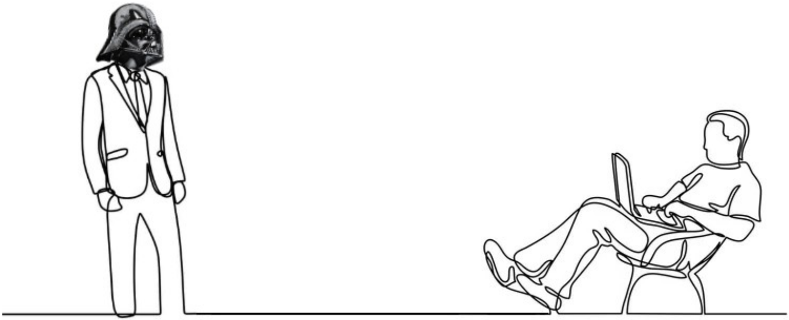
Введение в обучение с подкреплением

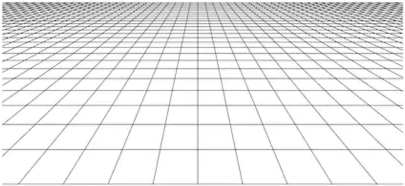
Першин Максим (181)

Факультет компьютерных наук
Высшая школа экономики, 2021

Ситуация



Задача



$$state = (x, y, \theta, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \beta_2, \beta_3, \beta_4)$$

Обучение с учителем

Предпосылки:

- $a(x) \in A$ – алгоритм из семейства алгоритмов A.
- X – множество объектов.
- y – множество ответов.
- $L(a(x), y)$ – функция потерь.

Задача:

$$a^*(x) = \arg\min_{a \in A} \sum_{i=1}^{\ell} L(a(x_i), y_i)$$

Обучение без учителя

Предпосылки:

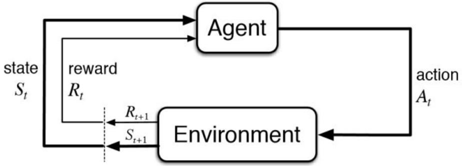
- $a(x) \in A$ – алгоритм из семейства алгоритмов A.
- X – множество объектов.
- Некоторое априорное знание о структуре выборки.

Задачи:

- Кластеризация
- Сокращение размерности
- Выделение параметров распределения
- ...

Reinforcement learning

S – множество состояний агента.
A – множество действий, которые может совершить агент.



Цель агента – подобрать политику $\pi : S \to A$, которая максимизирует суммарный выигрыш R:

$$R = \sum_{t=1} R_t \cdot \gamma^t, \quad R_t \in \mathbb{R}, \quad \gamma \in [0, 1]$$

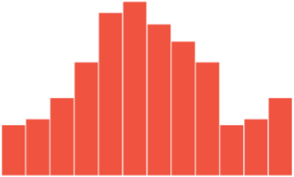
Дисконтирующий множитель

Дисконтирующий множитель – величина из отрезка $[0, 1]$.

- Определяет, насколько сильно уменьшается вклад от каждого следующего выигрыша в формуле суммарного выгрыша.
- Используется только в случае, когда у агента нет терминального состояния. В случае наличия терминального состояния γ выствляется равным 1.

Policy

Политика – функция из множества состояний агента во множество вероятностных распределений действий агента.

State $\longrightarrow$ 

Применения RL

Медицина	https://arxiv.org/abs/1908.08796
NLP	https://arxiv.org/abs/1705.04304
	https://arxiv.org/abs/1606.01541
Finances	https://arxiv.org/abs/1803.03916
Беспилотники	https://arxiv.org/abs/2002.00444
Deeplearning	https://arxiv.org/abs/1606.01885

Многорукий бандит

Жадная стратегия

Условия:

- Пусть у бандита есть k ручек.
- $N = \{0, ..., 0\}$ – сколько раз была выбрана i-ая ручка.
- $E = \{0, ..., 0\}$ – оценка матожидания выигрыша i-ой ручки.

