Лекция 2

Обучение с подкреплением: ценностные методы

Дополнительные главы машинного обучения

Андрей Фильченков

04.03.2021

План лекции

- Напоминания
- Оптимальная стратегия
- Итерации по стратегиям: MC, TD, SARSA
- Q-обучение
- Глубокое Q-обучение

- В презентации используются материалы курсов
  «Машинное обучение» К.В. Воронцова,
  «Машинное обучение с подкреплением» А.И. Панова
  CS234: Reinforcement Learning, E. Brusnkill
- Слайды доступны: **shorturl.at/wGV59**
- Видео доступны: **shorturl.at/ovBTZ**

План лекции

- Напоминания
- Оптимальная стратегия
- Итерации по стратегиям: MC, TD, SARSA
- Q-обучение
- Глубокое Q-обучение

Задача о разборчивой невесте

Вы — невеста (принцесса), желающая выйти замуж (за принца). n принцев уже выстроились в очередь перед вашей комнатой.

Каждый принц последовательно заходит. Либо вы говорите «да» и тут же играете свадьбу, либо говорите «нет», и он навсегда уходит (в слезах).

Какая для вас лучшая стратегия, чтобы максимизировать матожидание качества жениха?

Ответ на задачу

Если вы принцесса, то стратегия такова:

отказать первым n/e принцам,
а потом согласиться на первого, который лучше уже просмотренных.

$$\Pr(\text{он лучший}) = 1/e \approx 0.37$$

Формализация наблюдаемой среды

Среда находится в одном из **состояний** $s \in S$. Переход между состояниями обусловлен $\tau(t) \sim p(a_t, s_t)$

Агент инициализирует $\pi(1|s) = \{s_1(a|s)\}$, и далее на каждом шаге:

- агент делает действие $a_t \sim \pi(t|s_t)$;
- среда возвращает награду $r_a \sim \$(a_t|s_t)$ и переходит в состояние $s_{t+1} \sim \tau(t)$.
- агент меняет стратегию на $\pi(t+1|s)$.

Марковский процесс принятия решений:

$$\Pr(s_{t+1} = s'; r_{t+1} = r'|s_t, a_t, r_t, \dots) =$$
$$= \Pr(s_{t+1} = s'; r_{t+1} = r'|s_t, a_t).$$

Что можно оценивать

Будущая награда: $R_t = r_{t+1} + r_{t+2} + \cdots$

Дисконтированная награда: $R_t = \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$

где γ коэффициент дисконтирования

Ценность состояния s при стратегии π:

$$V^{\pi}(s) = \mathrm{E}_{\pi}(R_t | \pi(t) = \pi) = \mathrm{E}_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| \pi(t) = \pi\right)$$

Ценность действия в состоянии s при стратегии π

$$Q^{\pi}(s, a) = \mathrm{E}_{\pi}(R_t | \pi(t) = \pi, a_t = a) =$$
$$= \mathrm{E}_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| \pi(t) = \pi, a_t = a\right)$$

Уравнение Беллмана (напоминание)

Пусть нам известно **распределение перехода**

$$\mathcal{P}_{ss'}^{a} = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$$

и **ожидание награды**

$$\mathcal{R}_{ss'}^{a} = \mathrm{E}(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s').$$

Уравнение Беллмана:

$$V^{\pi(t)}(s) = \sum_{a} s_t(a|s) \sum_{s} \mathcal{P}_{ss'}^{a} \left(\mathcal{R}_{ss'}^{a} + \gamma V^{\pi}(s')\right).$$

Но это предположение никогда не выполняется.

План лекции

- Напоминания
- Оптимальная стратегия
- Итерации по стратегиям: MC, TD, SARSA
- Q-обучение
- Глубокое Q-обучение

Еще особенности

- Стационарность среды
- Эпизодичность среды
- Конечное число действий

Оптимальная *V*-функция

Уравнение Беллмана (проще):

$$V^{\pi}(s) = \mathrm{E}_a\left(r + \gamma V^{\pi}(s')\right)$$

Оптимальная *V*-функция:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

Неясно, как по $V^*(s)$ восстановить оптимальную стратегию.

