МАРКОВСКИЕ ПРОЦЕССЫ ПРИНЯТИЯ РЕШЕНИЙ

Сергей Николенко

Академия MADE — Mail.Ru
18 сентября 2020 г.

Random facts:

- 18 сентября 1066 г. Гаральд Гардерада вместе с Тостигом Годвинсоном высадились в устье реки Хамбер; через два дня он разгромил английские войска в битве у Фулфорда, но через пять дней потерпел сокрушительное поражение при Стэмфорд-Бридж
- 18 сентября 1698 г. в Бастилию был переведён таинственный узник под номером 64489001, известный как «Железная маска»; после его смерти в 1703 г. всю мебель и одежду специально уничтожили, закрасив стены и расплавив все металлические вещи
- 18 сентября 1830 г. лошадь победила первый американский паровоз в гонке на 9 миль от Райлиз Таверн до Балтимора, а 18 сентября 1893 г. было завершено строительство Великой Северной железной дороги между Миссисипи и Тихим океаном
- 18 сентября 1870 г. исследователь Генри Уошберн открыл гейзер Old Faithful, который впоследствии стал источником одного из самых известных учебных датасетов
- 18 сентября 1934 г. СССР вошёл в состав Лиги наций; хватило лет на пять

Конкурентные и контекстуальные бандиты

Adversarial bandits

- Конкурентные бандиты (adversarial bandits) — обобщение многоруких бандитов, в котором награды выбираются не из фиксированных распределений, а выбираются противником (adversary). Т.е. процесс из трёх шагов:
  - противник выбирает распределения наград;
  - агент выбирает ручку, не зная, какие распределения выбрал противник;
  - награда выбирается из соответствующего распределения.
- Здесь есть информационная асимметрия между агентом и противником; без ограничений противник всегда сможет победить любую стратегию, конечно, он выбирает награды $r_j(t)$ произвольно.

- Но сделать что-то можно!
- Можно даже доказать оценки на regret; под regret здесь понимается разница между доходом глобально наилучшей ручки

$$G_{\max}(T) = \max_j \sum_{t=1}^{T} r_j(t)$$

  и ожиданием нашего дохода.
- А идея алгоритма будет в том, чтобы
  - сэмплировать действие с вероятностями $p_1(t), \ldots, p_K(t)$ из распределения, которое будет смесью равномерного и экспоненциальных весов от текущей оценки накопленной награды;
  - делать действие, получить награду $r_{i_t}(t)$, но ожидаемую награду считать как $\hat{r}_{i_t}(t) = r_{i_t}(t)/p_i(t)$; тогда $\mathbb{E}\left[\hat{r}_{i_t}(t) \mid i_1, \ldots, i_{t-1}\right] = r_{i_t}(t)$ (ниже разберём подробнее).

- Алгоритм Exp3 (Exponential-weight algorithm for Exploration and Exploitation):
  - для данного $\gamma \in [0, 1]$, инициализируем $w_i(1) = 1$, $i = 1, \dots, K$;
  - на каждом раунде t:
    - считаем вероятности для каждого i:

$$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K};$$

    - берём следующее действие i_t случайно с вероятностью $p_i(t)$, получаем награду $r_{i_t}(t)$;
    - обновляем вес этого действия (остальные не меняются):

$$w_{i_t}(t+1) = w_{i_t}(t)e^{\frac{\gamma}{K}\frac{r_{i_t}(t)}{p_i(t)}}.$$

- Для такого алгоритма можно доказать adversarial оценку $2.63\sqrt{G_{\max} K \log K}$ для $\gamma = \min\left(1, \frac{K \log K}{(e-1)g}\right)$, где $G_{\max}$ – верхняя оценка на максимальный доход ручки $\max_j \sum_{t=1}^{T} r_j(t)$.

Контекстуальные бандиты

- А что если жизнь устроена сложнее? Пусть мы каждую ручку дёргаем в некотором *контексте*; например, даём рекомендации пользователям, и при этом о пользователях что-то знаем.
- Т.е. на каждом шаге наблюдаем контекст $x_t \in C$, потом выбираем $I_t \in 1, \ldots, K$, получаем $r_t \sim p(r_t \mid I_t, x_t)$.
- Тогда мы уже должны не одну ручку выбрать, а обучить стратегию $\pi : C \to 1, \ldots, K$.

