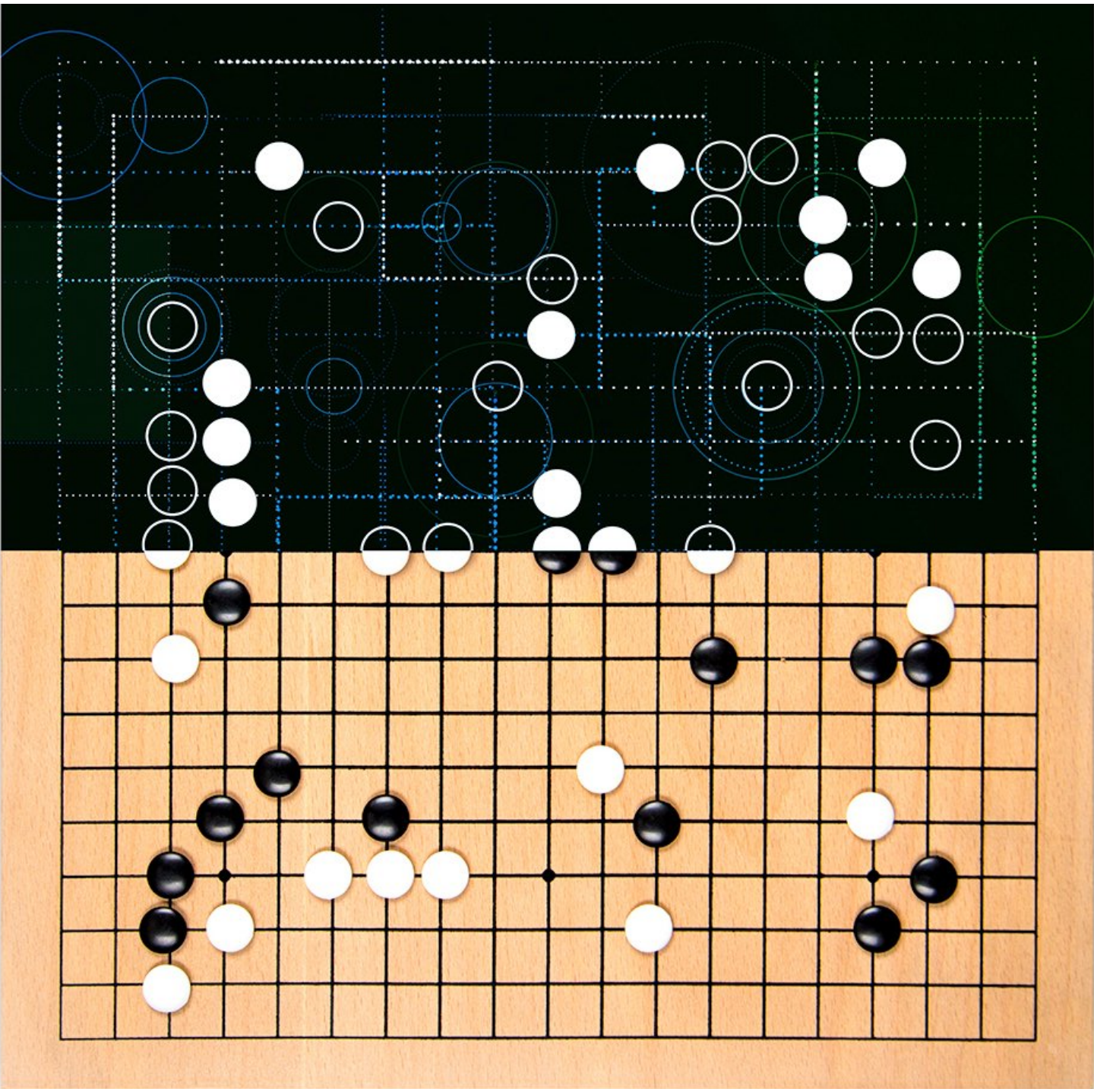
Что дальше?

- Май 2017:
  - 23–27 мая: матч против Кэ Цзэ, который считается игроком №1 в мире
  - Игра против команды из лучших китайских игроков
  - Человек и AlphaGo (чередующиеся ходы) против человека и AlphaGo: «совместное обучение»
- Можно ли эти технологии перенести за пределы игр, «в реальный мир»?
- Достоинство AlphaGo: алгоритмы довольно общие, особенности игры го почти не используются
- Анонсировано применение в медицинской диагностике

Заключение

- Программа AlphaGo использует метод Монте-Карло для поиска в дереве, использующий методы машинного обучения (нейронные сети) в качестве подзадач
- Внимательная проработка деталей

за внимание!



Спасибо