ALPHAZERO И ПОРОЖДАЮЩИЕ МОДЕЛИ

Сергей Николенко

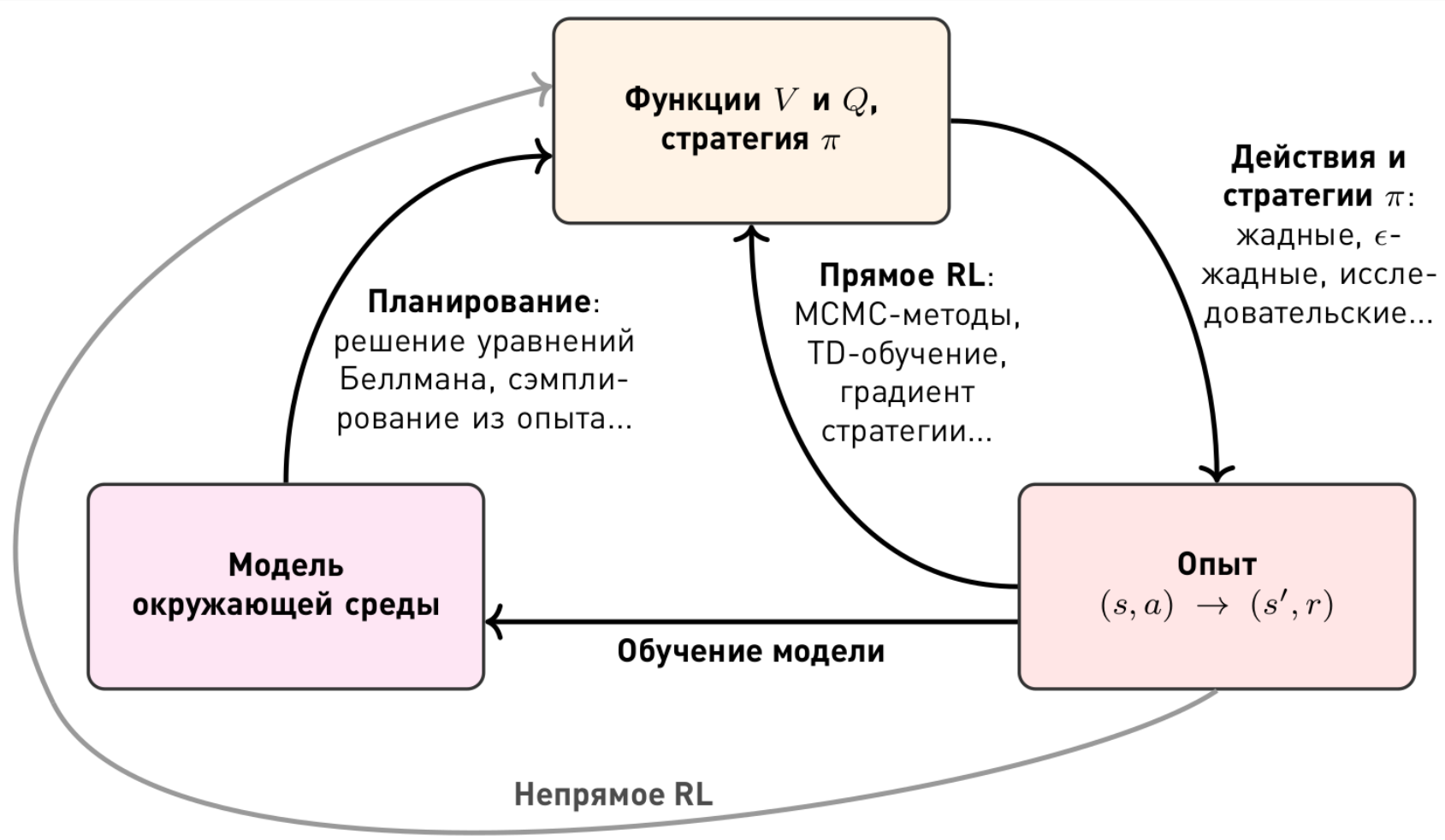
СПбГУ — Санкт-Петербург

16 мая 2024 г.

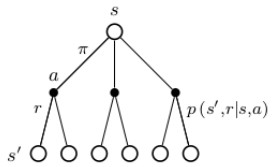
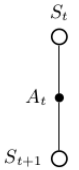
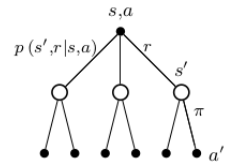
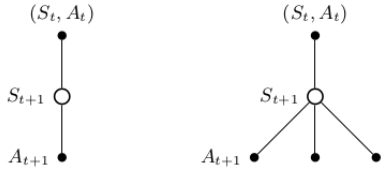
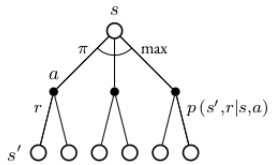
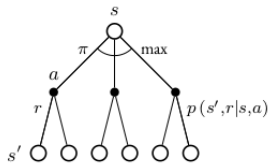
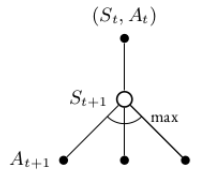
Random facts:

- 18 мая 1703 г. — день основания Балтийского флота: 30 лодок под командованием Петра I захватили два крупных шведских корабля «Гедан» и «Астрильд»
- 18 мая 1756 г. Великобритания объявила войну Франции, и началась Семилетняя война; 18 мая 1803 г. Великобритания объявила войну Франции (закончился Амьенский мир), и продолжились наполеоновские войны, а 18 мая 1804 г. Наполеон I был провозглашён императором французов
- 18 мая 1896 г. на Ходынском поле должна была пройти коронация Николая II, где должны были раздавать эмалированные кружки, колбасу, пряники и сласти; в 5 часов утра на Ходынке собрались уже более 500 тысяч человек, и в давке погибли почти 2000; в тот же день в доме №46 по Невскому проспекту был открыт первый в Петербурге кинотеатр
- 18 мая 1909 г. балетом «Князь Игорь» открылись первые Дягилевские вечера русского балета в Париже
- 18 мая 1985 г. в газете «Правда» появилась рубрика «Трезвость — норма жизни»

Обучение с подкреплением



Обзор методов

	Функция	Обновление в ожидании	Обновление по выборке
Оценка стратегии π	$V_\pi(S_t) :=$	**Уравнение Беллмана** $\sum_{a\in\mathcal{A}} \pi(a\|S_t) \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\|s,a)\,(r+\gamma V_\pi(s'))$ 	**TD**(0) $V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$ 
	$Q_\pi(S_t,A_t) :=$	**Уравнение Беллмана** $\sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\|S_t,A_t)\,(r+\gamma \sum_{a'\in\mathcal{A}} \pi(a'\|s')\,Q_\pi(s',a'))$ 	**Sarsa, Sarsa с ожиданием** $Q(S_t,A_t) + \alpha\,(R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t))$ $Q(S_t,A_t) + \alpha\,\big(R_{t+1} + \gamma \sum_a \pi(a\|S_{t+1})\,Q(S_{t+1},a) - Q(S_t,A_t)\big)$ 
Управление, поиск π_*	$V_*(S_t) :=$	**Уравнение Беллмана** $\max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\|S_t,a)\,(r+\gamma V_*(s'))$ 	
	$Q_*(S_t,A_t) :=$	**Уравнение Беллмана** $\sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r\|S_t,A_t)\left(r+\gamma \max_{a'} Q_*(s',a')\right)$ 	**Q-обучение** $Q(S_t,A_t) + \alpha\,(R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t))$ 

AlphaGo

What happened

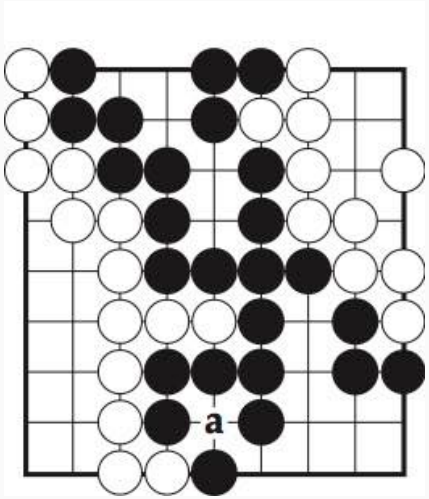
- 8-15 марта 2016 года прошёл исторический матч.
- AlphaGo (Google DeepMind) $\mathbf{4:1}$ Ли Седоль 9p (Корея).

What happened

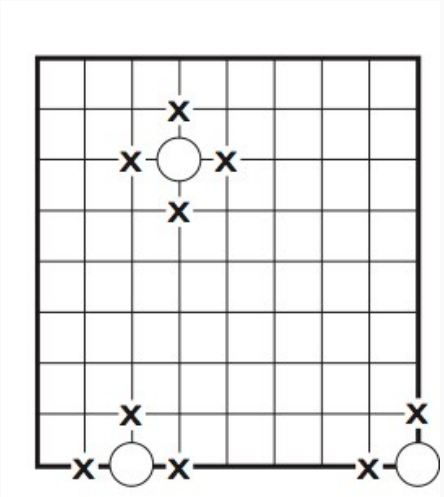
- 8-15 марта 2016 года прошёл исторический матч.
- AlphaGo (Google DeepMind) $\mathbf{4:1}$ Ли Седоль 9p (Корея).

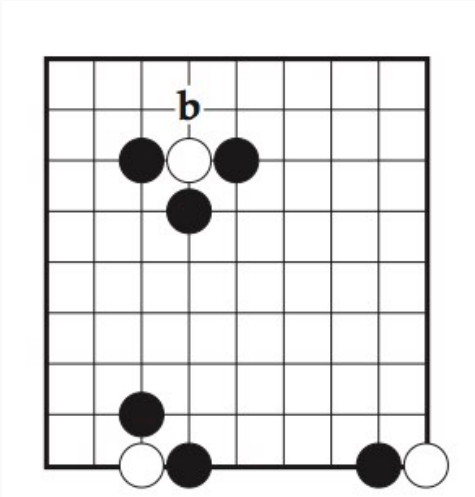
- ...but why do we care?

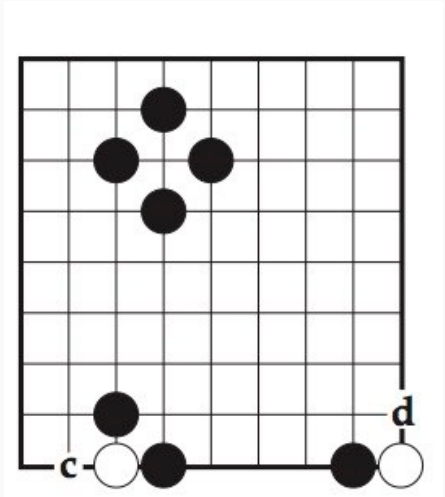
Правила го



- Вкратце о правилах: камни ставят на доску 19×19 (9×9, 13×13).
- Камни снимаются, когда их окружают.
- Побеждает тот, кто занимает больше территории (плюс съеденные камни).





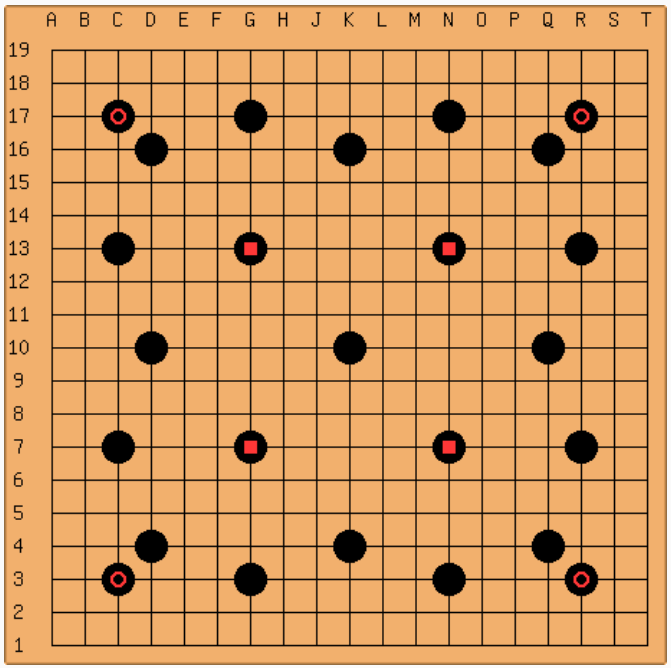


История компьютерного го

- 1968: Albert Zobrist – первая программа, влияние камней, Zobrist hashing.
- Середина 1980х: появились PC, интерес к этому.
- Ing Prize: 40M тайваньских долларов, если компьютер сможет до 2000 года обыграть профессионала 1 дана.
- Программы 1990х: Goliath (1989-1991), Go Intellect, Handtalk, Goemate.
- (David Fotland, 1993): Many Faces of Go, долгое время одна из ведущих программ.
- Они работали на pattern matching и базах знаний.
- И насколько же хорошо они играли?..

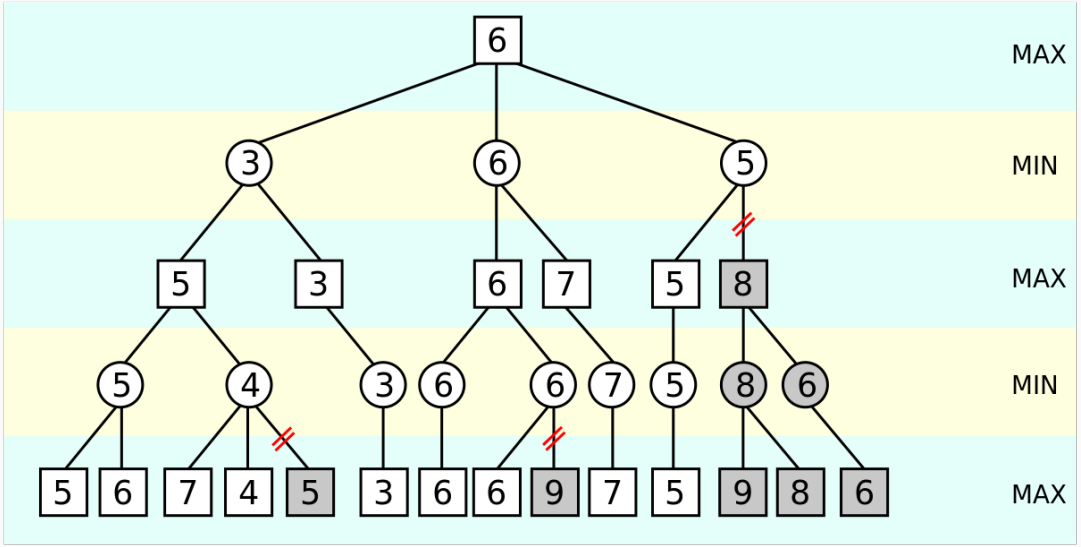
О силе игры

- ...да совсем никак.
- 1997: Janice Kim 1p победила HandTalk *на 25 камнях форы*.
- В том же году HandTalk выиграл часть Ing Prize, победив трёх 11-13-летних любителей 2-6d на 11 камнях форы.
- 2001: Many Faces выиграл у insei (1kyu pro) *на 15 камнях*.
- Это те годы, когда DeepBlue уже давно победил Каспарова.
- В чём же дело? Почему так сложно?



Почему го -- это сложно

- В других играх (шахматы) отлично работает alpha-beta search: строим минимакс-дерево ходов, выкидываем ходы, которые заведомо хуже других, ищем в глубину.



- Почему не в го?

Почему го -- это сложно

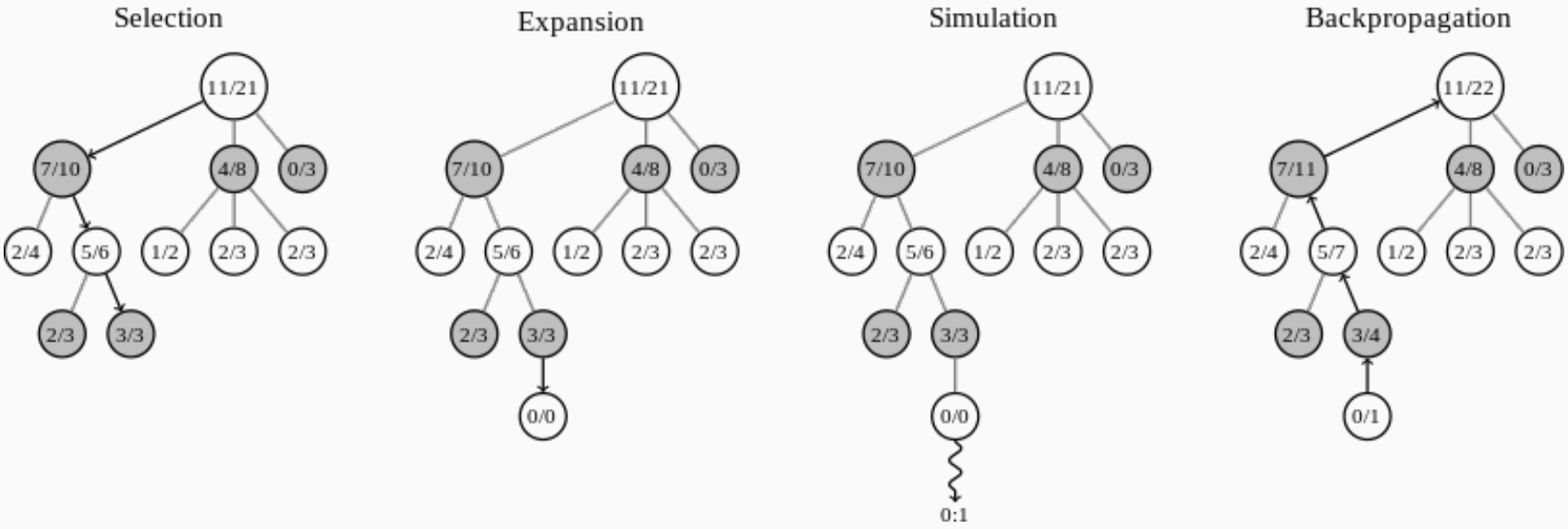
- Почему го сложнее, чем шахматы?
- Очень большая ширина дерева (branching factor):
  - в шахматах 30-40 возможных ходов, из них многие сразу приводят к потерям и их можно обрезать;
  - в го около 250 возможных ходов, и из них «вполне разумных» тоже около сотни.
- Оценка позиции:
  - в шахматах можно посчитать материал и какие-то простые эвристики, и если разрыв большой, скорее всего оценка очевидна;
  - в го тоже может упасть большая группа, но это редкость, обычно потери территориальные, труднооцениваемые;
  - очень сложная для автоматизации задача – даже просто оценить конечную позицию в го (когда люди уже согласились, кто выиграл).