Контекстуальные бандиты

- Наивный подход: давайте запустим Exp3 для каждого контекста по отдельности.
- Это не так уж плохо, если контекстов мало.
- Но получается дополнительный сомножитель $\sqrt{|C|}$ в оценке на regret, а это нехорошо, вдруг их очень много.

Контекстуальные бандиты

- Другая идея – давайте вместо этого запускать Exp3 *по стратегиям* (пусть их мало); на каждом раунде t:
  - получаем совет $\xi_{k,t}$ от каждой стратегии $k \in \Pi$ в виде распределения вероятностей на ручках;
  - берём следующее действие I_t случайно с распределением $p_t = \mathbb{E}_{k \sim q_t}\left[\xi_{k,t}\right]$, получаем награду $x_{I_t,t}$;
  - вычисляем ожидаемую награду для каждого i $\tilde{x}_{i,t}$ и ожидаемую награду для каждой стратегии

$$\tilde{y}_{k,t} = \mathbb{E}_{i \sim \xi_{k,t}}\left[\tilde{x}\right]_{i,t} = \sum_{i=1}^{K} \xi_{k,t}(i) \tilde{x}_{i,t}.$$

  - обновляем распределение на стратегиях

$$q_{j,t+1} \propto e^{-\eta \sum_{s=1}^{t} \tilde{y}_{k,s}}.$$

- Это алгоритм Exp4 (Exponential-weight algorithm for Exploration and Exploitation with Experts)

- ...но откуда возьмётся $\tilde{x}_{i,t}$?
- Очень важный трюк в reinforcement learning: off-policy estimation.
- Пусть мы хотим оценить качество стратегии π, но играли мы по другой стратегии p. Что делать?
- Нам нужно оценить

$$R(x, \pi(x)) = \frac{1}{n} \sum_{s=1}^{n} \mathbb{E}_{r \sim p(r | x_s, \pi(x_s))} \left[r \right].$$

- Это было бы легко, если бы мы могли оценить для каждой ручки a

$$R(x, a) = \mathbb{E}_{r \sim p(r | a, x)} \left[r \right].$$

Контекстуальные бандиты

- Оказывается, это можно сделать по данным другой стратегии. Рассмотрим

$$\hat{R}(x_s, a) = r_s \frac{[a_s = a]}{p(a_x \mid x_s)}.$$

- Тут надо только предполагать, что $p(a_x \mid x_s)$ положительно, т.е. стратегия p покрывает стратегию π.
- Тогда

$$\mathbb{E}\left[\hat{R}(x_s, a) \mid x_s, a\right] = \mathbb{E}_{r_s}\left[\mathbb{E}_{a_s \sim p(x_s)}\left[r_s \frac{[a_s = a]}{p(a_x \mid x_s)}\right]\right] =$$
$$= \mathbb{E}_{r_s}\left[r_s \frac{p(\pi(x_s) = a_s)}{p(a_s \mid x_s)}\right] = R(x_s, a),$$

  у которого $a_s \sim p$.
- Т.е. надо просто перевзвесить вознаграждения.

- Итого:

Algorithm 2 Exp4 Algorithm (Exponential weights algorithm for Exploration and Exploitation with Experts)

Input: Set of K arms, set of experts Π.
Parameter: real number η
Let q_1 be the uniform distribution over the experts (policies), $\{1, \cdots, |\Pi|\}$.
For each round $t = 1, \cdots, T$:

- **Receive** expert advice $\xi_{k,t}$ for each expert $k \in \Pi$, where each $\xi_{k,t}$ is a probability distribution over arms.
- **Draw** an arm I_t from the probability distribution $p_t = (p_{1,t}, \cdots, p_{K,t})$, where $p_t = \mathbb{E}_{k\sim q_t}\xi_{k,t}$.
- **Observe** loss $\ell_{I_t,t}$. For each arm i, compute $\tilde{\ell}_{i,t}$, using the Inverse Propensity Score trick in Theorem 1 to obtain an unbiased estimator for the loss of arm i:

$$\tilde{\ell}_{i,t} = \frac{\ell_{i,t}}{p_{i,t}} \mathbb{1}_{I_t=i} \qquad i = 1, \cdots, K$$

- **Compute** the estimated loss for each expert, by taking the expected loss over the expert's predictions.