Q-функция

$$Q^{\pi}(s,a) = \mathrm{E}_{\pi}\left(\sum_{t=1}^{\infty} \gamma^k r_t \,|\, s_0 = s, a_0 = a\right)$$

$$Q^{\pi}(s,a) = r + \gamma \mathrm{E}_{s'} V^{\pi}(s')$$
$$V^{\pi}(s) = E_a Q^{\pi}(s,a)$$

Оптимальная Q-функция

Уравнение Беллмана:

$$Q^{\pi}(s,a) = r + \gamma \mathrm{E}_{a'} \mathrm{E}_{s'} \big(Q^{\pi}(s',a') \big)$$

Оптимальная Q-функция:

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

Принцип оптимальности Беллмана

Принцип оптимальности Беллмана: жадная стратегия в предположении оптимальности дальнейшего поведения оптимальна.

$$Q^*(s,a) = r + \gamma \mathrm{E}_{s'} \max_{a'} Q^*(s',a')$$

$$V^*(s) = \max_{a} \big(r + \gamma V^*(s') \big)$$

Оптимальная стратегия

Зададим частный порядок на множестве стратегий:

$$\pi \geq \pi', \text{если } V^{\pi}(s) \geq V^{\pi'}(s)\ \forall s$$

Для MDP:

- существует оптимальная стратегия
- для всех оптимальных стратегий π

$$V^{\pi}(s) = V^{*}(s)$$

$$Q^{\pi}(s,a) = Q^{*}(s,a)$$

Вид оптимальной стратегии

$$\pi(a|s) = \begin{cases} 1, \text{если } a = \arg\max\limits_a Q^*(s,a) \\ 0, \text{иначе} \end{cases}$$

Однако оптимального решения уравнения Беллмана в общем виде не существует, поэтому применяются итерационные методы.

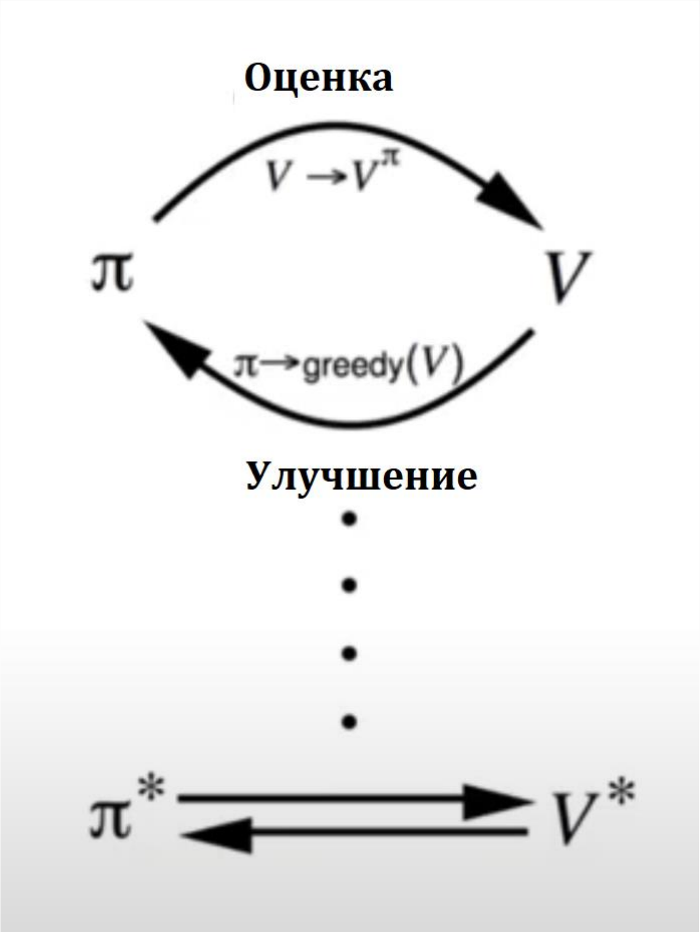
План лекции

- Напоминания
- Оптимальная стратегия
- Итерации по стратегиям: MC, TD, SARSA
- Q-обучение
- Глубокое Q-обучение

Итеративный подход

Попеременно:
- оценка значения V
- обновление π

Как оцениваем V?