Почему го -- это сложно

- Тактика – life and death problems:
  - казалось бы, компьютеры считают варианты быстро;
  - но в го на решение локальных проблем влияют далеко расположенные камни и вся позиция; редко можно обрезать часть доски и сделать полный перебор;
  - а статус групп может измениться под влиянием глобальных эффектов и взаимодействовать с ними;
  - кроме того, человеку в го считать проще, чем в шахматах.
- Стратегия: нужно понимать взаимодействие камней на всей доске и очень тонко оценивать возникающую позицию, досчитать большинство веток до «явного преимущества», как в шахматах, невозможно.
- Ко-борьба: программы всегда отвратительно играли ко, бездарно тратили ко-угрозы, не замечали, что ко может возникнуть.

Monte-Carlo tree search

- 2006 (Remi Coulom, затем MoGo, затем все): революция компьютерного го – Monte-Carlo tree search (MCTS).
- Запускаем *случайные симуляции* с текущей позиции, смотрим, в каких ветках больше выигрышей, повторяем.



Monte-Carlo tree search

- Основная часть MCTS – формула, по которой выбирают узел для дальнейшего анализа:
  - это всё основано на тех же многоруких бандитах;
  - UCT (upper confidence bound UCB1 applied to trees) – выбираем узел с максимальным

$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln t}{n_i}},$$

где w_i – число побед, n_i – число симуляций, c – параметр (часто $\sqrt{2}$), $t = \sum_i n_i$ – общее число симуляций.

Monte-Carlo tree search

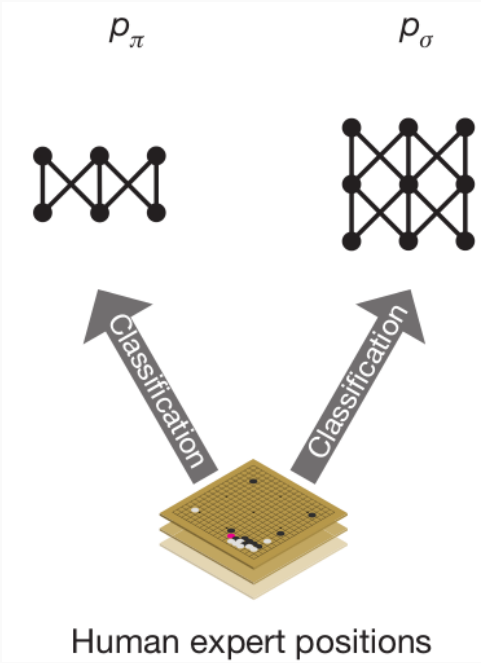
- И пошли более разумные результаты:
  - 2006: CrazyStone выиграл ICGA Computer Olympiad на доске 9×9;
  - 2007: CadiaPlayer – чемпион по General Game Playing;
  - 2008: MoGo достиг уровня дана на доске 9×9;
  - 2009: Fuego побеждает одного из топ-игроков на доске 9×9;
  - 2009: Zen достигает уровня 1d на сервере KGS на доске 19×19;
  - 2012: Zen выиграл 3:1 матч у 2d любителя на доске 19×19;
  - 2013: CrazyStone выиграл у Yoshio Ishida 9p на четырёх камнях;
  - около 2015-2016 CrazyStone и Zen, основанные на MCTS, достигали уровня 6-7d на KGS (но это совсем не то же, что профессиональные даны).
- MCTS работал и в других играх (Hex, Arimaa, MtG, Settlers).
- Но в 2016-м «внезапно» появилась AlphaGo. Что же она делает?

AlphaGo: история

- AlphaGo разработана в 2014-2015гг. в Google DeepMind: David Silver, Aja Huang и другие.
- В октябре 2015 года распределённая версия AlphaGo (1202 CPUs + 176 GPUs) победила Fan Hui 2p, чемпиона Европы, со счётом 5:0 (и 3:2 с укороченным контролем).
- Об этом рассказывает статья в *Nature* «Mastering the game of Go with deep neural networks and tree search», основной источник об AlphaGo.
- В 2016 – победа 4:1 над Lee Sedol 9p, одним из лучших профессионалов в мире.
- Что же изменилось в AlphaGo по сравнению с MCTS-программами?

AlphaGo: краткий обзор

- SL policy network предсказывает ходы:
  - 13-слойная свёрточная сеть выделяет признаки по всей доске;
  - обучается на профессиональных играх (30 млн позиций с KGS);
  - на выходе распределение ходов $p_\sigma(a \mid s)$;
  - получается 57.0% правильных предсказаний (очень много);
  - ещё обучаем быструю, но более плохую стратегию p_π (точность 24.2%, но оценка позиции за 2нс вместо 3мс).

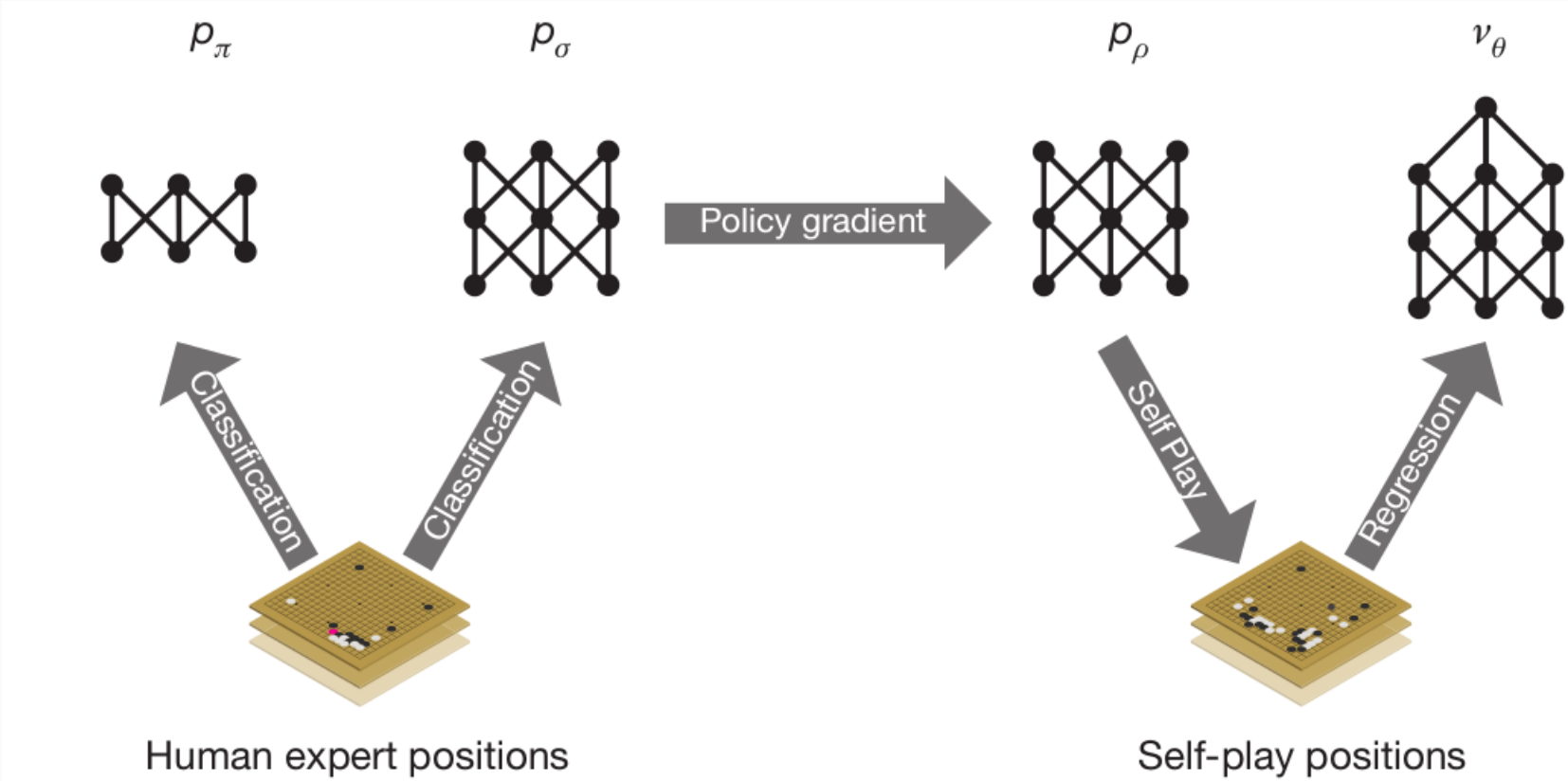


AlphaGo: краткий обзор

- SL policy network предсказывает ходы.
- RL policy network улучшает policy network за счёт обучения с подкреплением:
  - та же структура (13-слойная свёрточная сеть);
  - инициализируется с SL policy network;
  - затем играет сама с собой (с одной из предыдущих итераций обучения);
  - веса дообучаются так, чтобы максимизировать вероятность выигрыша;
  - RL policy network выигрывает 80% игр против SL, 85% игр против Pachi (одной из лучших MCTS-программ);
  - и это всё ещё без всякого счёта, просто глобальной оценкой позиции!

AlphaGo: краткий обзор

- SL policy network предсказывает ходы.
- RL policy network улучшает policy network за счёт обучения с подкреплением.

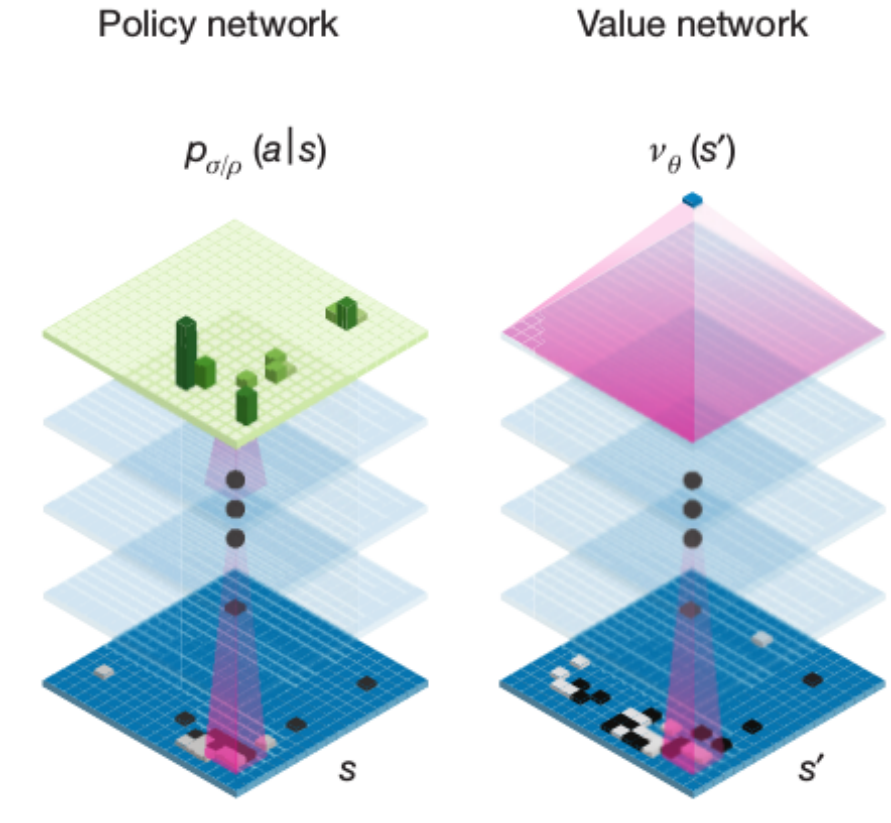


AlphaGo: краткий обзор

- SL policy network предсказывает ходы.
- RL policy network улучшает policy network за счёт обучения с подкреплением.
- Затем обучаем функцию оценки позиции $v_\theta(s)$:
  - предсказываем результат игры по позиции нейронной сетью;
  - структура сети похожа на policy networks, но теперь один выход;
  - если обучить на наборе профессиональных игр, будет просто overfitting, недостаточно данных;
  - поэтому обучаем на наборе self-play, порождённом RL policy network;
  - получается сеть для оценки позиции, которая работает очень быстро по сравнению с MCTS и даёт качество на уровне MCTS с RL policy (в 15K раз быстрее) и гораздо лучше, чем обычный MCTS.

AlphaGo: краткий обзор

- SL policy network предсказывает ходы.
- RL policy network улучшает policy network за счёт обучения с подкреплением.
- Затем обучаем функцию оценки позиции $v_\theta(s)$.

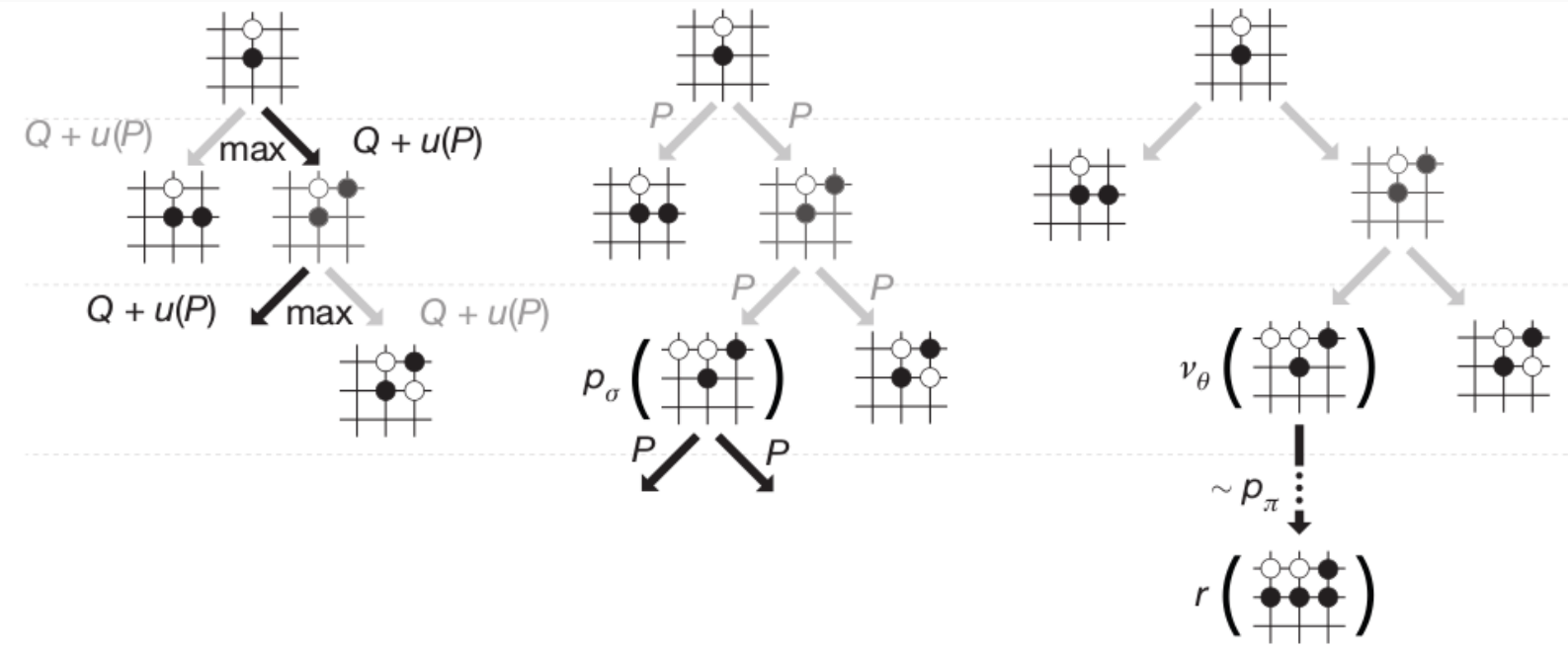


AlphaGo: краткий обзор

- SL policy network предсказывает ходы.
- RL policy network улучшает policy network за счёт обучения с подкреплением.
- Затем обучаем функцию оценки позиции $v_\theta(s)$.
- А теперь можно и деревья посчитать:
  - строим MCTS-подобное дерево;
  - априорные вероятности инициализируются через SL policy network как $p_\sigma(a \mid s)$;
  - в листе L считаем значение value network $v_\theta(s_L)$ и результат random rollout z_L при помощи стратегии p_π, объединяем результат;
  - любопытно, что SL policy работает лучше для построения дерева (видимо, более разнообразные варианты), но value function лучше строить на основе более сильной RL policy.

AlphaGo: краткий обзор

- SL policy network предсказывает ходы.
- RL policy network улучшает policy network за счёт обучения с подкреплением.
- Затем обучаем функцию оценки позиции $v_\theta(s)$.
- А теперь можно и деревья посчитать.