На каждой итерации:

- Выбираем ручку с наибольшей оценкой матожидания выигрыша и дергаем ее.
- Обновляем величины:

$$N_i = N_i + 1 \qquad E_i = E_i - \frac{E_i}{N_i} + \frac{R}{N_i}$$

ε-жадная стратегия

Алгоритм:

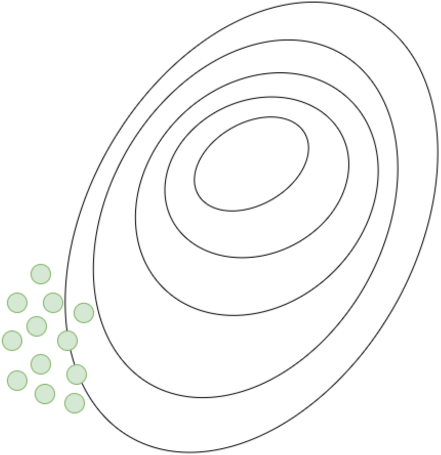
- Пусть ε – константа из интервала $(0, 1)$.
- Инициализируем счетчики как в жадной стратегии.
- С вероятностью ε выбираем случайную ручку автомата, с вероятностью $1 - \varepsilon$ выбираем ручку с максимальной оценкой матожидания выигрыша.
- Дергаем за ручку и обновляем счетчики как в жадной стратегии.

Кросс-энтропийный метод

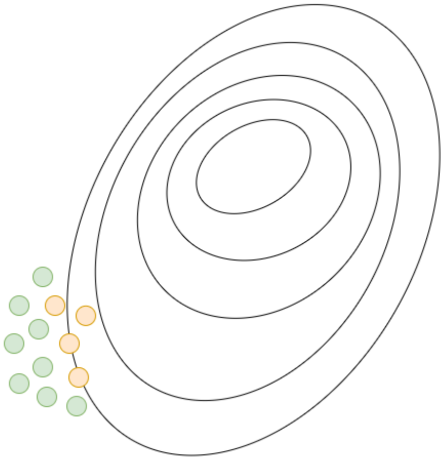
Алгоритм:

- Политика – табличка ($|S|$ строк, $|A|$ столбцов), инициализированная случайными вероятностями.
- Повторяем в цикле:
    - Играем N игр и выбираем из них подмножество с лучшим выигрышем R – элитные сессии.
    - Меняем политику, подстраивая ее под элитные сессии.