Итеративный подход

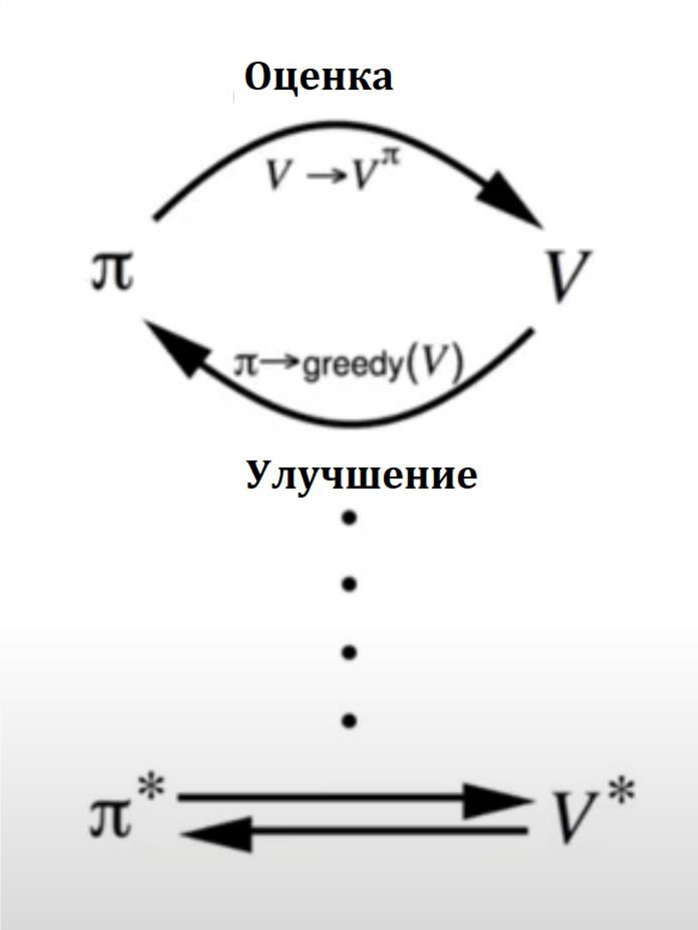
Попеременно:

- оценка значения V

$$V^*(s) = \max_a \left(r + \gamma V^*(s')\right)$$

- обновление π

Как обновляем π?



Итерация по ценностям

Попеременно:

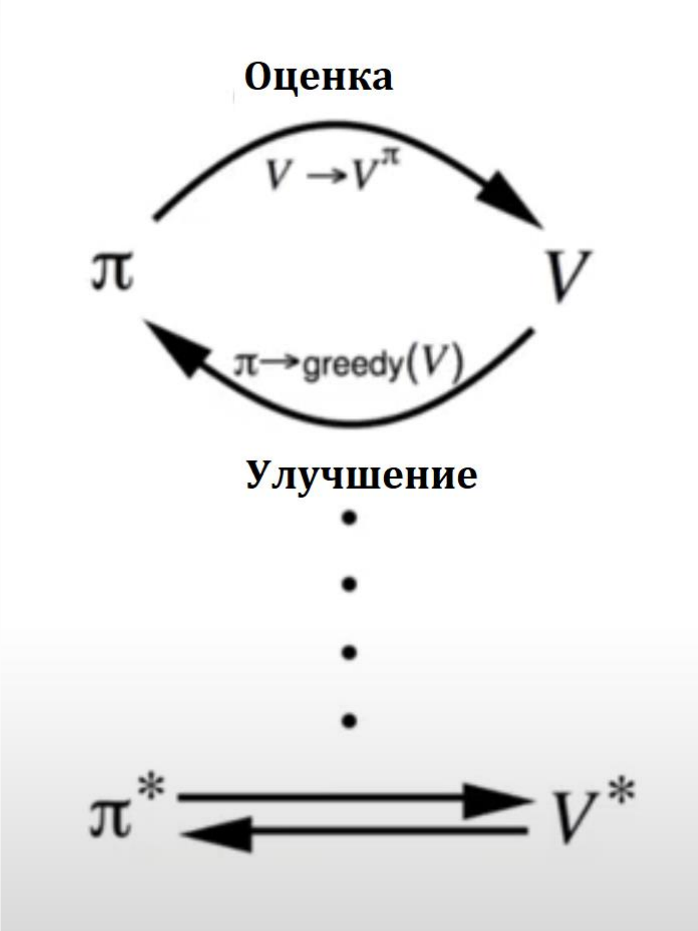
- оценка значения V

$$V^*(s) = \max_a \left(r + \gamma V^*(s')\right)$$

- обновление π

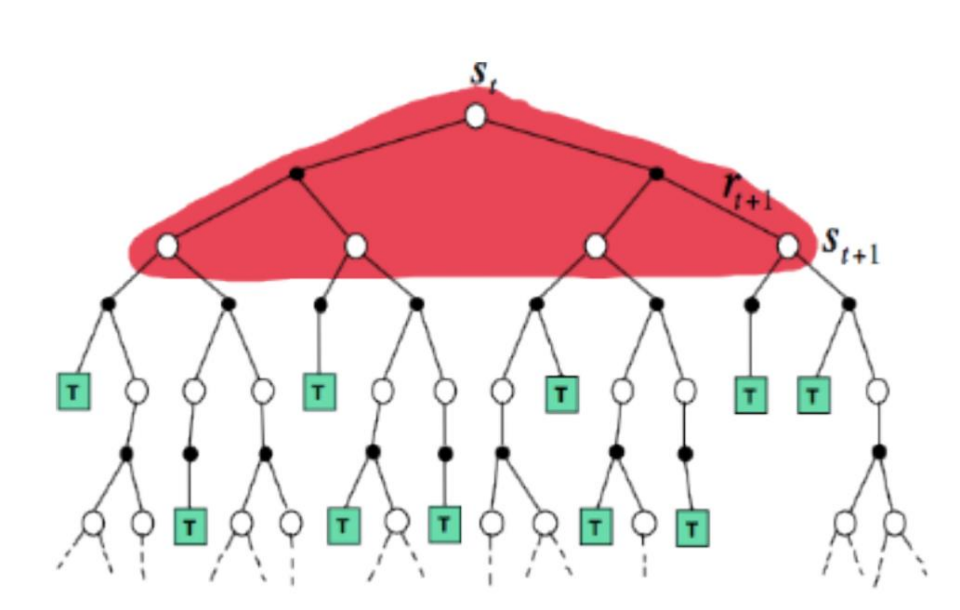
$$\pi^*(s) = \arg\max_a Q^*(s,a) =$$

$$= \arg\max_a (r + \gamma \mathrm{E}_{s'} V^*(s'))$$



Свойства

- Фактически, динамическое программирование
- Сходится к оптимальной политике
- Должны рассмотреть все действия и все ответы среды



Метод Монте-Карло

Идея: будем использовать эмпирическое среднее вместо матожидания

$$V_{(t+1)}(s_t) = \sum_{i=1}^{t} r_i[s_i = s_t]/k_t(s)$$

$$k_t(s) = \textstyle\sum_{i=1}^{t}[s_i = s_t]$$

Варианты

Можно учитывать состояния:

- только при первом посещении (first-visit)
- при каждом посещении (every-visit)