Матч с Ли Седолем

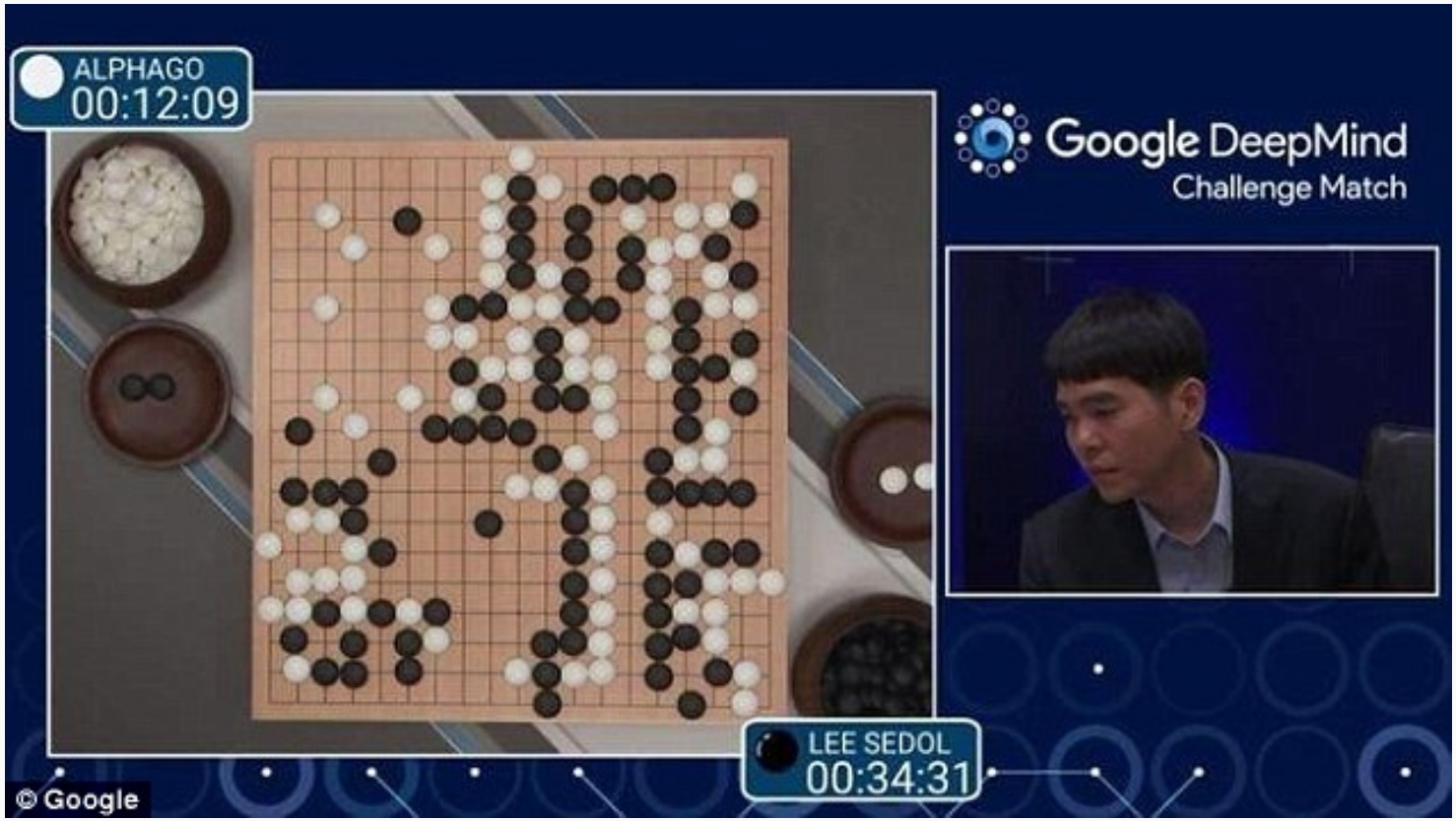
- Перед началом матча Ли Седоль предсказывал свою победу со счётом 5:0 или 4:1.
- Практически все эксперты были согласны с таким прогнозом.

Матч с Ли Седолем

- Первая партия: Ли Седоль необычно разыграл фусеки (дебют), желая, видимо, свернуть с «накатанных» для AlphaGo путей (примерно так играли против компьютера в шахматы, когда это было возможно).

Матч с Ли Седолем

- Исход игры долго был неясен, AlphaGo, по мнению экспертов, делала ошибки, но около 120-130 хода белые (AlphaGo) получили преимущество и безупречно разыграли йосе (эндшпиль), без шансов для Ли Седоля.



Матч с Ли Седолем

- Во второй партии AlphaGo сделала ход, которого комментаторы совершенно не ожидали, и снова безупречно удержала небольшое преимущество в йосе.
- Ли Седоль после второй партии: «AlphaGo played a near perfect game».

Матч с Ли Седолем

- Исход третьей партии решился в первой части игры: AlphaGo блестяще провела сложную тактическую борьбу (слабое место других программ) и точными ходами захватила преимущество.
- Счёт стал 3:0, матч был проигран, казалось, что победа роботов будет абсолютной...

Матч с Ли Седолем

- ...но в четвёртой партии уже Ли Седоль сделал несколько великолепных ходов, успешно использовал недочёты в игре AlphaGo и выиграл!

Матч с Ли Седолем

- Пятая партия была отчасти экспериментальной, Ли Седоль попросил чёрный цвет (которым играть сложнее) и думал, что понимает, как обыграть AlphaGo… но, по своим собственным словам, пожадничал и проиграл.
- «The week with AlphaGo felt like a bamboo clapper».

AlphaGo Master

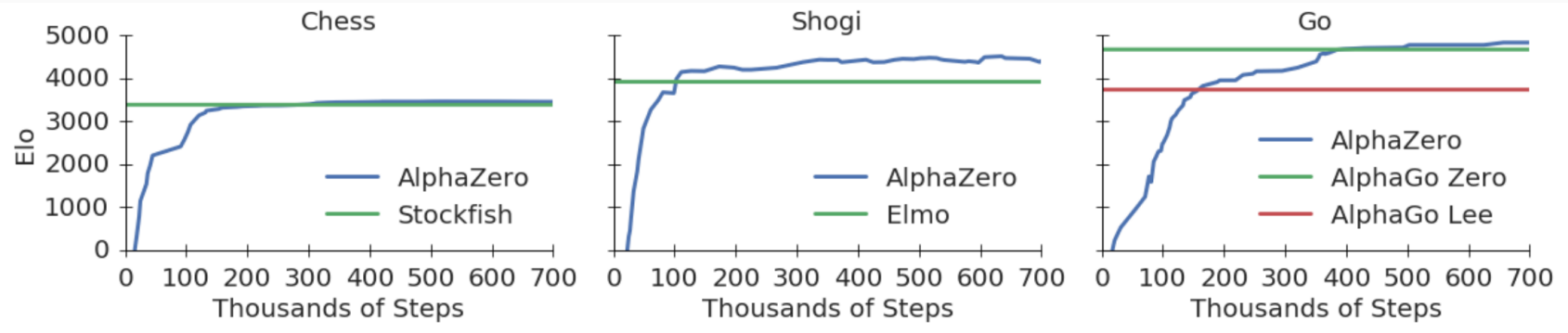
- В конце мая 2017 г. AlphaGo Master (новая версия) обыграла Ke Jie (#1 рейтинга), без шансов, 3:0...

- ...и сейчас AlphaGo ушла из большого спорта, оставив небольшое, но очень важное наследие.

AlphaZero

AlphaGo Zero и AlphaZero

- Следующие супер-новости от DeepMind: AlphaGo Zero (Silver et al., 2017a) и особенно AlphaZero (Silver et al., 2017b).
- AlphaZero обучается без всяких данных, только играя сама с собой.
- И обыгрывает AlphaGo в го, Stockfish в шахматы, Elmo в сёги...



- Как это?

AlphaGo Zero и AlphaZero

- Обычно шахматные программы (тот же Stockfish) основаны на функции оценки позиции и, главное, альфа-бета поиске с отсечениями.
- Альфа-бета поиск обычно работает на основе минимакса. Но AlphaZero действует по-другому:
  - обучается с подкреплением исключительно на играх против самой себя;
  - оценивает позиции глубокой нейронной сетью (о признаках потом);
  - использует MCTS-поиск, оценивая всего 80K позиций в секунду для шахмат (у Stockfish 70M);
  - вместо минимакса усредняет оценку по поддереву; интуиция: ошибки в оценке позиции при этом усредняются и "взаимно сокращаются", а при минимаксе самая большая ошибка пропагируется в корень.

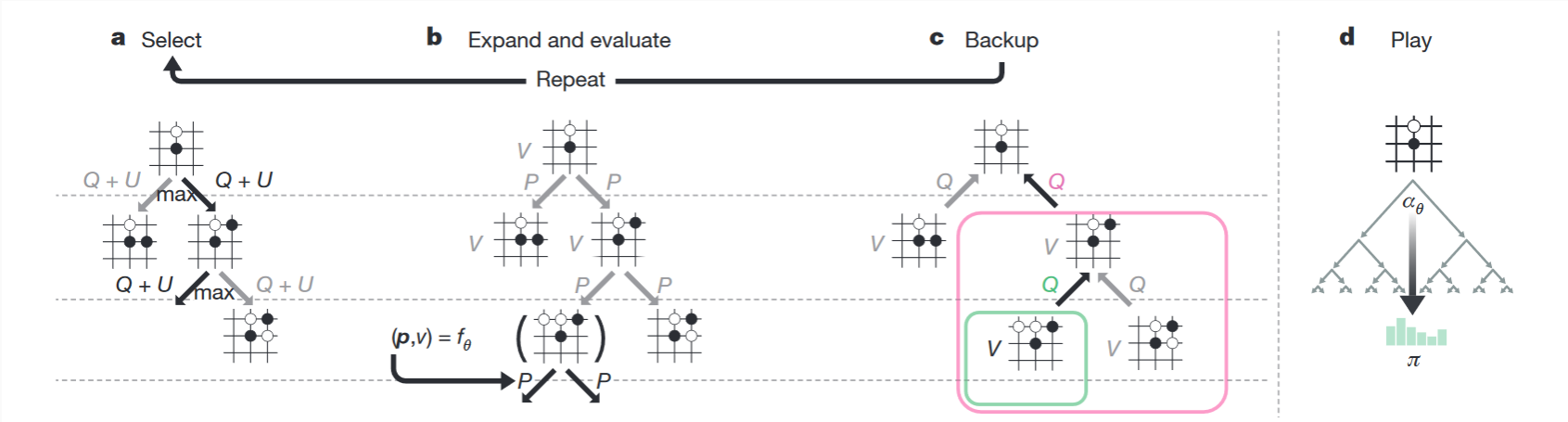
AlphaGo Zero и AlphaZero

- Признаки AlphaZero:

Go		Chess		Shogi	
Feature	Planes	Feature	Planes	Feature	Planes
P1 stone	1	P1 piece	6	P1 piece	14
P2 stone	1	P2 piece	6	P2 piece	14
		Repetitions	2	Repetitions	3
				P1 prisoner count	7
				P2 prisoner count	7
Colour	1	Colour	1	Colour	1
		Total move count	1	Total move count	1
		P1 castling	2		
		P2 castling	2		
		No-progress count	1		
Total	17	Total	119	Total	362

AlphaGo Zero и AlphaZero

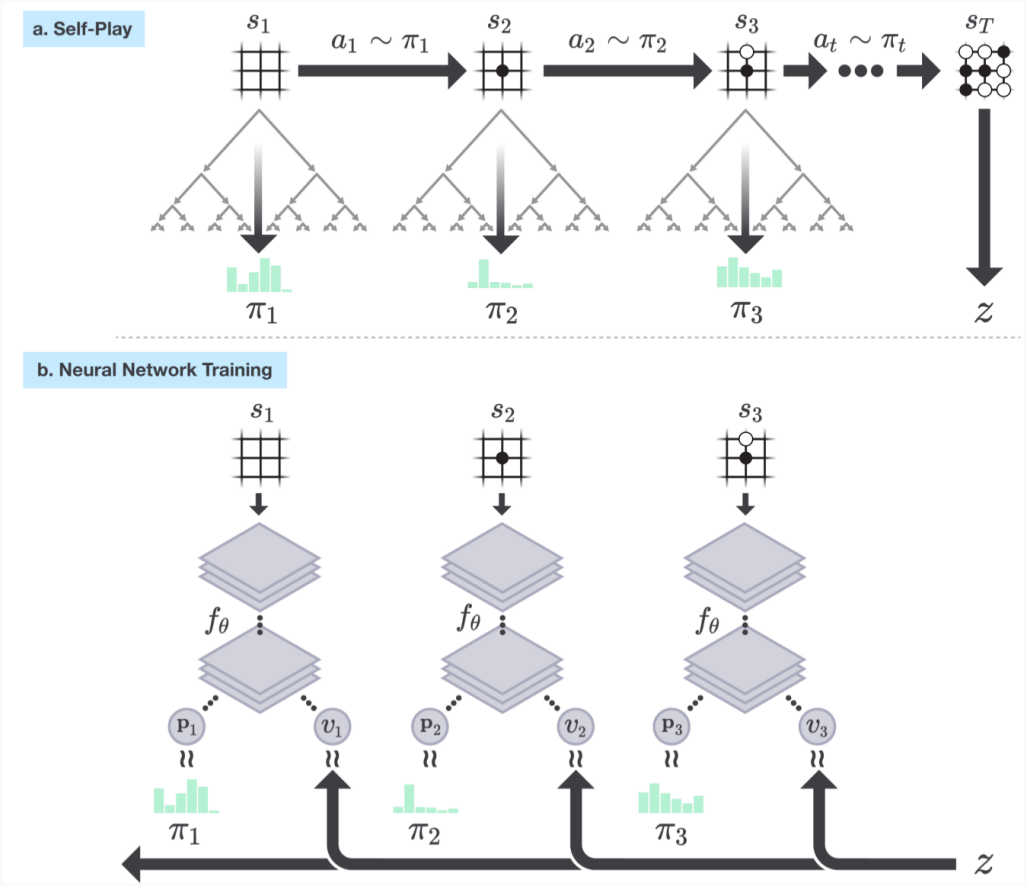
- Как мы делаем ход (картинка из AlphaGo Zero):



- Сразу MCTS при обучении:
  - оцениваем узлы по $Q + U$, где Q — среднее значение детей узла, U — добавка для оптимизма;
  - дойдя до листа, оцениваем через сеть, записываем v в узел, создаём детей по распределению из сети p, больше для этой вершины сеть не запускаем.
- После такого цикла выбираем ход, где больше N, или с дополнительным exploration при обучении.

AlphaGo Zero и AlphaZero

- Обучение с подкреплением (картинка из AlphaGo Zero):



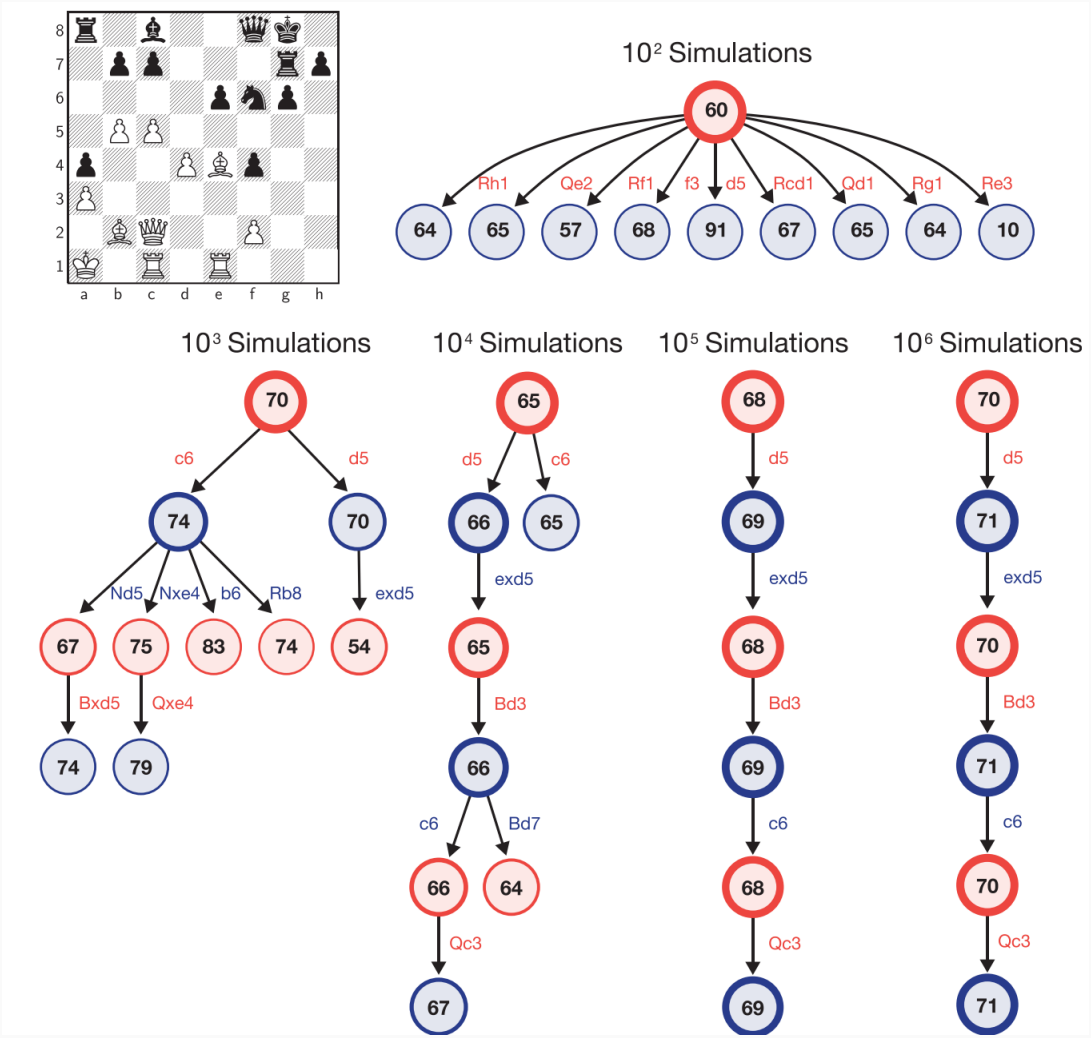
- Накапливаем опыт и минимизируем $L = (z - v)^2 + \pi^\top \log \mathbf{p} + c\|\theta\|^2$, где p — вероятности сети, π — улучшенные через MCTS вероятности.