$$\tilde{y}_{k,t} = \mathbb{E}_{i\sim\xi_{k,t}} \tilde{\ell}_{i,t} = \sum_{i=1}^{K} \xi_{k,t}(i)\tilde{\ell}_{i,t} \qquad k = 1, \cdots, |\Pi|$$

- **Compute** the new probability distribution over the experts $q_{t+1} = (q_{1,t+1}, \cdots, q_{N,t+1})$, where

$$q_{j,t+1} = \frac{\exp(-\eta \sum_{s=1}^{t} \tilde{y}_{k,s})}{\sum_{k=1}^{|\Pi|} \exp(-\eta \sum_{s=1}^{t} \tilde{y}_{k,s})}$$

Контекстуальные бандиты

- А ещё, конечно, в контекстуальном бандите можно сделать модель какую-нибудь. Например, LinUCB предполагает, что ожидаемая награда – это линейная функция:

$$r_{a,t} = \mathbf{x}_{a,t}^{\top}\theta_a^* + \epsilon_t,$$

  где θ_a^* — вектор коэффициентов, который нужно оценивать, а $\mathbf{x}_{a,t}$ — вектор признаков контекста.
- И тогда по LinUCB надо выбирать $I_t = \arg\max_a u_{a,t}$, где

$$u_{a,t} = \max_{\theta_a \in C_{a,t-1}} \mathbf{x}_{a,t}^{\top}\theta_a,$$

  а $C_{a,t-1}$ — это доверительное множество, аналог доверительного интервала, которое мы в обычном UCB оценивали.

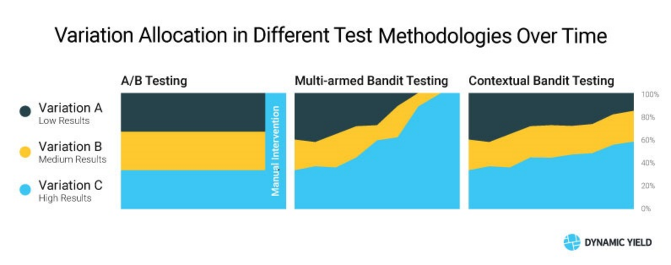
- Итого:

Algorithm 3 LinUCB with Contextual Bandits

Input: $R \in \mathbb{R}^+$, regularization parameter λ
for $t = 1, 2, \dots, T$ **do**
  Observe feature vectors of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{a,t} \in \mathbb{R}^d$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \lambda \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\theta}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
    $C_{a,t} \leftarrow \left\{ \theta_a^* \in \mathbb{R}^d : \left|\left|\hat{\theta}_{a,t} - \theta_a^*\right|\right|_{\mathbf{A}_a} \leq R\sqrt{2\log\left(\frac{\det(\mathbf{A}_a)^{1/2}\det(\lambda I)^{-1/2}}{\delta}\right)} + \lambda^{1/2} S \right\}$
    $p_{a,t} \leftarrow \arg\max_{\hat{\theta}_a \in C_{a,t}} \mathbf{x}_{a,t}^T \hat{\theta}_a$
  end for
  Choose arm $a_t = \arg\max_{a \in \mathcal{A}_t} p_{a,t}$ with ties broken arbitrarily, and observe payoff r_t
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{a_t,t}\mathbf{x}_{a_t,t}^T$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t \mathbf{x}_{a_t,t}$
end for

Зачем нужны многорукие бандиты

- Многорукие бандиты используются, например, для A/B тестирования.



- Можно и для оптимизации гиперпараметров в тех же нейронных сетях (или где угодно).
- Контекстуальные – для рекомендаций, для выбора из нескольких вариантов и т.д.
- Но для нас сейчас это первый шаг, упрощённая постановка...

Агенты с несколькими состояниями

- Вернёмся теперь к задаче с несколькими состояниями.
- Вознаграждения (rewards) – на каждом шаге: r_t, r_{t+1}, ...
- Но что такое «хорошо» in the long run? Как оценивать поведение алгоритма в приведённом выше сеттинге?
- Если есть естественное конечное число шагов (партия), то это *эпизодическая задача* (episodic task), и логично суммировать вознаграждение по эпизоду (до терминального состояния).
- А что с продолжающимися задачами?

Разные модели

- Ваши версии?

Разные модели

- Модель *конечного горизонта*: агент обращает внимание только на следующие h шагов: $E\left[\sum_{t=0}^{h} r_t\right]$.