Инкрементальная запись

Перепишем:

- Для стационарного случая

$$V_{(t+1)}(s_t) = V_{(t)}(s_t) + \frac{1}{k_t(s_t)}\Big(R_{(t)}(s_t) - V_{(t)}(s_t)\Big)$$

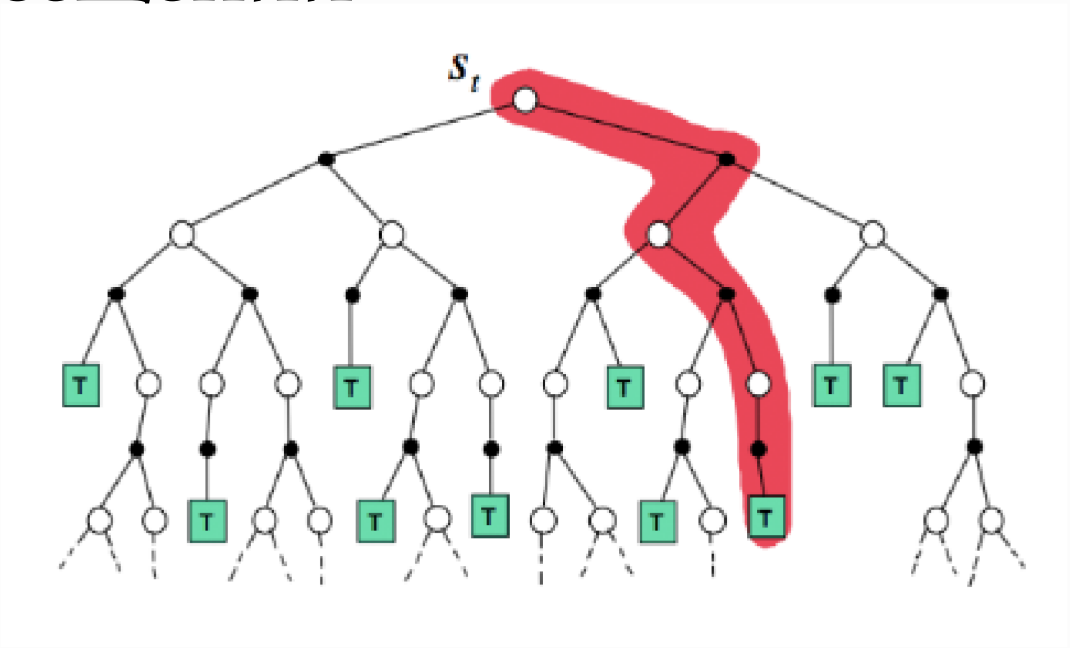
где $k_t(s_t) = \sum_{i=1}^{t}[s_i = s_t]$

- Для нестационарного случая

$$V_{(t+1)}(s_t) = V_{(t)}(s_t) + \alpha_t\Big(R_{(t)}(s_t) - V_{(t)}(s_t)\Big)$$

Свойства

- При первом посещении несмещенная оценка
- При каждом посещении смещенная оценка с более низкой дисперсией



Анализ

Достоинства

- Обычно хорошо сходится
- Не очень сильно зависит от начального приближения

Недостатки

- Нужно ждать конца эпизода
- Не может обучаться на каждом шаге

Метод временных разниц

Монте-Карло:

$$V_{(t+1)}(s_t) = V_{(t)}(s_t) + \alpha_t \left(R_{(t)}(s_t) - V_{(t)}(s_t) \right)$$