AlphaGo Zero и AlphaZero

- Более того, AlphaZero даже проще и прямолинейнее, чем AlphaGo Zero:
  - AlphaGo Zero использовала симметрию го для аугментаций, а AlphaZero этого не делает;
  - AlphaGo Zero поддерживала текущего лучшего противника из предыдущих итераций, периодически его заменяя, а AlphaZero просто использует всегда текущую сеть;
  - свёрточная сеть хорошо приспособлена для го, а для шахмат/сёги интуитивно не очень, но на это тоже забили.
- И тем не менее, всё получилось очень хорошо.

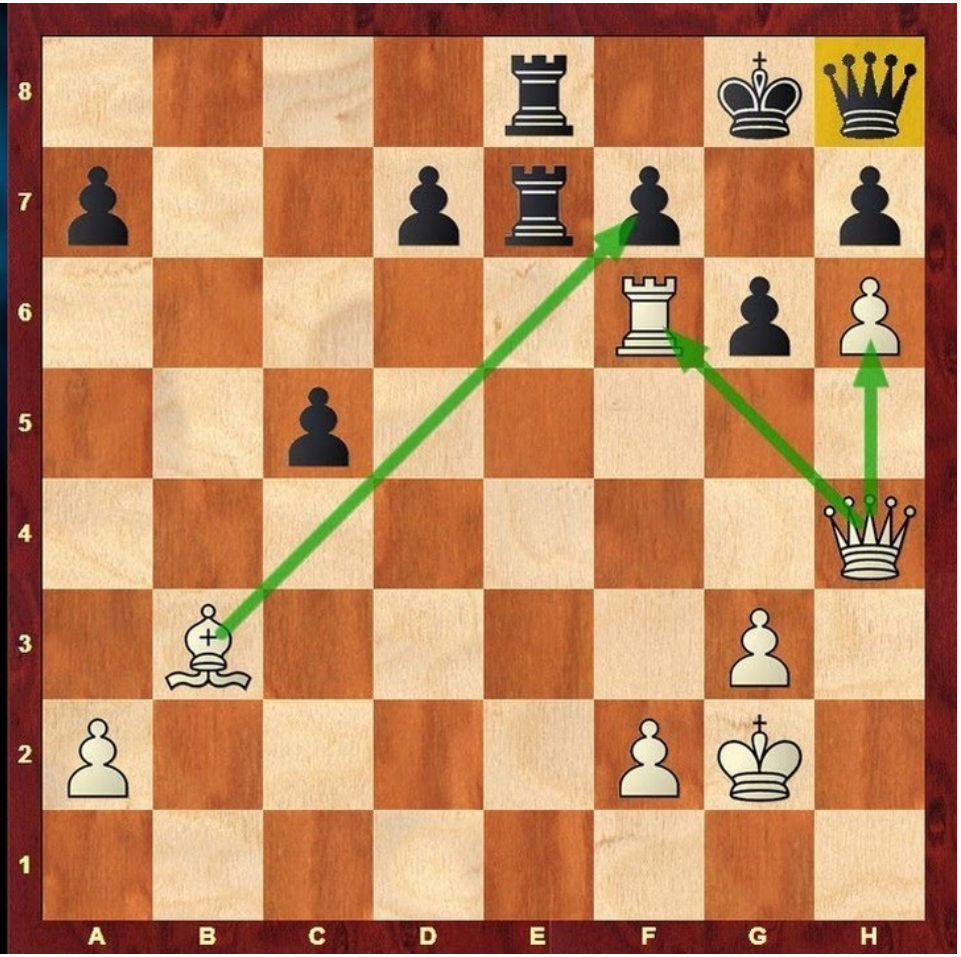
AlphaGo Zero и AlphaZero

- Пример раздумий MCTS:



AlphaGo Zero и AlphaZero

- И всё, да здравствует король!

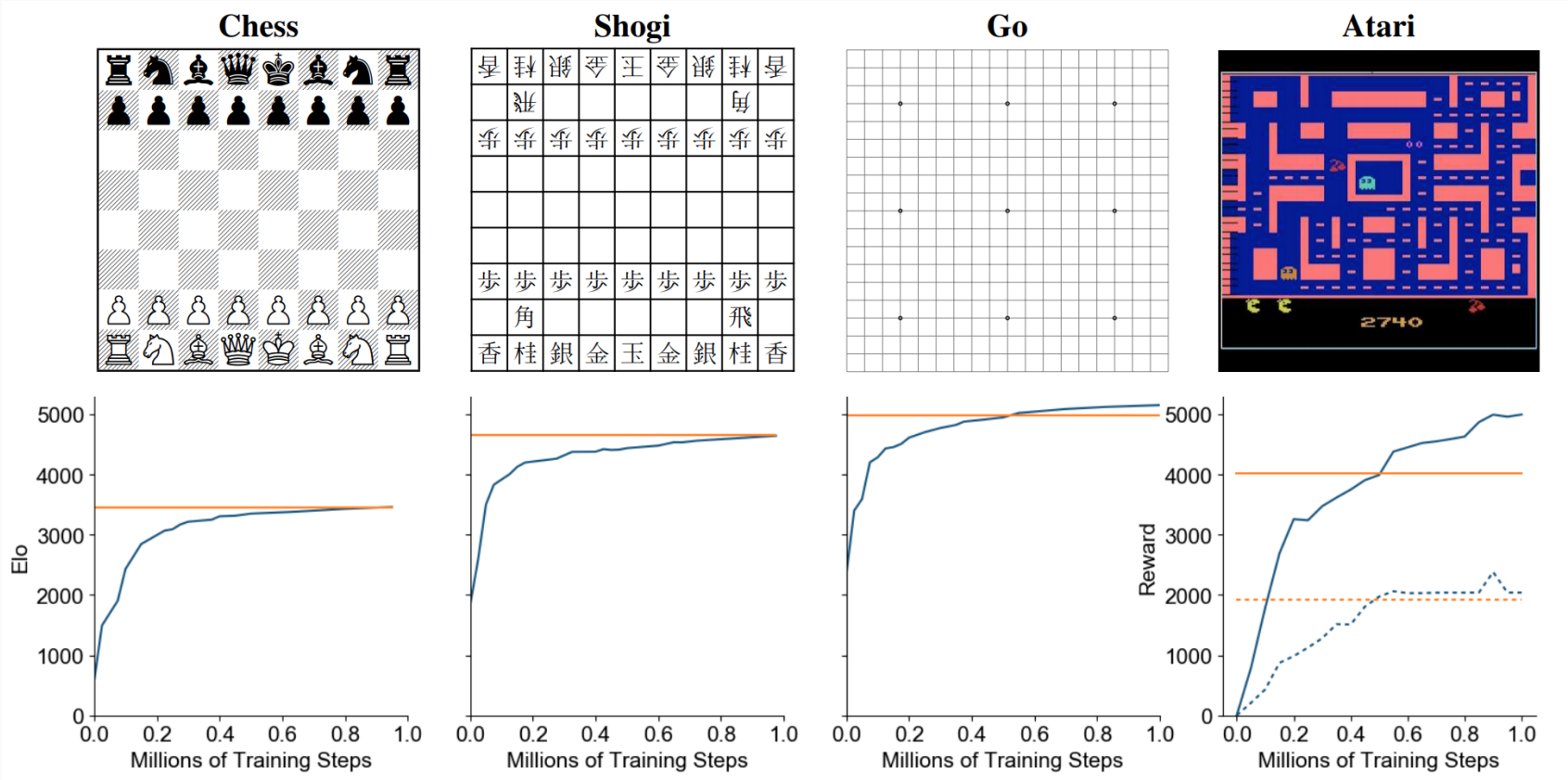


- Но и это ещё не вся история...

MuZero

MuZero

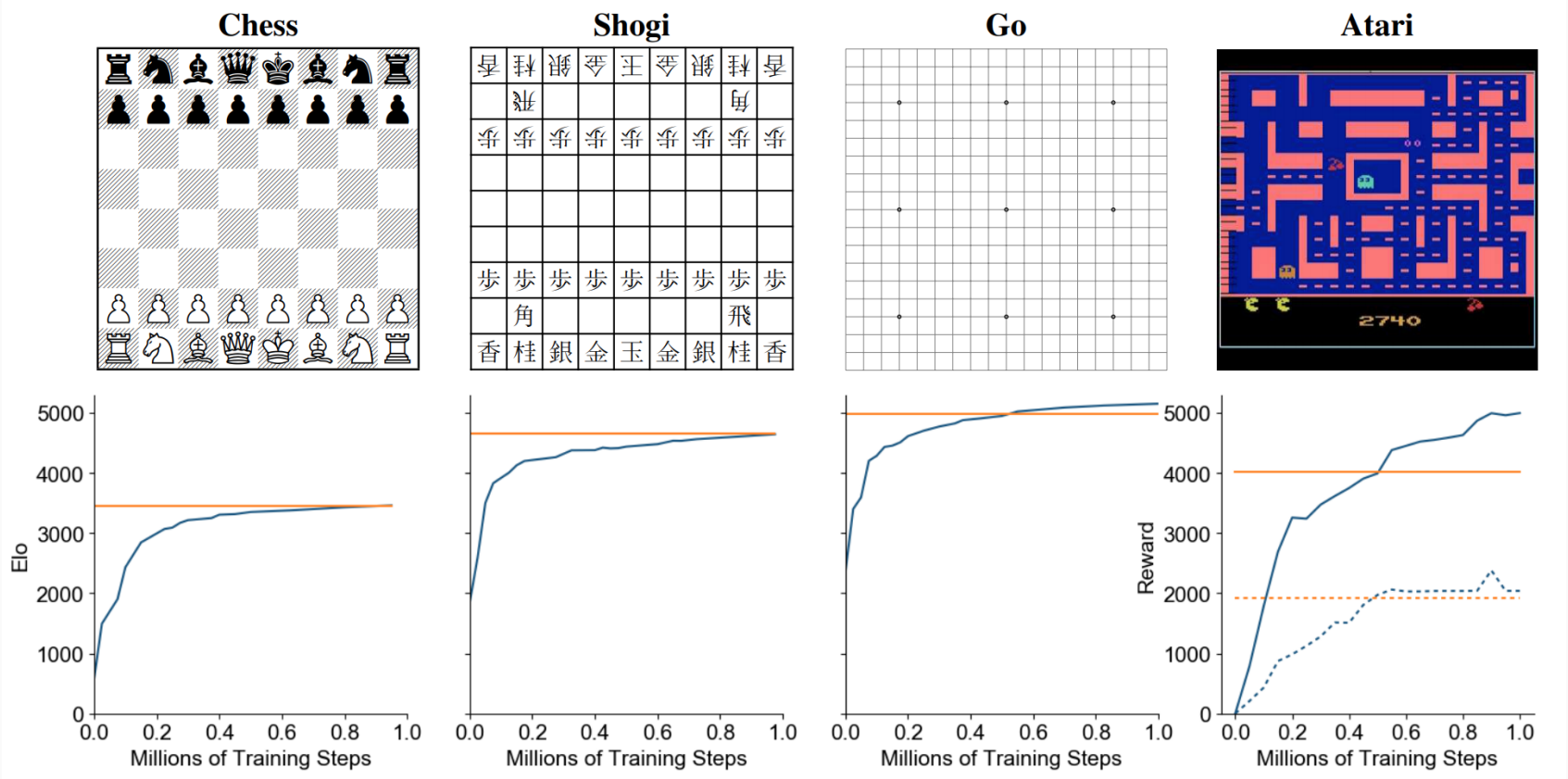
- В конце 2019 года появилась работа о MuZero, которая ещё лучше (Schrittwieser et al., 2019)!



- Как они смогли? Ключевое слово здесь...

MuZero

- ...планирование! Не зря мы его изучали.
- MuZero поддерживает модель динамики MDP, обучая её тоже по ходу дела



- Давайте посмотрим в подробностях...

MuZero

- MuZero предсказывает на каждом шаге t три вещи для каждого будущего шага $k = 1, \dots, K$ на основе наблюдений $o_1, \dots, o_t$:
  - стратегию $\mathbf{p}_t^k \approx \pi\left(a_{t+k+1} | o_1, \dots, o_t, a_{t+1}, \dots, a_{t+k}\right)$;
  - функцию $v_t^k \approx \mathbb{E}\left[R_{t+k+1} + \gamma R_{t+k+2} + \dots \mid o_1, \dots, o_t, a_{t+1}, \dots, a_{t+k}\right]$;
  - награду $r_t^k \approx R_{t+k}$.

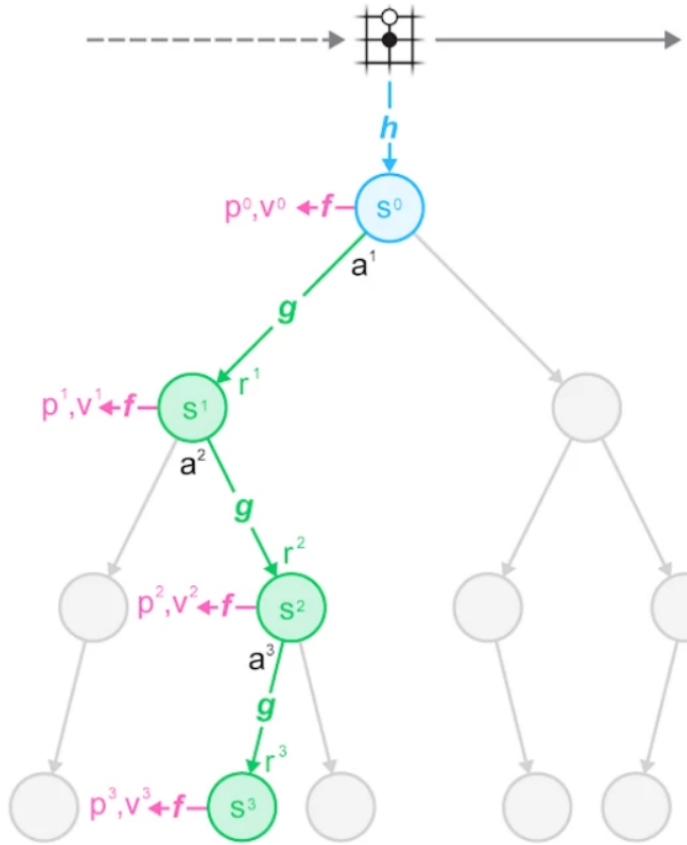
MuZero

- Внутри это представлено тремя функциями:
  - функция динамики процесса: $r^k, s^k = g_\theta(s^{k-1}, a^k)$, но здесь s^k — не состояние среды, а внутреннее состояние модели;
  - функция предсказания $\mathbf{p}^k, v^k = f_\theta(s^k)$, как в AlphaZero;
  - функция представления $s^0 = h_\theta(o_1, \dots, o_t)$.