Разные модели

- Модель *конечного горизонта*: агент обращает внимание только на следующие h шагов: $E\left[\sum_{t=0}^{h} r_t\right]$.
- Модель *бесконечного горизонта*: хотелось бы учесть все возможные шаги в будущем, т.к. время жизни агента может быть не определено. Но при этом чем раньше получим прибыль, тем лучше. Как это учесть?

Разные модели

- Модель *конечного горизонта*: агент обращает внимание только на следующие h шагов: $E\left[\sum_{t=0}^{h} r_t\right]$.
- Модель *бесконечного горизонта*: хотелось бы учесть все возможные шаги в будущем, т.к. время жизни агента может быть не определено. Но при этом чем раньше получим прибыль, тем лучше. Как это учесть?
- $$E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right],$$

  где γ — некоторая константа (discount factor).

Разные модели

- Модель *конечного горизонта*: агент обращает внимание только на следующие h шагов: $E\left[\sum_{t=0}^{h} r_t\right]$.
- Модель *бесконечного горизонта*: хотелось бы учесть все возможные шаги в будущем, т.к. время жизни агента может быть не определено. Но при этом чем раньше получим прибыль, тем лучше. Как это учесть?
- $$E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right],$$

  где γ — некоторая константа (discount factor).
- Модель *среднего вознаграждения* (average-reward model):

$$\lim_{h\to\infty} E\left[\frac{1}{h}\sum_{t=0}^{h} r_t\right].$$

Что мы будем использовать

- Все модели разные, приводят к разным результатам.
- Обычно используется модель бесконечного горизонта с некоторым фиксированным discount factor. Её и мы будем использовать.
- Кроме того, её можно обобщить на эпизодические задачи: достаточно просто положить $\gamma = 1$ и добавить одно лишнее состояние с вознаграждением 0, которое будет замкнуто на себя. Так что отныне навсегда

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

Как оценивать качество обучения?

- Предыдущие модели могут оценивать готовый обучившийся алгоритм. Как оценивать качество обучения?

Как оценивать качество обучения?

- Предыдущие модели могут оценивать готовый обучившийся алгоритм. Как оценивать качество обучения?
- Рано или поздно сходится к оптимальному. Это часто можно доказать, но может быть слишком медленно и/или со слишком большими потерями по дороге.

Как оценивать качество обучения?

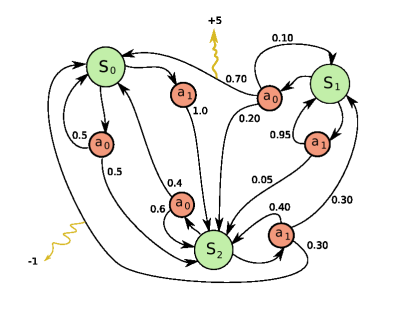
- Предыдущие модели могут оценивать готовый обучившийся алгоритм. Как оценивать качество обучения?
- Рано или поздно сходится к оптимальному. Это часто можно доказать, но может быть слишком медленно и/или со слишком большими потерями по дороге.
- Сходится с большой скоростью. Два подхода:
    - Скорость сходимости к какой-то фиксированной доле оптимальности. Какой?
    - Насколько хорошо себя ведёт алгоритм после фиксированного времени. Какого?

Как оценивать качество обучения?

- Предыдущие модели могут оценивать готовый обучившийся алгоритм. Как оценивать качество обучения?
- Рано или поздно сходится к оптимальному. Это часто можно доказать, но может быть слишком медленно и/или со слишком большими потерями по дороге.
- Сходится с большой скоростью. Два подхода:
  - Скорость сходимости к какой-то фиксированной доле оптимальности. Какой?
  - Насколько хорошо себя ведёт алгоритм после фиксированного времени. Какого?
- Минимизировать цену (regret), т.е. уменьшение общей суммы выигрыша по сравнению с оптимальной стратегией с самого начала. Это очень хорошая мера, но результаты о ней получить очень сложно.

Марковские процессы

- *Марковский процесс принятия решений* (Markov decision process) состоит из:
  - множества состояний S; множества действий A;
  - функции поощрения $R : S \times A \to \mathbb{R}$; ожидаемое вознаграждение при переходе из s в s' после a – $R^a_{ss'}$;
  - функции перехода между состояниями $p^a_{ss'} : S \times A \to \Pi(S)$, где $\Pi(S)$ — множество распределений вероятностей над S. Вероятность попасть из s в s' после a равна $P^a_{ss'}$.
- Модель *марковская*: переходы не зависят от истории.