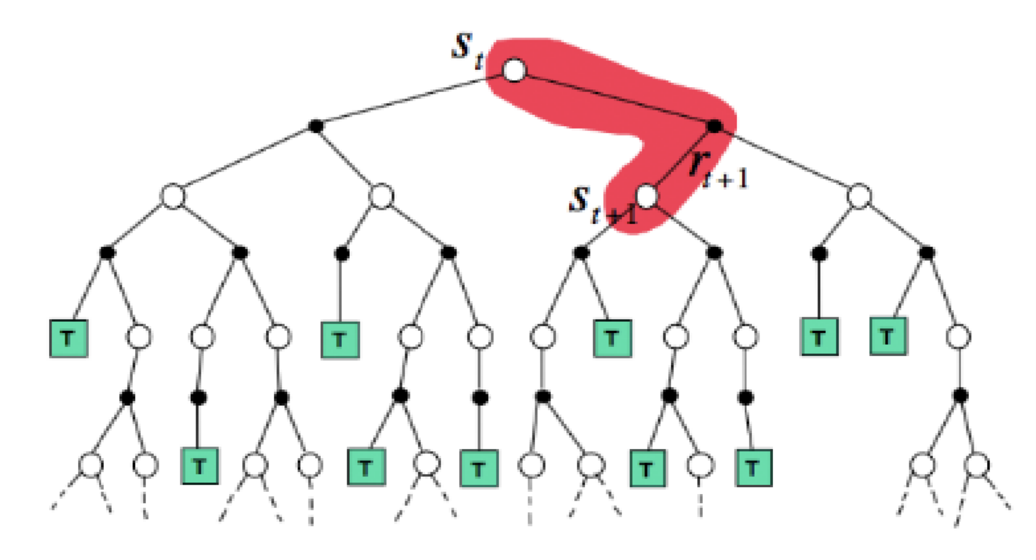
Метод временной разницы (temporal difference method, TD):

$$V_{(t+1)}(s_t) =$$
$$= V_{(t)}(s_t) + \alpha_t(r_t + \gamma V_{(t+1)}(s_{t+1}) - V_{(t)}(s_t))$$

Для $\sum_t \alpha_t = \infty, \sum_t \alpha_t^2 < \infty, V_{(t)}(s) \to V^*(s)$ при посещении всех состояний бесконечное число раз.

Свойства

- MC сходится к решению с минимальной среднеквадратичной ошибкой
- TD сходится к решению максимально правдоподобной марковской модели



Анализ

Достоинства:

- Не нужно ждать конца эпизода
- Работает для сред без термальных состояний
- Может обучаться на каждом шаге
- Низкая дисперсия
- Эффективнее MC

Недостатки:

- Дает смещенную оценку
- Более чувствителен к начальному приближению, чем MC

SARSA

SARSA (state-action-reward-state-action) :

$$Q_{(t+1)}(s_t, a_t) = Q_{(t)}(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Используем ε-жадную стратегию для выбора действия.

Еще

С

План лекции

- Напоминания
- Оптимальная стратегия
- Итерации по стратегиям: MC, TD, SARSA
- Q-обучение
- Глубокое Q-обучение

Свой и чужой опыт

Алгоритм обучения по чужому опыту (off-policy), может использовать для обучения опыт взаимодействия произвольной стратегии.

Алгоритм обучения по собственному опыту (on-policy) требуется опыт взаимодействия некоторой конкретной, предоставляемой самим алгоритмом, стратегии.

On-policy алгоритмы обучаются на своих ошибках, когда off-policy алгоритмы могут учиться как на своих, так и на чужих ошибках

Обучение по чужому опыту

Строим целевую стратегию $\pi(a|s)$ для оценки ценности по иной актуальной стратегии $\mu(a|s)$, производящей наблюдения.

Для MC и TD можно добавить правку по значимости $\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}$.

Идея Q-обучения

Будем обучать по чужому опыту $Q(s, a)$.

Следующее действие выбираем по стратегии μ

Рассматриваем последующие действия a' по стратегии π

Обновляем Q в соответствии с полезностями альтернативного действия:

$$Q_{(t+1)}(s_t, a_t) = Q_{(t)}(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a') - Q(s_t, a_t))$$

Улучшение π и μ

Будем улучшать и π, и μ

Например, обе ε-жадной стратегией.