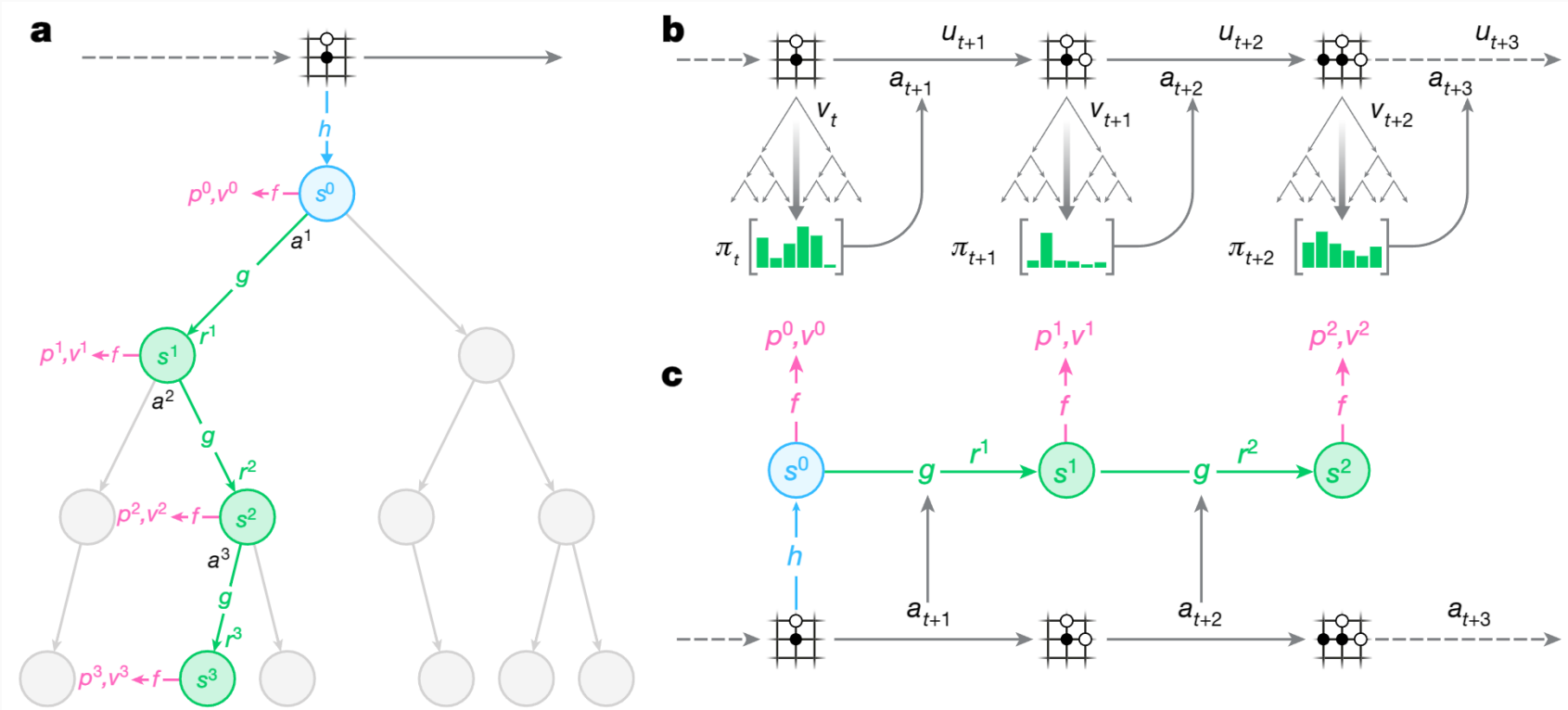
representation $h_\theta(o_1, .., o_t) = s^0$

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

dynamics $g_\theta(s^{k-1}, a^k) = r^k, s^k$

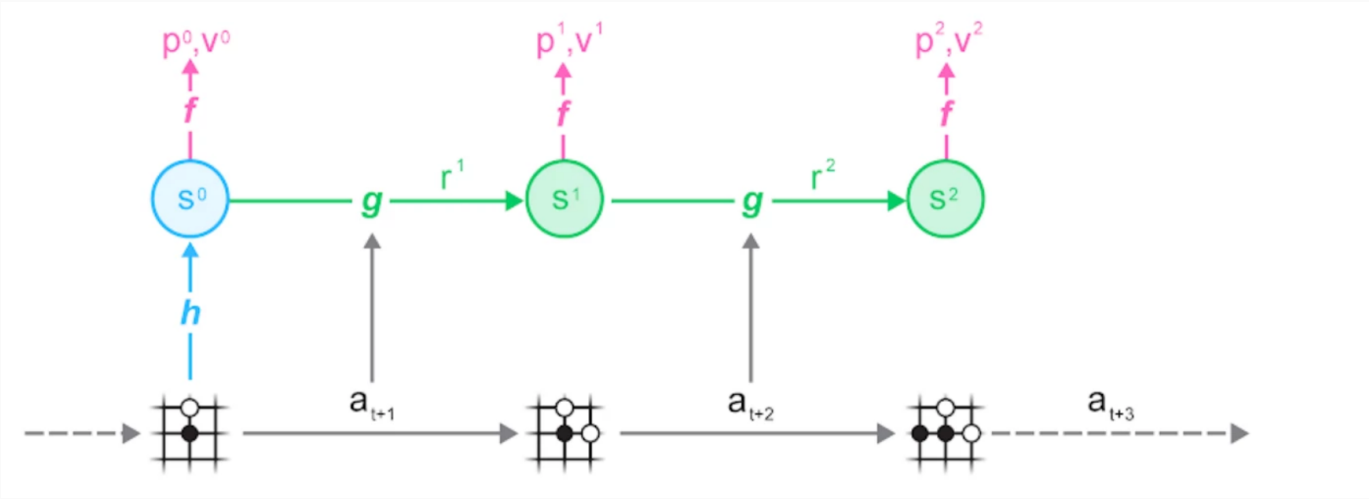


- Когда есть такое представление, можно искать по гипотетическим будущим траекториям $a^1, \ldots, a^k$.



MuZero

- При обучении разворачиваем на несколько шагов вперёд (чтобы динамику процесса тоже обучать) и используем или TD, или MC, сравнивая предсказанные и реальные значения, награды и действия

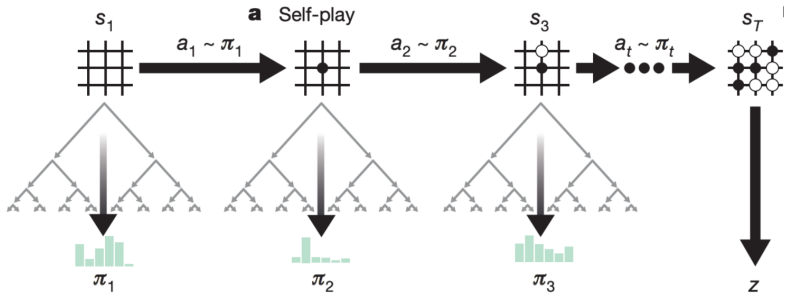


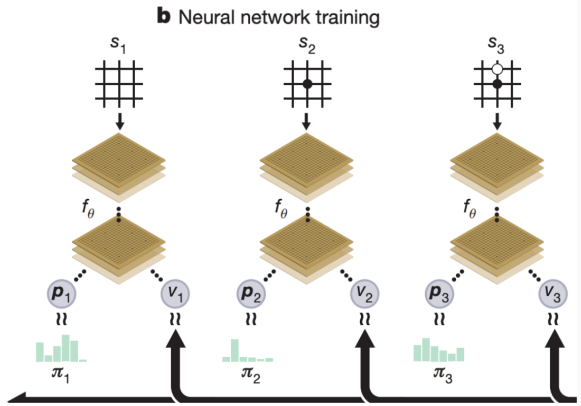
- Общая функция потерь

$$L_t(\theta) = \sum_{k=0}^{K} \left(L_r(u_{t+k}, r_t^k) + L_v(z_{t+k}, v_t^k) + L_p(\pi_{t+k}, \mathbf{p}_t^k) \right) + c\|\theta\|^2.$$

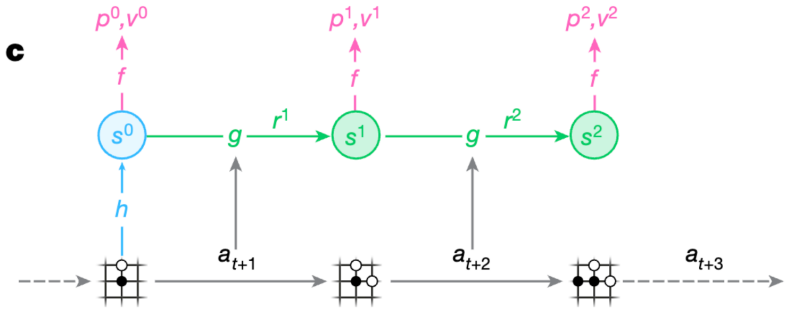
MuZero

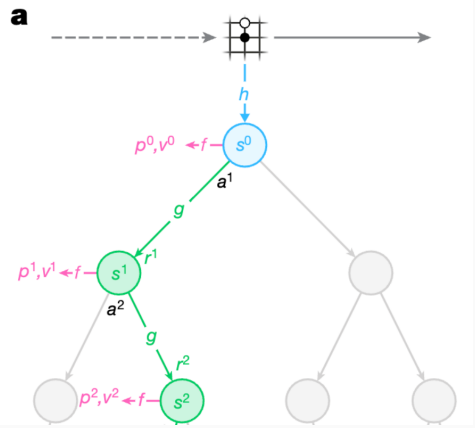
- Итого:





AlphaZero: the RL agent plays in an environment, using MCTS to look ahead to improve the policy during training



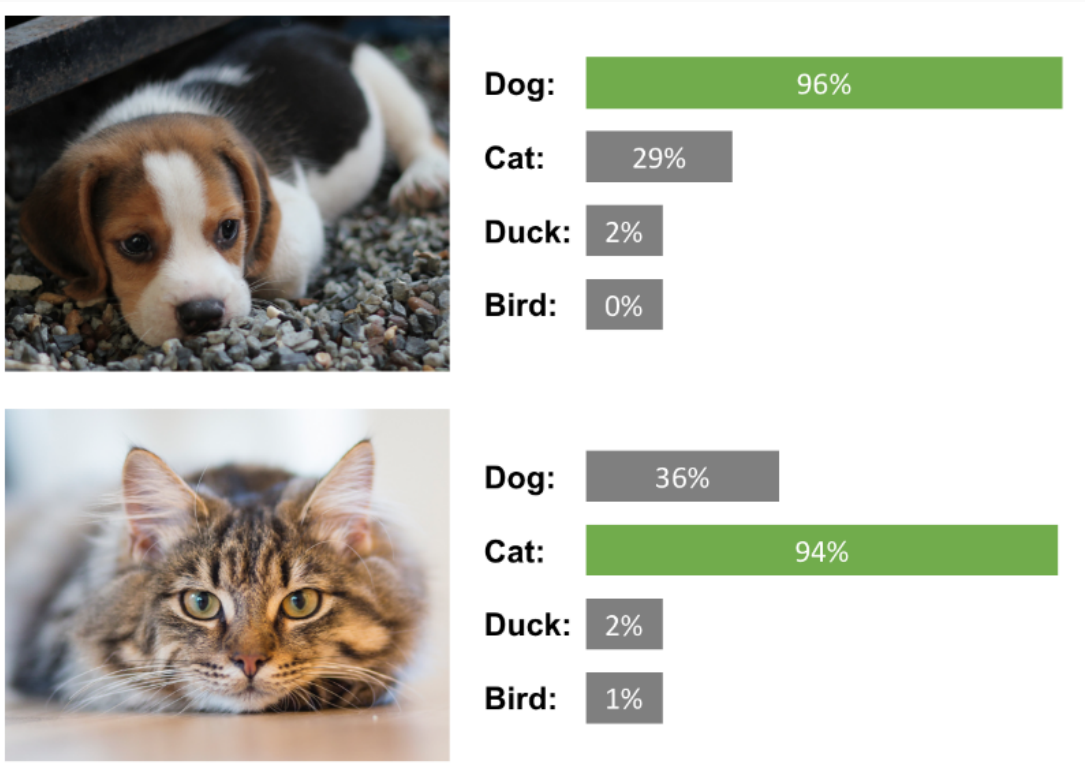


MuZero: the RL agent predicts future states by "dreaming" about possible transitions and bases its MCTS search on it

Порождающие и дискриминирующие модели в машинном обучении

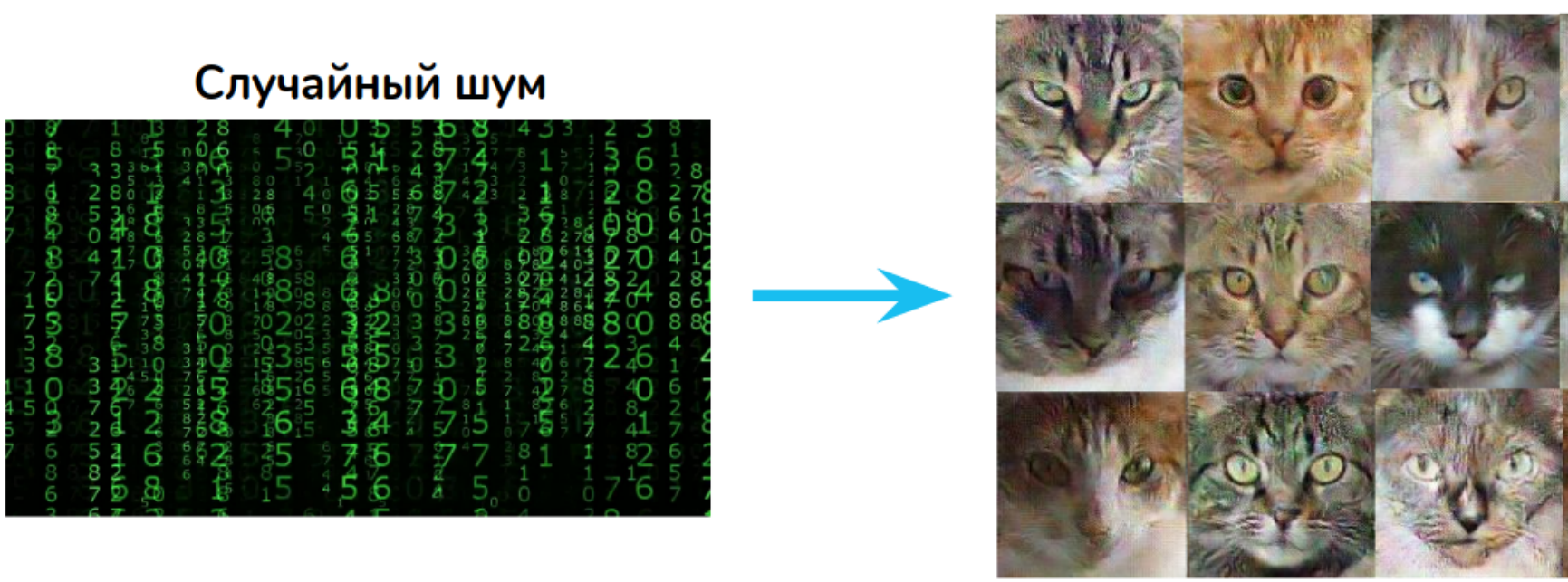
Порождающие модели

- Итак, мы уже видели глубокое обучение и с учителем, и без.
- Но можем ли мы породить что-то новое?
- Модели бывают дискриминирующие (discriminative), которые моделируют $p(y \mid \mathbf{x})$:



Порождающие модели

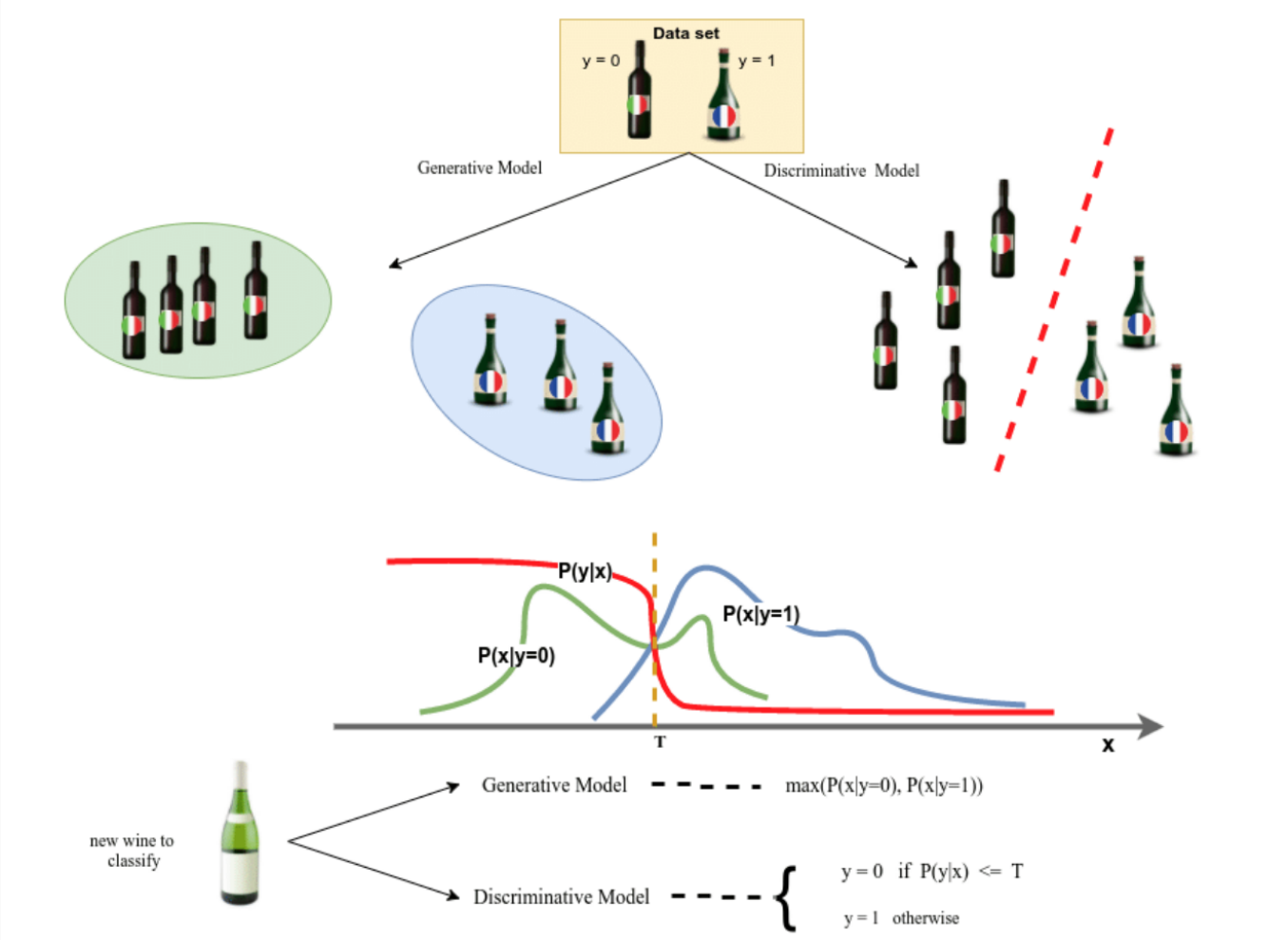
- А бывают порождающие (generative), которые моделируют $p(\mathbf{x}, y)$, и из них тогда можно сэмплировать:



- Давайте чуть подробнее. Во-первых, зачем это надо?
- $p(\mathbf{x}, y) = p(y \mid \mathbf{x})p(\mathbf{x})$, конечно, но разница есть.