- Главный момент – разница между *reward function* (непосредственное подкрепление) и *value function* (общее подкрепление, ожидаемое, если начать с этого состояния).

- Суть многих методов обучения с подкреплением – в том, чтобы оценивать и оптимизировать value function.

Подкрепление и значение состояния

- Для марковских процессов можно формально определить:

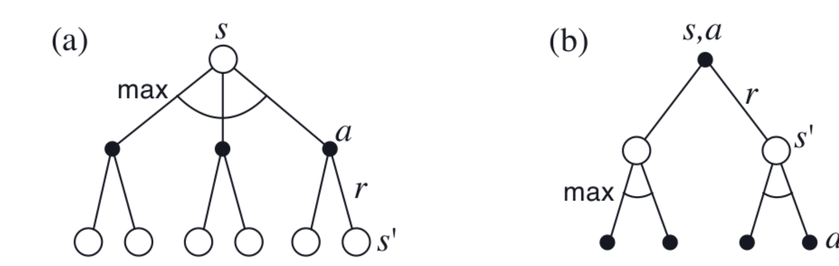
$$V^\pi(s) = \mathrm{E}_\pi\left[R_t \mid s_t = s\right] = \mathrm{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s\right].$$

- Можно более детализированно – общее подкрепление, ожидаемое, если начать с состояния s и действия a:

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

Подкрепление и значение состояния

- Функции V и Q – это как раз то, что нам нужно оценить; если бы мы их знали, можно было бы просто выбирать то a, которое максимизирует $Q(s, a)$.



- Для известной стратегии π V^π удовлетворяют уравнениям Беллмана:

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

$$= \mathrm{E}_\pi\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s\right] =$$

$$= \sum_a \pi(s,a)\sum_{s'} P_{ss'}^a\left(R_{ss'}^a + \gamma\mathrm{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_{t+1} = s'\right]\right) =$$

$$= \sum_a \pi(s,a)\sum_{s'} P_{ss'}^a\left(R_{ss'}^a + \gamma V^\pi(s')\right).$$

Основные задачи

- Теоретически всё готово, но у нас много проблем:
    - уравнения знаем, но пока не знаем, как их решать, то есть как найти V^π для данного π?
    - разных стратегий очень, очень много — как найти оптимальную стратегию поведения агента в данной модели и соответствующие V^*?
    - но уравнений тоже не знаем — в реальности обычно P и R не даны, их тоже нужно обучить; как?
    - более того, их обычно даже записать не получится, слишком уж много состояний в любой реальной задаче... что делать?

- Давайте есть слона по частям...

Оптимальные значения состояний

- *Оптимальное значение состояния* — ожидаемая суммарная прибыль, которую получит агент, если начнёт с этого состояния и будет следовать оптимальной стратегии:

$$V^*(s) = \max_\pi E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right].$$

- Эту функцию можно определить как решение уравнений

$$V^*(s) = \max_a \sum_{s' \in S} P_{ss'}^a \left(R_{ss'}^a + \gamma V^*(s')\right),$$

а затем выбрать оптимальную стратегию

$$\pi^*(s) = \arg\max_a \sum_{s' \in S} P_{ss'}^a \left(R_{ss'}^a + \gamma V^*(s')\right).$$

- Как решать уравнения?

Policy evaluation

- Чтобы посчитать value functions для данной стратегии π, можно просто итеративно пересчитывать по уравнениям Беллмана:

$$V^{\pi}(s) := \sum_{a} \pi(s,a) \sum_{s' \in S} P^{a}_{ss'} \left(R^{a}_{ss'} + \gamma V^{\pi}(s') \right),$$

  пока не сойдётся.