Q-обучение

(Q-обучение) Q-learning:

Аппроксимируем оптимальную функцию ценности действия

$$Q^*(s,a) = \mathrm{E}_\pi\left(r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1},a') \,\middle|\, s_t = s, a_t = a\right).$$

$$Q_{(t+1)}(s_t,a_t) = Q_{(t)}(s_t,a_t) +$$
$$+\alpha_t(r_t + \gamma \max_{a'} Q_{(t+1)}(s_{t+1},a') - Q_{(t)}(s_t,a_t))$$

Для $\sum_t \alpha_t = \infty, \sum_t \alpha_t^2 < \infty, V_{(t)}(s) \to V^*(s)$ при посещении всех состояний бесконечное число раз.

План лекции

- Напоминания
- Оптимальная стратегия
- Итерации по стратегиям: MC, TD, SARSA
- Q-обучение
- Глубокое Q-обучение

Отход от таблиц

Во всех предыдущих случаях мы считали, что можем хранить оценки для каждой пары (s, a). Но в реальности у среды может быть крайней много состояний.

Что с этим делать?

Параметрическое Q и сети

Будем считать, что $Q(s, a, \theta)$ параметрическое, с некоторым параметром $\theta \in \Theta$, который мы будем обучать.

Теперь для восстановления $Q(s, a, \theta)$ мы можем использовать знакомые алгоритмы, например, **глубокие нейронные сети**.

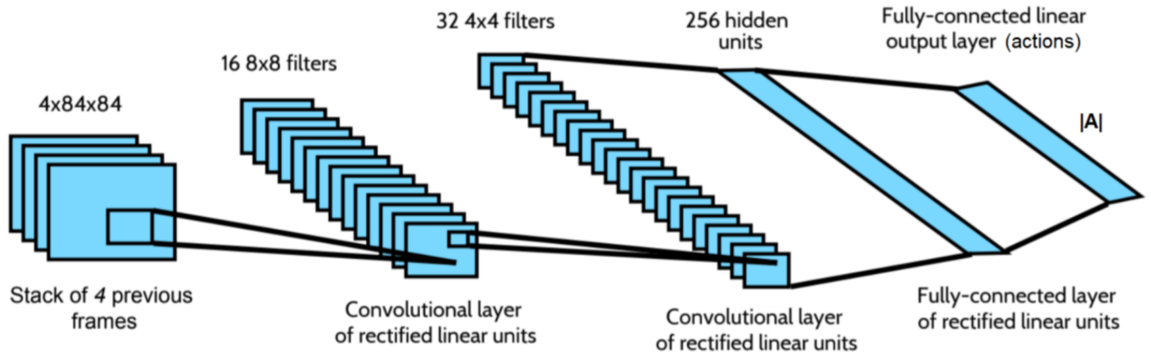
Это понизит требования к памяти для хранения, времени и объемов данных для обучения.

End-to-end работа с состояниями

Глубокие сети позволяют обрабатывать состояния в том представлении, в котором их получает агент.

Пример: игры Atari

Beam Rider | Breakout | Space Invaders | Seaquest | Pong | Enduro | Q*bert



Обучение

Цель обучения — аппроксимировать

$$y_t = \begin{cases} r_t, & \text{если } s_t \text{ терминальное} \\ r_t + \gamma \max\limits_a Q(s_{t+1}, a, \theta_t) & \text{иначе} \end{cases}$$

Функция потерь

$$\mathcal{L}_t(\theta) = (Q(s_t, a_t, \theta) - y_t)^2$$

Применяем SGD:

$$\theta_t = \theta_t - \eta(Q(s_t, a_t, \theta) - y_t)\nabla_\theta Q(s_t, a_t, \theta)$$

Две хитрости

1. Использовать experience replay
2. Затормаживать веса сети, дающей оценку для лосса обучения

Что еще бывает

- Twin DQN
- Double DQN
- Dueling DQN
- Prioritized DQN