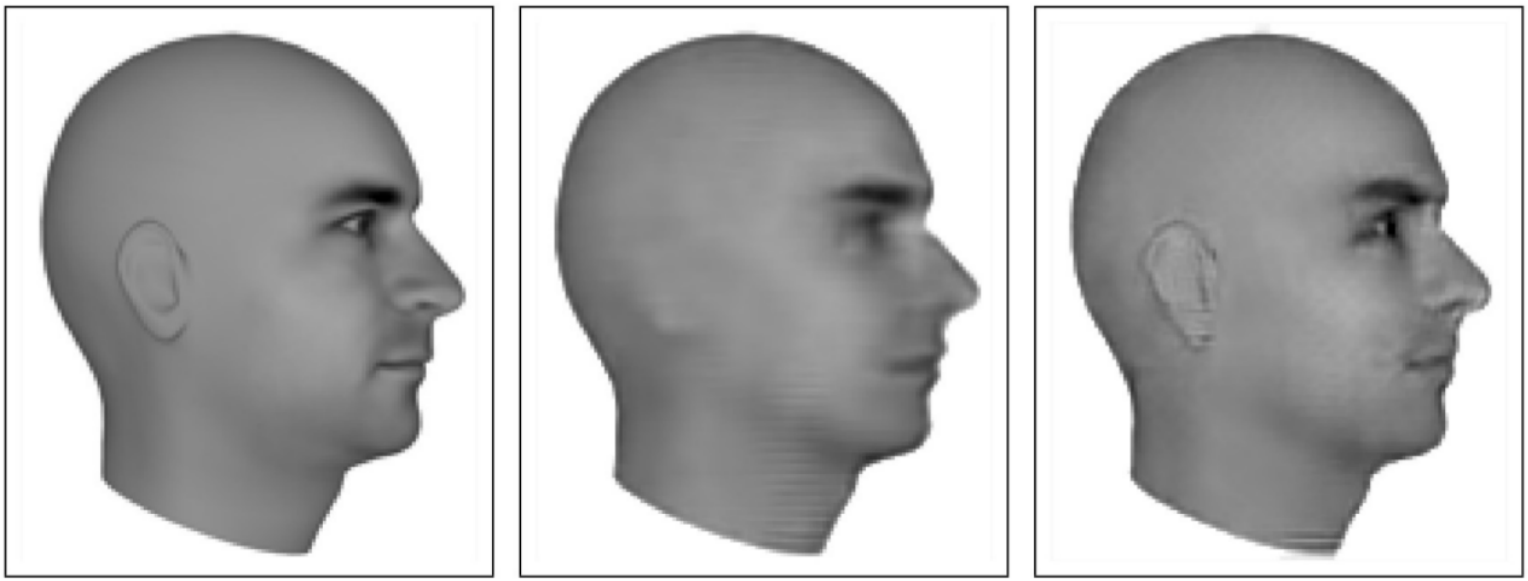
Порождающие модели

- Иллюстрация:



Порождающие модели

- Порождающие модели:
  - проверяют, насколько мы поняли распределение;
  - могут обучаться с недостатком данных и без разметки – semi-supervised learning (очень важно!);
  - могут обучаться мультимодальным выходам, когда есть несколько правильных ответов (см. ниже);
  - могут служить моделями окружающего мира в обучении с подкреплением (об этом позже);
  - ну и просто иногда действительно нужно именно порождать.



Порождающие модели

- Обычно порождающие модели максимизируют правдоподобие:

$$\theta^* = \arg\max \prod_{\mathbf{x} \in D} p_{\text{model}}(\mathbf{x}; \theta) = \arg\max \sum_{\mathbf{x} \in D} \log p_{\text{model}}(\mathbf{x}; \theta).$$

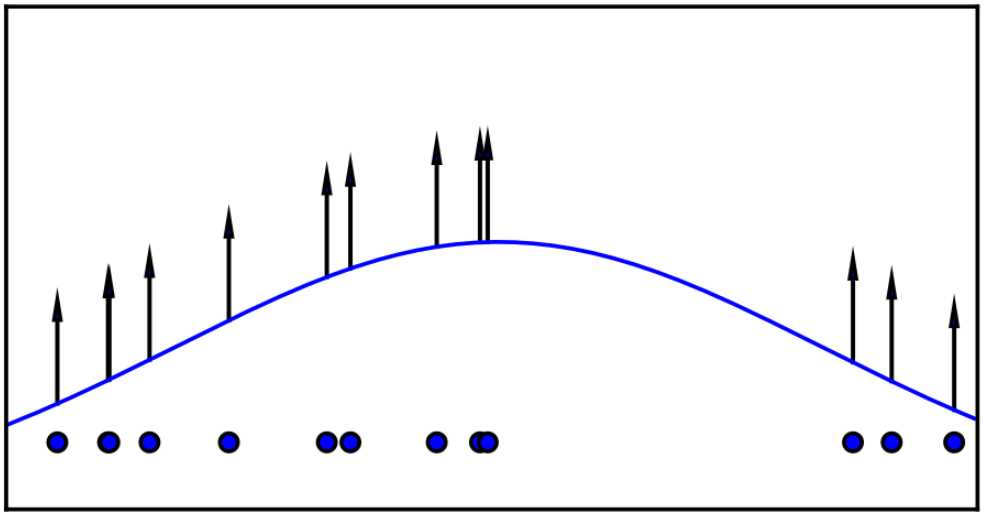
- Важный другой взгляд – это то же самое, что минимизировать KL между «распределением данных» p_{data} и p_{model}:

$$\theta^* = \arg\min \mathrm{KL}\left(p_{\text{data}} \| p_{\text{model}}\right) = \arg\min - \int p_{\text{data}}(x) \ln \frac{p_{\text{model}}(x)}{p_{\text{data}}(x)} \mathrm{d}x,$$

потому что p_{data} для нас дано в ощущениях через D, и это всё равно что дискретное равномерное распределение.

Порождающие модели

- Иначе говоря, точки данных «тянут» вверх распределение p_{model}:



$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{\theta})$$

- Только в большой размерности и не просто с гауссианами всё куда сложнее...
- Большая разница в том, предполагаем ли мы, что можем явно определить и посчитать плотность p_{model}.

Порождающие и дискриминативные модели

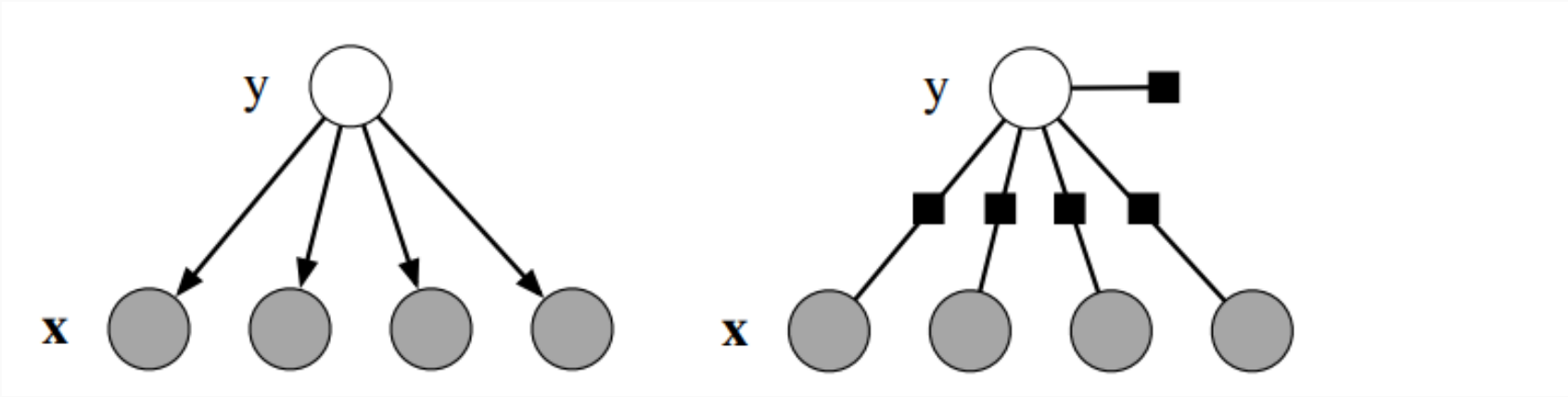
- Давайте рассмотрим пару примеров, которые дают понимание разницы между порождающими и дискриминативными моделями.
- Заодно расскажу вам, что такое CRF, всё польза. :)
- Неожиданный заход: в чём разница между наивным Байесом и логистической регрессией?..

Порождающие и дискриминативные модели

- Наивный Байес – это порождающая модель:

$$p(y, \mathbf{x}) = p(y) \prod_{k=1}^{K} p(x_k \mid y).$$

- Предположение в том, что признаки независимы.



Порождающие и дискриминативные модели

- В логистической регрессии предположение в том, что $\log p(y \mid \mathbf{x})$ – это линейная функция от $\mathbf{x}$:

$$p(y \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{\theta_y + \sum_{k=1}^{K} \theta_{y,k} x_k},$$

  где θ – веса, θ_y – свободный член (похож на $p(y)$), $Z(\mathbf{x})$ – нормировочная константа.
- Можно, кстати, переписать в чуть более общем виде через признаки:

$$p(y \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{\sum_{k=1}^{K} \theta_k f_k(y, \mathbf{x})},$$

  где $f_{k,j}(y, \mathbf{x}) = [k = y] x_j$ для векторов весов признаков и $f_k(y, \mathbf{x}) = [k = y]$ для bias weights.
- Но из наивного Байеса тоже можно получить $p(y \mid \mathbf{x})$, ведь $p(y, \mathbf{x}) = p(\mathbf{x}) p(y \mid \mathbf{x})$...

Порождающие и дискриминативные модели

- Более того, действительно получится то же самое. У них одно и то же пространство гипотез, и одно можно перевести в другое:
  - если обучать наивного Байеса максимизировать условное правдоподобие, получится логистическая регрессия;
  - а если интерпретировать логистическую регрессию как порождающую модель с

$$p(y, \mathbf{x}) = \frac{\exp\left(\sum_{k=1}^{K} \theta_k f_k(y, \mathbf{x})\right)}{\sum_{y', \mathbf{x}'} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y', \mathbf{x}')\right)},$$

    то получится тот же классификатор, что из наивного Байеса.
- Наивный Байес и логистическая регрессия образуют generative-discriminative pair.

Порождающие и дискриминативные модели

- Т.е. единственная разница в том, что наивный Байес – порождающая модель, а логистическая регрессия – дискриминативная.
- Если бы мы могли получить идеальную модель $p^*(\mathbf{y}, \mathbf{x}) = p^*(\mathbf{y})p^*(\mathbf{y} \mid \mathbf{x})$, то не было бы никакой разницы.
- Но на практике есть разница, оценивать ли $p(\mathbf{y})p(\mathbf{x} \mid \mathbf{y})$, а затем вычислять из этого $p(\mathbf{y} \mid \mathbf{x})$, или сразу напрямую оценивать $p(\mathbf{y} \mid \mathbf{x})$.
- Порождающие модели могут больше, но у дискриминативных больше свободы.

Порождающие и дискриминативные модели

- Например, пусть есть порождающая модель с параметрами θ:

$$p_g(\mathbf{y}, \mathbf{x}; \theta) = p_g(\mathbf{y}; \theta) p_g(\mathbf{x} \mid \mathbf{y}; \theta).$$

- Можно её переписать как $p_g(\mathbf{y}, \mathbf{x}; \theta) = p_g(\mathbf{x}; \theta) p_g(\mathbf{x} \mid \mathbf{y}; \theta)$, где $p_g(\mathbf{x}; \theta) = \sum_{\mathbf{y}} p_g(\mathbf{y}, \mathbf{x}; \theta)$, $p_g(\mathbf{y} \mid \mathbf{x}; \theta) = p_g(\mathbf{y}, \mathbf{x}; \theta) / p_g(\mathbf{x}; \theta)$.
- А соответствующая ей дискриминативная модель должна иметь априорное распределение $p(\mathbf{x})$, которое может получиться из неё, т.е. $p(\mathbf{x}) = p_c(\mathbf{x}; \theta') = \sum_{\mathbf{y}} p_g(\mathbf{y}, \mathbf{x}; \theta')$, и условное распределение, которое может получаться как $p_c(\mathbf{y} \mid \mathbf{x}; \theta) = p_g(\mathbf{y}, \mathbf{x}; \theta) / p_g(\mathbf{x}; \theta)$:

$$p_c(\mathbf{y}, \mathbf{x}) = p_c(\mathbf{x}; \theta') p_c(\mathbf{y} \mid \mathbf{c}; \theta).$$

- Разница в том, что теперь не обязательно $\theta = \theta'$, и дискриминативная модель более выразительна.

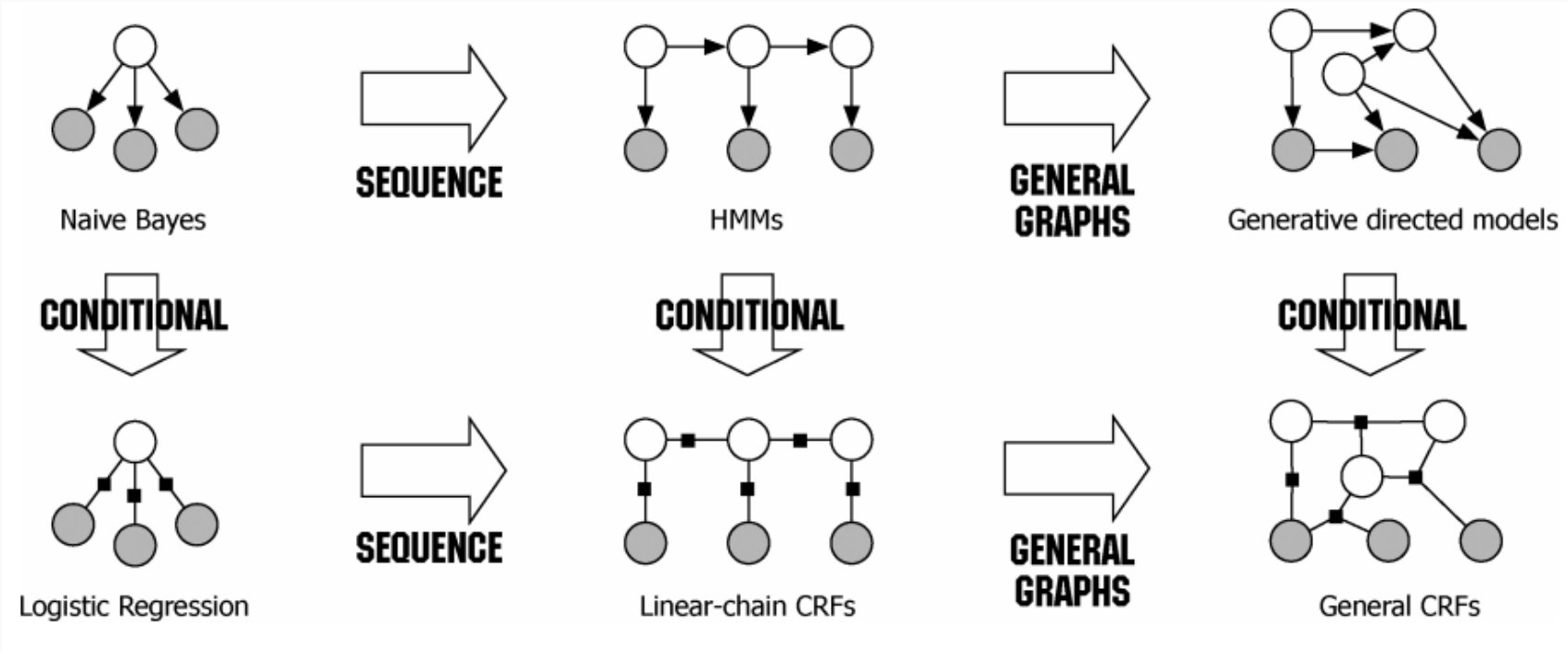
Порождающие и дискриминативные модели

- В результате мы будем хуже моделировать $p(\mathbf{x})$ (на которое в классификации нам наплевать), и лучше подходить к $p(\mathbf{y} \mid \mathbf{x})$ (но и риск оверфиттинга выше).
- Другой пример порождающей модели – скрытая марковская модель: моделируем наблюдаемые $X = \{x_t\}_{t=1}^T$, предполагая, что есть скрытые состояния $Y = \{y_t\}_{t=1}^T$, и делаем предположения о независимости:

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^{T} p(y_t \mid y_{t-1}) p(x_t \mid y_t).$$

Порождающие и дискриминативные модели

- Так вот, CRF – это дискриминативные модели, и линейная CRF – это дискриминативная модель, соответствующая HMM:



Линейные CRF

- Чтобы прийти к linear chain CRF, сначала перепишем HMM в другом виде:

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \prod_{t=1}^{T} \exp\left(\sum_{i,j \in S} \theta_{i,j} [y_t = i][y_{t-1} = j] + \right.$$
$$\left. + \sum_{i \in S} \sum_{o \in O} \mu_{oi} [y_t = i][x_t = o] \right),$$

где S – множество скрытых состояний, O – множество наблюдаемых, а в терминах HMM

$$\theta_{ij} = \log p(y_t = i \mid y_{t-1} = j),$$
$$\mu_{oi} = \log p(x = o \mid y = i), \qquad Z = 1.$$

- Более того, верно и обратное: если распределение раскладывается как выше написано, то это HMM.

Линейные CRF

- И здесь тоже, как в логистической регрессии, можно ввести функции признаков (feature functions) $f_k(y_t, y_{t-1}, x_t)$:

$$f_{ij}(y, y', x) = [y = i][y' = j] \text{ для каждого перехода,}$$
$$f_{io}(y, y', x) = [y = i][x = o] \text{ для каждой наблюдаемой.}$$

- И теперь (k бегает и по (i, j), и по (i, o))

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t)\right).$$

- Для выбранных f_k это в точности эквивалентно HMM.