- Соответственно, для оптимального надо решать уравнения с максимумами:

$$V^{*}(s) := \max_{a} \sum_{s' \in S} P^{a}_{ss'} \left(R^{a}_{ss'} + \gamma V^{*}(s') \right).$$

Итеративное решение (по значениям)

- Можно то же самое по Q: пока не сойдётся,

$$Q(s,a) := \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma \sum_{a'} \pi(s,a') Q(s,a') \right).$$

- А потом просто $V(s) := \max_a Q(s,a)$.
- Оптимальное $Q^*(s,a)$ тоже нехитро:

$$Q^*(s,a) := \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma \max_{a'} Q^*(s,a') \right).$$

Другой вариант

- Пересчёт в предыдущем алгоритме использует информацию от всех состояний–предшественников. Можно сделать другой вариант:

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)).$$

- Он работает, если каждая пара (s,a) встречается бесконечное число раз, s' выбирают из распределения $P^a_{ss'}$, а r сэмплируют со средним $R(s,a)$ и ограниченной дисперсией.
- Но ведь на самом деле нам не V и не Q нужно, а оптимальная стратегия...

Улучшение стратегий

- Мы ищем V^π, чтобы улучшить π. Как улучшить π?
- Естественная идея: давайте жадно выбирать a в s как $\arg\max_a Q^\pi(s, a)$ после вычисления Q^π.
- Policy improvement theorem: для π и π', если для всех s

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s),$$

  то π' не хуже π, т.е. $\forall s\ V^{\pi'}(s) \geq V^\pi(s)$.
- Как доказать?

Улучшение стратегий

- Просто будем разворачивать V^π:

$$\begin{aligned}
V^\pi \leq & Q^\pi(s, \pi'(s)) = \mathrm{E}_{\pi'}\left[r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_{t+1} = s\right] \\
\leq & \mathrm{E}_{\pi'}\left[r_{t+1} + \gamma Q^\pi(s_{t+1}), \pi'(s_{t+1})) \mid s_{t+1} = s\right] \\
\leq & \ldots \leq \\
\leq & \mathrm{E}_{\pi'}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s_{t+1} = s\right] = V^{\pi'}(s).
\end{aligned}$$

Итеративное решение (по стратегиям)

- Ищем оптимальную стратегию итеративным алгоритмом.
- `PolicyIteration` – инициализировать π, потом, пока $\pi \neq \pi'$, повторять:
  - вычислить значения состояний для стратегии π, решив систему линейных уравнений

$$V^{\pi}(s) := \sum_{a} \pi(s, a) \sum_{s' \in S} P^{a}_{ss'} \left(R^{a}_{ss'} + \gamma V^{\pi}(s') \right);$$

  - улучшить стратегию на каждом состоянии:

$$\pi'(s) := \arg\max_{a} Q^{\pi}(s, a) = \arg\max_{a} P^{a}_{ss'} \left(R^{a}_{ss'} + \gamma V^{\pi}(s') \right);$$

- Почему оно сходится?

Итеративное решение (по стратегиям)

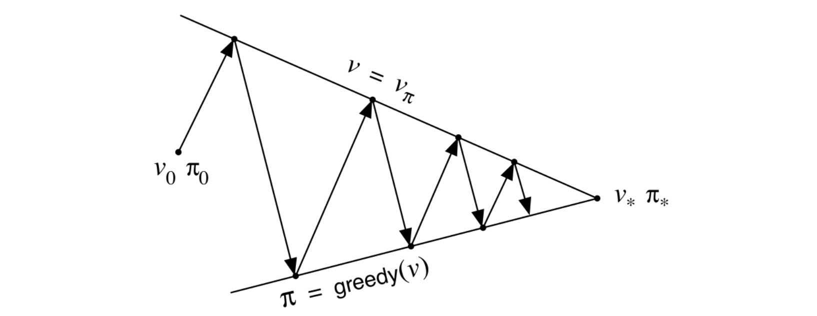
- Сходится, т.к. на каждом шаге строго улучшаем целевую функцию, а всего существует конечное число ($|A|^{|S|}$) стратегий.
- Но, конечно, это медленно, надо V^π пересчитывать; проще делать на каждой итерации ровно один шаг пересчёта V^π, а потом сразу выбирать жадную стратегию:

$$V_{k+1}(s) := \max_a \sum_{s' \in S} P^a_{ss'} \left(R^a_{ss'} + \gamma V_k(s') \right).$$

- Это называется value iteration.

Итеративное решение (по стратегиям)

- Есть другие похожие методы – их всех объединяет подход, основанный по сути на чём-то вроде EM-алгоритма с динамическим программированием.



- Это может быть достаточно эффективно даже для больших задач (с трюками, позволяющими не всё пространство исследовать).

Спасибо!

Спасибо за внимание!