Линейные CRF

- Можно теперь выразить условное распределение, и это уже будет linear chain CRF:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{\prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t)\right)}{\sum_{\mathbf{y}'} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y'_t, y'_{t-1}, x_t)\right)}.$$

- В общем виде linear chain CRF именно так и определяется:

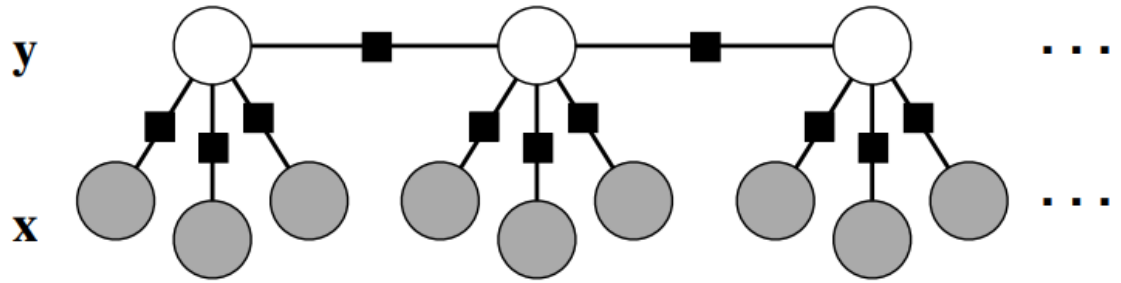
$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right),$$

т.е. просто разрешим общий вид признаков и, может быть, несколько компонент у $\mathbf{x}_t$.

Линейные CRF

- Linear chain CRF:

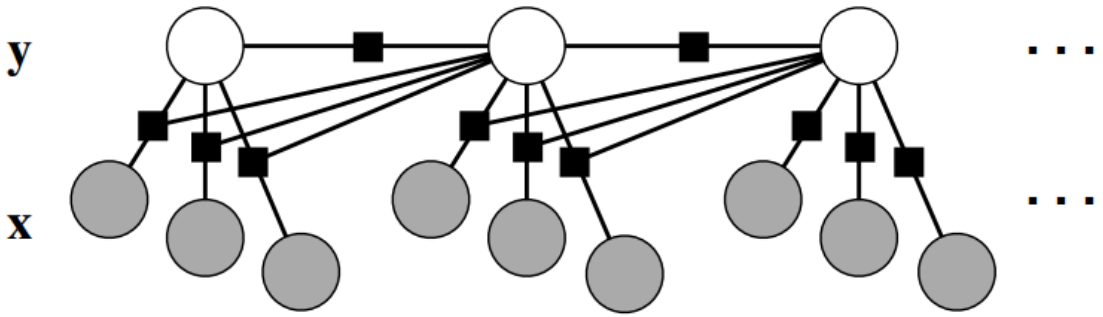
$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right).$$



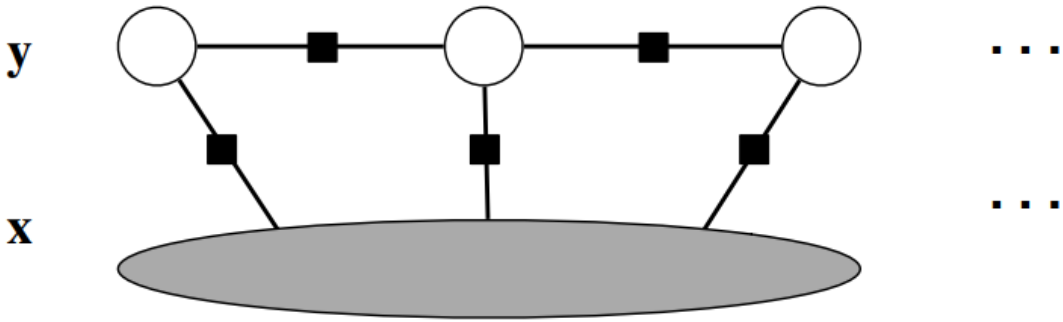
- Разложение $p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, \mathbf{x}_t)$, со специальным видом Ψ_t.
- Задача – оценить параметры θ из данных.
- Часто в NLP применяется: частеречная разметка, named entity recognition и т.п.

Линейные CRF

- Более общий случай, чем HMM; например, признак перехода может зависеть от наблюдаемой, просто добавим $f = [y_t = j][y_{t-1} = i][x_t = o]$:



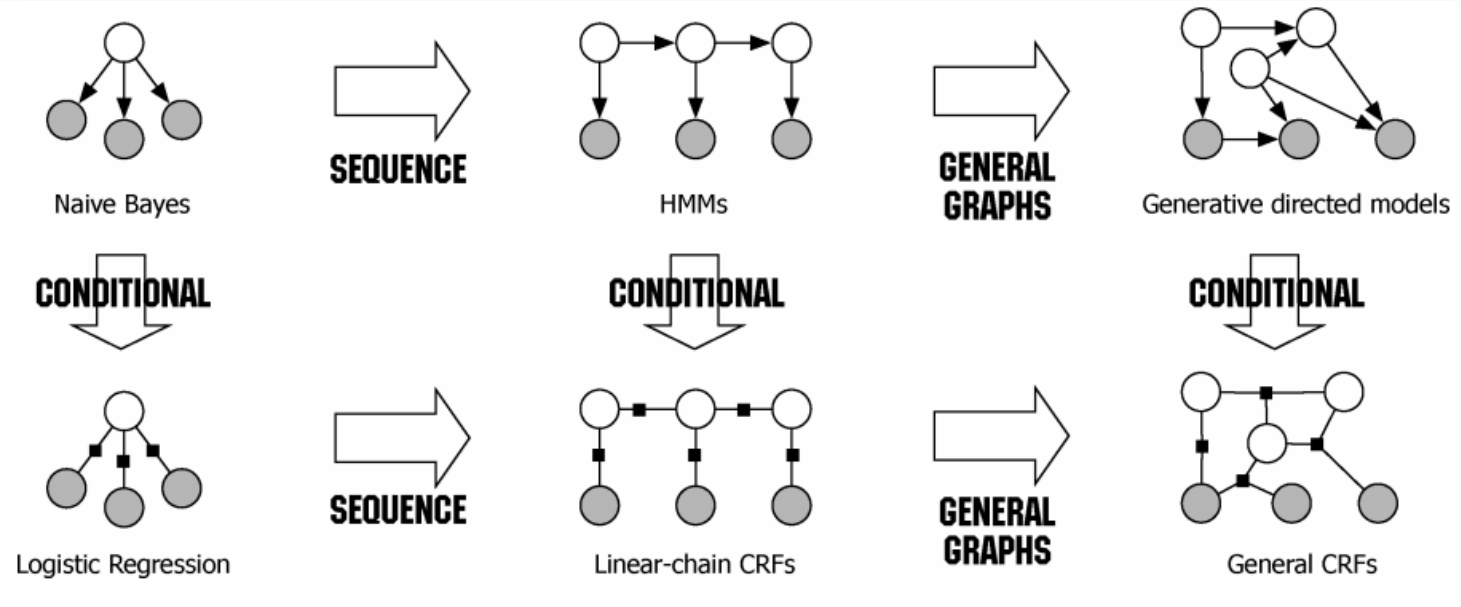
- А может зависеть от всех наблюдений сразу, $f_k(y_t, y_{t-1}, \mathbf{x})$:



Линейные CRF

- А совсем общая CRF – это модель, в которой условное распределение $p(\mathbf{y} \mid \mathbf{x})$ раскладывается по какому-то графу, не обязательно цепи:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^{A} \Psi_a(\mathbf{y}_a, \mathbf{x}_a).$$



- В чём отличие от просто общей формы ненаправленной модели?..

Линейные CRF

- О форме факторов:
  - обычно мы обучаем CRF с линейными весами: $\log \Psi_a(\mathbf{y}_a, \mathbf{x}_a) = \exp\left(\sum_{k=1}^{K(A)} \theta_{ak} f_{ak}(\mathbf{y}_a, \mathbf{x}_a)\right)$, у каждого фактора свои веса;
  - если $\mathbf{x}$ и $\mathbf{y}$ дискретные, то это вообще не ограничивает общности (возьмём f_{ak} по всем комбинациям);
  - часто параметры так или иначе связаны; например, в линейной цепи мы считаем, что $\Psi_t(y_t, y_{t-1}, \mathbf{x}_t)$ одинаковые по всем t;
  - можно определить clique templates для этого:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{C_p \in C} \prod_{\Psi_c \in C_p} \Psi_c(\mathbf{y}_c, \mathbf{x}_c; \theta_p), \text{ где}$$

$$\Psi_c(\mathbf{y}_c, \mathbf{x}_c; \theta_p) = \exp\left(\sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\mathbf{x}_c, \mathbf{y}_c)\right);$$

  - есть много расширений и вариантов с разными формами parameter tying.

Линейные CRF

- О признаках:
  - для дискретных наблюдаемых часто имеют смысл label-observation features:

$$f_{pk}(\mathbf{y}_c, \mathbf{x}_c) = [\mathbf{y}_c = \mathbf{y}'_c] q_{pk}(\mathbf{x}_c);$$

    этот признак не ноль только для конкретной конфигурации $\mathbf{y}'_c$, но для неё может быть любым; observation functions $q_{pk}(\mathbf{x}_c)$ – например, «слово x_t начинается с большой буквы»;
  - unsupported features: бывает, что признаки вообще не встречаются в данных, особенно когда их очень много (миллионы); например, «$x_t =$ “если” и $y_t =$ “CityName”»; но они всё равно могут быть полезны, если дают отрицательный вес неподходящему ответу;

Линейные CRF

- О признаках:
  - edge-observation features: когда фактор перехода зависит от всех функций наблюдения: «$x_t =$ “Новый” и $y_t =$ “CityName” и $y_{t-1} =$ “CityName”»;
  - если это слишком много, можно заменить на node-observation features: отдельно «$x_t =$ “Новый” и $y_t =$ “CityName”», отдельно «$y_t =$ “CityName” и $y_{t-1} =$ “CityName”»;

Edge-observation features:

$$f(y_t, y_{t-1}, \mathbf{x}_t) = q_m(\mathbf{x}_t)\mathbf{1}_{\{y_t=y\}}\mathbf{1}_{\{y_{t-1}=y'\}} \qquad \forall y, y' \in \mathcal{Y}, \forall m$$

$$f(y_t, \mathbf{x}_t) = q_m(\mathbf{x}_t)\mathbf{1}_{\{\mathbf{y}_t=y\}} \qquad \forall y \in \mathcal{Y}, \forall m$$

Node-observation features:

$$f(y_t, y_{t-1}, \mathbf{x}_t) = \mathbf{1}_{\{y_t=y\}}\mathbf{1}_{\{y_{t-1}=y'\}} \qquad \forall y, y' \in \mathcal{Y}$$

$$f(y_t, \mathbf{x}_t) = q_m(\mathbf{x}_t)\mathbf{1}_{\{y_t=y\}} \qquad \forall y \in \mathcal{Y}, \forall m$$

Линейные CRF

- О признаках:
  - отдельные метки Start/Stop для первых/последних элементов последовательности.
  - важно не забывать, что в $f(y_t, y_{t-1}, \mathbf{x}_t)$ среди $\mathbf{x}_t$ могут быть разные x: «x_{t+2} = “Уэльс” и y_t = “StateName”» (если есть такая фича, это, видимо, не Англия);
  - можно использовать менее общие признаки как регуляризатор (backoff): например, полезно включать и $\Psi_t(y_t, y_{t-1}, \mathbf{x}_t)$, и $\Psi_t(y_t, \mathbf{x}_t)$, хотя формально уже вторые не нужны;
  - можно добавлять как признаки результаты других моделей (более простых, или обученных на других датасетах и т.п.);
  - input-dependent structure: можно разрешать структуре графа зависеть от $\mathbf{x}$; например, можно поощрять, чтобы одно и то же слово имело одну и ту же метку.

Примеры

- Пример – named entity recognition (NER):

In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space – Alibaba GPE , Baidu ORG , and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space . The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the 'future AI PERSON platforms'. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL , with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE .

To further elaborate on the geographical trends, North America LOC has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG , IBM ORG , and Microsoft ORG .

- Например, датасет CoNLL2003 – люди (PER), организации (ORG), места (LOC), другие (MISC), и никаких (O). Девять видов меток:

$$\mathcal{Y} = \{\text{B-PER}, \text{I-PER}, \text{B-LOC}, \text{I-LOC}, \text{B-ORG}, \text{I-ORG}, \text{B-MISC}, \text{I-MISC}, \text{O}\}$$

Примеры

- Можно построить модель так:

t	y_t	$\mathbf{x}_t$
0	B-ORG	U.N.
1	O	official
2	B-PER	Ekeus
3	O	heads
4	O	for
5	B-LOC	Baghdad

- И ввести признаки label-label и label-word:

$$f_{ij}^{LL}(y_t, y_{t-1}, \mathbf{x}_t) = [y_t = i][y_{t-1} = j],$$
$$f_{iv}^{LW}(y_t, y_{t-1}, \mathbf{x}_t) = [y_t = i][\mathbf{x}_t = v].$$

- Всего 81 label-label и $9 \times 21249 = 191241$ label-word (в CoNLL2003), но большая часть бесполезны, конечно.

Примеры

- Но в реальности надо добавить выразительности:

t	y_t	$\mathbf{x}_t$
0	B-ORG	($\langle$START$\rangle$, U.N., official)
1	O	(U.N., official, Ekeus)
2	B-PER	(official, Ekeus, heads)
3	O	(Ekeus, heads, for)
4	O	(heads, for, Baghdad)
5	B-LOC	(for, Baghdad, $\langle$END$\rangle$)

- Три вида label-word теперь, и ещё label-observation можно добавить:

$$
\begin{aligned}
f_{iv}^{LW0}(y_t, y_{t-1}, \mathbf{x}_t) &= [y_t = i][\mathbf{x}_{t0} = v], \\
f_{iv}^{LW1}(y_t, y_{t-1}, \mathbf{x}_t) &= [y_t = i][\mathbf{x}_{t1} = v], \\
f_{iv}^{LW2}(y_t, y_{t-1}, \mathbf{x}_t) &= [y_t = i][\mathbf{x}_{t2} = v], \\
f_{ib}^{LO}(y_t, y_{t-1}, \mathbf{x}_t) &= [y_t = i] q_b(\mathbf{x}_t).
\end{aligned}
$$

Примеры

- Например, в (McCallum, Li, 2003):

W=v	$w_t = v$	$\forall v \in \mathcal{V}$
T=j	part-of-speech tag for w_t is j (as determined by an automatic tagger)	$\forall$POS tags j
P=I−j	w_t is part of a phrase with syntactic type j (as determined by an automatic chunker)	
Capitalized	w_t matches `[A-Z][a-z]+`	
Allcaps	w_t matches `[A-Z][A-Z]+`	
EndsInDot	w_t matches `[^\.]+.*\.`	
	w_t contains a dash	
	w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`	
Acro	w_t matches `[A-Z][A-Z\\.]*\\.[A-Z\\.]*`	
Stopword	w_t appears in a hand-built list of stop words	
CountryCapital	w_t appears in list of capitals of countries	
⋮	many other lexicons and regular expressions	
$q_k(\mathbf{x}, t+\delta)$ for all k and $\delta \in [-1, 1]$		

Примеры

- Например, в (McCallum, Li, 2003):

t	y_t	Active observation functions
1	B-ORG	P=I-NP@1 W=⟨START⟩@-1 INITCAP P=I-NP T=NNP T=NN@1 ACRO ENDSINDOT W=official@1 W=U.N.
2	O	P=I-NP@1 INITCAP@1 P=I-NP T=JJ@1 CAPITALIZED@1 T=NNP@-1 P=I-NP@-1 INITCAP@-1 T=NN ENDSINDOT@-1 ACRO@-1 W=official W=U.N.@-1W=Ekeus@1
3	B-PER	P=I-NP@1 INITCAP P=I-NP P=I-NP@-1 CAPITALIZED T=JJ T=NN@-1 T=NNS@1 W=official@-1 W=heads@1 W=Ekeus
4	O	P=I-NP P=I-NP@-1 INITCAP@-1 STOPWORD@1 T=JJ@-1 CAPITALIZED@-1 T=IN@1 P=I-PP@1 T=NNS W=for@1 W=heads W=Ekeus@-1
5	O	T=NNP@1 P=I-NP@1 INITCAP@1 LOC@1 CAPITALIZED@1 P=I-NP@-1 STOPWORD COUNTRYCAPITAL@1 P=I-PP T=IN T=NNS@-1 W=for W=Baghdad@1 W=heads@-1
6	B-LOC	INITCAP P=I-NP T=NNP CAPITALIZED STOPWORD@-1 T=.@1 P=O@1 PUNC@1 W=⟨END⟩@1 COUNTRYCAPITAL T=IN@-1 P=I-PP@-1 W=for@-1 W=Baghdad

Примеры

- Теперь к нашему основному примеру – сегментации, т.е. разметке картинки на семантические части.
- Формально $\mathbf{x} = (x_1, \dots, x_T)$, картинка размера $\sqrt{T} \times \sqrt{T}$. Предсказываем $\mathbf{y} = (y_1, \dots, y_T)$, сегментацию.
- Куча признаков в классическом CV:
  - гистограммы интенсивностей пикселей, например в окрестности 5×5;
  - градиенты картинки;
  - texton-признаки, SIFT-признаки и т.п. (может быть, позже).
- Как их всех в CRF добавить?

Примеры

- Базовая идея – по-прежнему MRF:

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{(i,j) \in N} \Psi(y_i, y_j), \quad p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}) \prod_{i=1}^{T} p(x_i \mid y_i).$$

- Но трудно добавить признаки по подмножествам пикселей с предыдущего слайда, потому что $p(\mathbf{x} \mid \mathbf{y})$ получается сложная.
- А в CRF всё с этим проще: мы разрешаем факторам зависеть от произвольных признаков (observation functions) всей картинки.

Примеры

- Например, пусть $q(x_i)$ – вектор признаков участка вокруг x_i, $\nu(x_i, x_j)$ – признаки пары участков (их похожесть-непохожесть), например набор элементов матрицы $q(x_i)q(x_j)^\top$, и из них определяем CRF с такими observation functions:

$$f_m(y_i, x_i) = \mathbf{1}_{\{y_i=m\}} q(x_i) \quad \forall m \in \{0, 1\}$$

$$g_{m,m'}(y_i, y_j, x_i, x_j) = \mathbf{1}_{\{y_i=m\}} \mathbf{1}_{\{y_j=m'\}} \nu(x_i, x_j) \quad \forall m, m' \in \{0, 1\}$$

$$f(y_i, x_i) = \begin{pmatrix} f_0(y_i, x_i) \\ f_1(y_i, x_i) \end{pmatrix}$$

$$g(y_i, y_j, x_i, x_j) = \begin{pmatrix} g_{00}(y_i, y_j, x_i, x_j) \\ g_{01}(y_i, y_j, x_i, x_j) \\ g_{10}(y_i, y_j, x_i, x_j) \\ g_{11}(y_i, y_j, x_i, x_j) \end{pmatrix}$$

Примеры

- Конкретный пример функций:

$$\nu(x_i, x_j) = \exp\left(-\beta(x_i - x_j)^2\right),$$
$$g(y_i, y_j, x_i, x_j) = \left[y_i \neq y_j\right]\nu(x_i, x_j).$$

- И получаем CRF

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i=1}^{T} \theta^\top f(y_i, x_i) + \alpha \sum_{(i,j)\in N} \lambda^\top g(y_i, y_j, x_i, x_j)\right),$$

  где α, θ, λ – параметры.

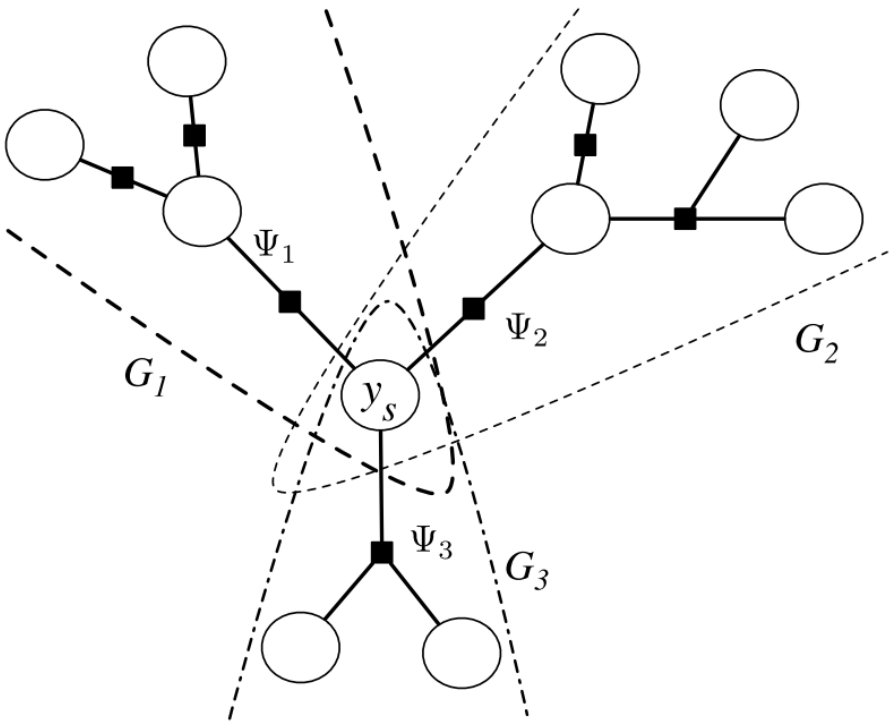
- Теперь встаёт вопрос: а как эту всю красоту обучить-то?

Вывод в CRF

- Вывод в CRF состоит из двух похожих задач:
  - применить CRF, т.е. найти метки $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$;
  - чтобы оценить параметры, надо найти маргиналы $p(y_t \mid \mathbf{x})$ и $p(y_t, y_{t-1} \mid \mathbf{x})$ (потом увидим почему).
- Всё это чертовски напоминает HMM для linear chain CRF и просто вывод в графических моделях для CRF общего вида.

Вывод в CRF

- Linear chain CRF – те же forward-backward, Viterbi, $\alpha_t(j)$, $\beta_t(j)$, $\gamma_t(j)$...
- В общем виде – тот же belief propagation



Вывод в CRF

- А когда так не получается, те же MCMC-методы, вариационные приближения...
- Но с CRF надо иметь в виду, что:
  - есть обычно большая разреженность в признаках и значениях факторов, которую надо бы использовать;
  - второе замечание – как справиться с underflow при выводе? можно считать в логарифмах:

$$\log \alpha_t(j) = \bigoplus_{i \in S} \left(\log \Psi_t(j, i, \mathbf{x}_t) + \log \alpha_{t-1}(i) \right),$$

  но разве это проще? вроде бы при подсчёте $a \oplus b = \log(e^a + e^b)$ всё равно тот же underflow? hint: нет.

Вывод в CRF

- А как оценивать параметры CRF $\theta = \{\theta_k\}_{k=1}^K$?
- Максимальное правдоподобие: по данным $D = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N$ надо максимизировать условное правдоподобие

$$\ell(\theta) = \sum_{i=1}^{N} \log p(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}; \theta).$$

- Для linear chain CRF, например,

$$\ell(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}).$$

- Можно ещё добавить регуляризацию, L_2 или L_1 на θ.
- Как максимизировать?

Вывод в CRF

- Возьмём производную:

$$\frac{\partial \ell}{\partial \theta_k} = \sum_{i=1}^{N} \sum_{t=1}^{T} f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \\ - \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{y,y'} f_k(y, y', \mathbf{x}_t^{(i)}) p(y, y' \mid \mathbf{x}^{(i)}) - \frac{\partial \Omega}{\partial \theta_k}.$$

- Первое слагаемое – ожидание f_k в эмпирическом распределении $p_d(\mathbf{y}, \mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \left[\mathbf{y} = \mathbf{y}^{(i)}\right] \left[\mathbf{x} = \mathbf{x}^{(i)}\right]$.
- Второе слагаемое – ожидание f_k в распределении модели $p(\mathbf{y} \mid \mathbf{x}; \theta) p_d(\mathbf{x})$.
- Если найдём максимум, они совпадут – приятное свойство.
- Ещё приятно, что $\ell(\theta)$ вогнута, т.к. $g(\mathbf{x}) = \log \sum_i \exp x_i$ – выпуклые функции; если ещё добавить строго выпуклый регуляризатор вроде L_2, у ℓ будет единственный глобальный оптимум.

Вывод в CRF

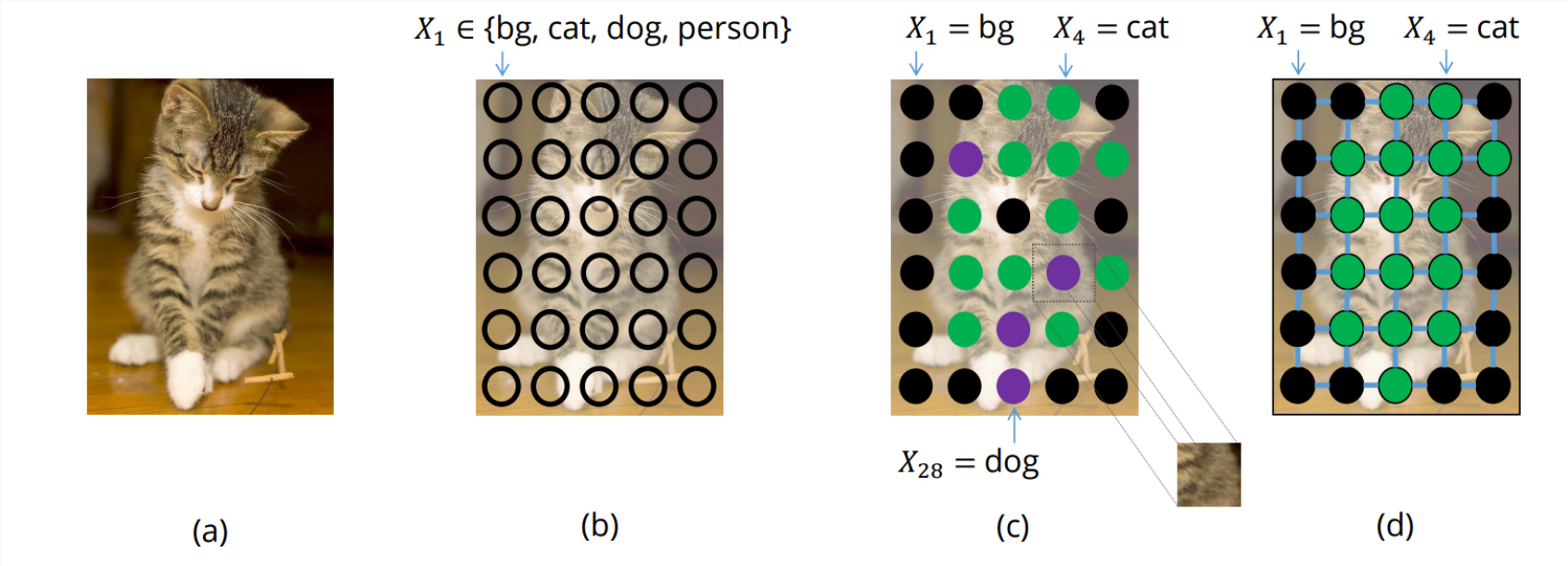
- Можно применять стандартные алгоритмы: метод Ньютона, его ускоренную версию L-BFGS...
- И для CRF общего вида то же самое на самом деле:

$$\ell(\theta) = \sum_{C_p \in C} \sum_{\Psi_c \in C_p} \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\mathbf{x}_c, \mathbf{y}_c) - \log Z(\mathbf{x}).$$

- Тоже всё вогнутое, можно использовать методы второго порядка.
- Но проблема: для каждого шага надо считать $p(y, y' \mid \mathbf{x}^{(i)})$, т.е. делать вывод в CRF для каждого тренировочного примера.
- Если так не получается, то надо делать стохастический градиентный подъём.

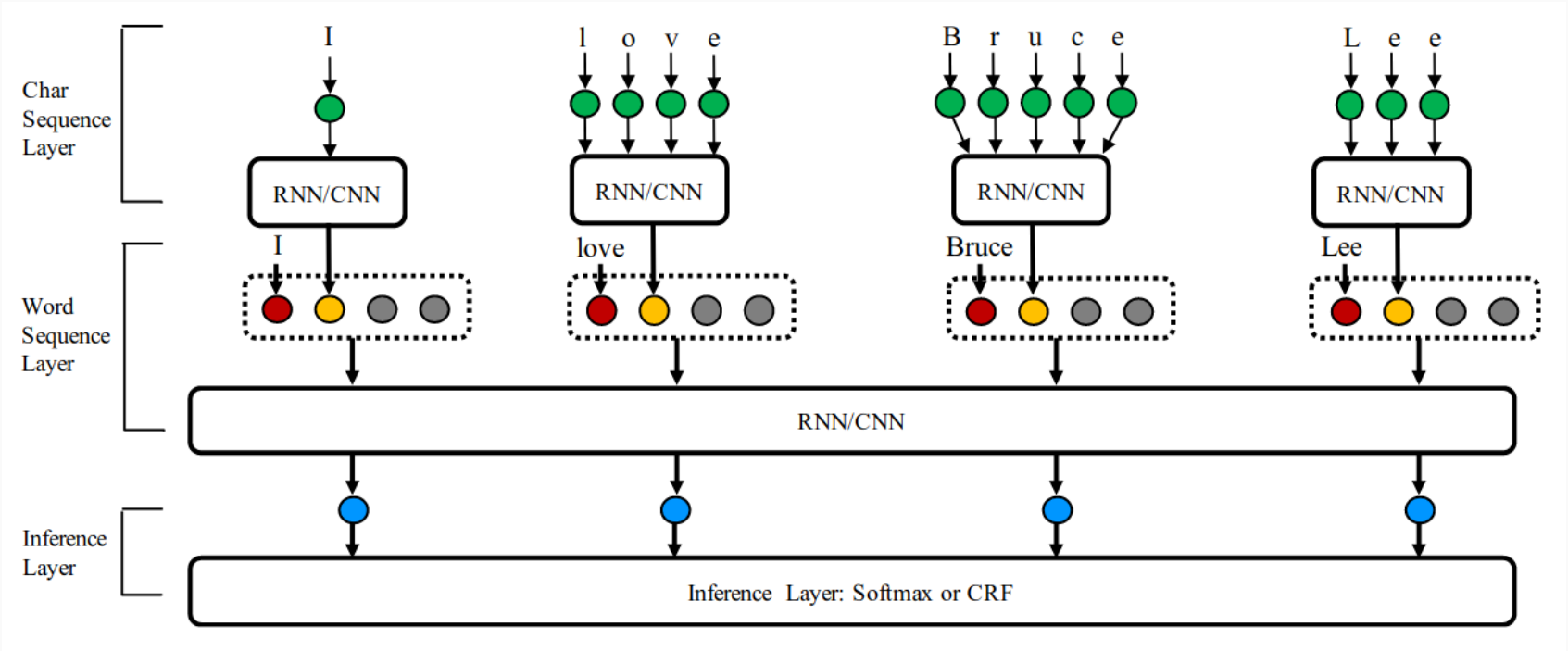
Deep learning + CRF

- И теперь можно пытаться вставлять в CRF признаки из глубоких сетей.
- Например, предскажем метку каждого пикселя классификатором (на основе всей картинки или близлежащей части), а потом вставим в CRF:



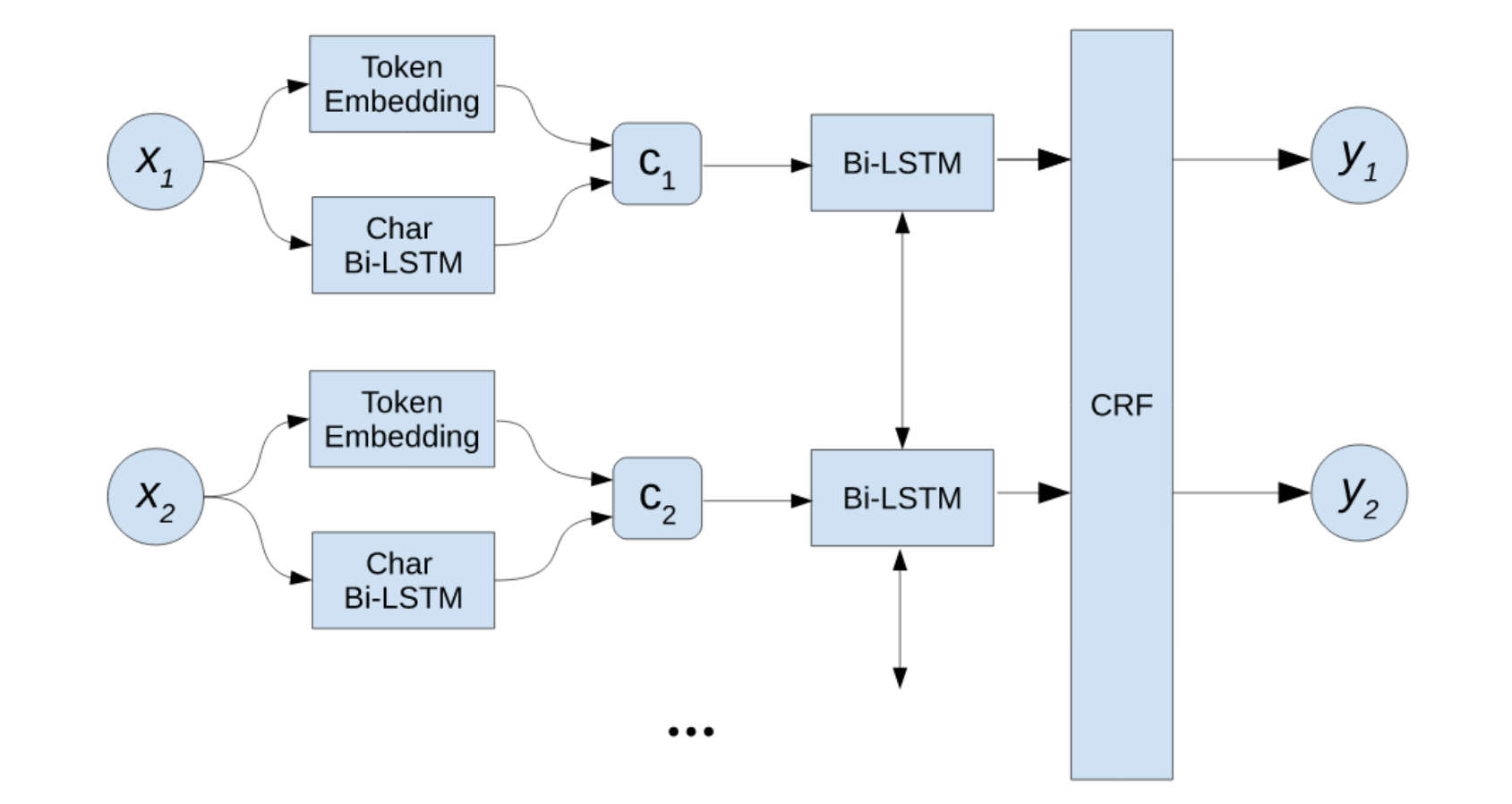
Deep learning + CRF

- Или вот NER – там RNN+CRF архитектуры до сих пор актуальны, в том числе в архитектурах вида «учитель-ученик», чтобы трансформеры не гонять:



Deep learning + CRF

- Вот, например, как NER от DeepPavlov работал (Anh et al., 2017):



Deep learning + CRF

- Structured prediction: более общий контекст, когда мы пытаемся сделать структурированные предсказания.
- CRF – один вид, бывают ещё structured SVM, например, которые вообще не очень вероятностные.
- Одна из тех моделей, которые долго оставались актуальными после прихода DL.

Спасибо за внимание!