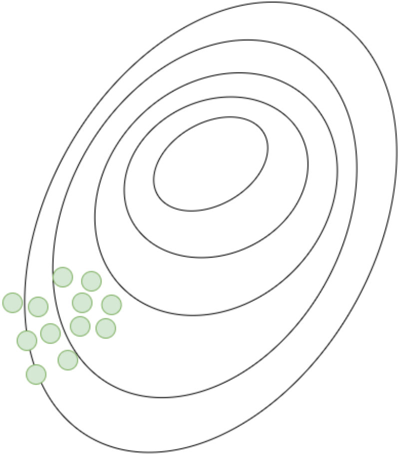
Кросс-энтропийный метод



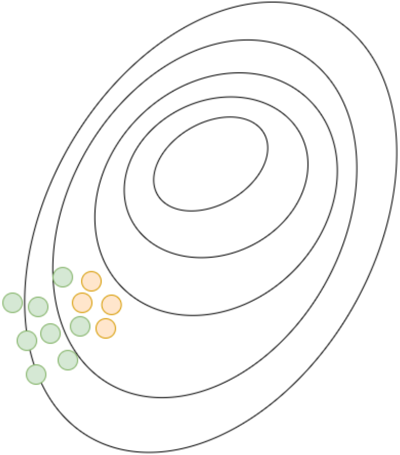
Кросс-энтропийный метод



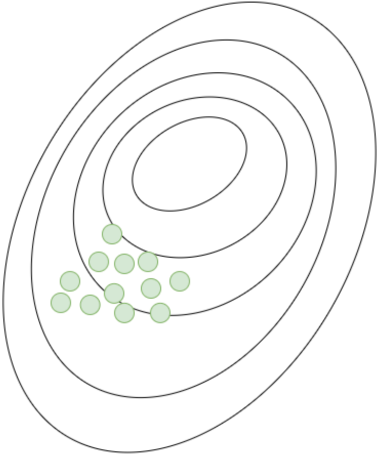
Кросс-энтропийный метод



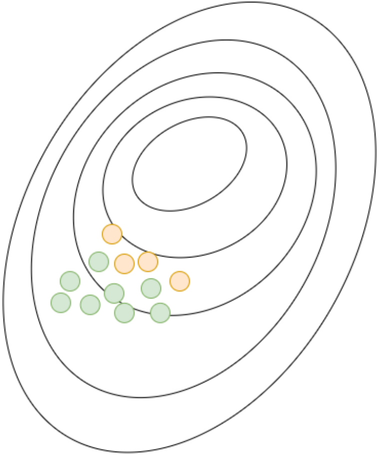
Кросс-энтропийный метод



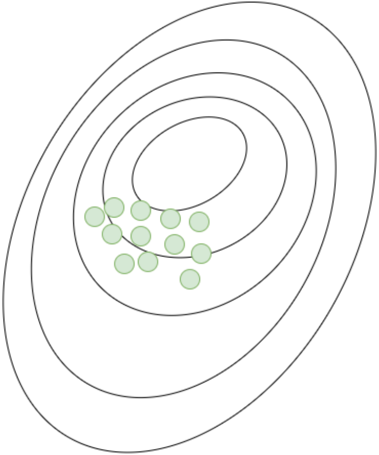
Кросс-энтропийный метод



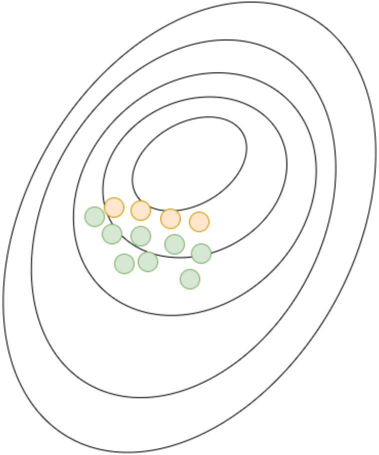
Кросс-энтропийный метод



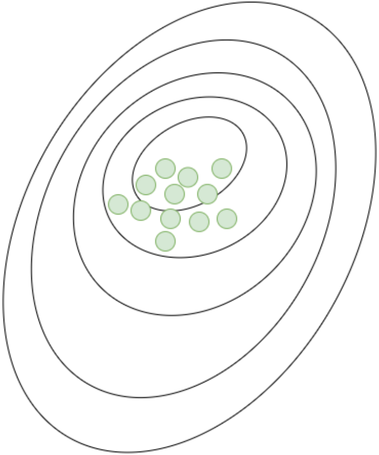
Кросс-энтропийный метод



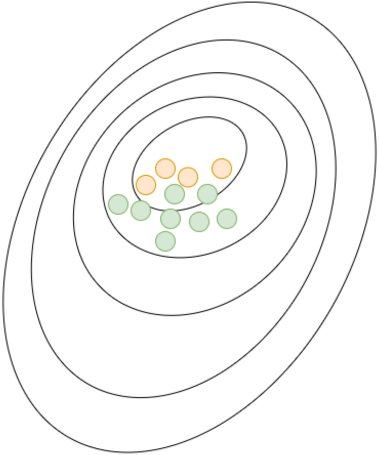
Кросс-энтропийный метод



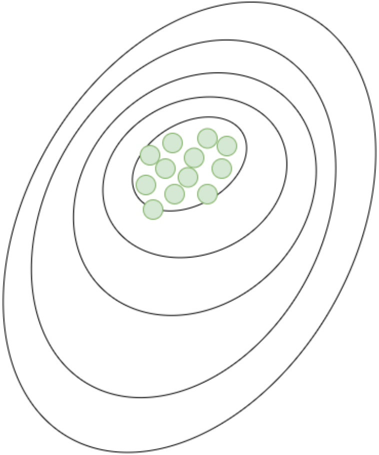
Кросс-энтропийный метод



Кросс-энтропийный метод



Кросс-энтропийный метод



Обновление политики

В самом простом случае:

- Старая политика отбрасывается.
- Список элитных состояний и соответствующих им действий: $Elites = \{(s_1, a_1), (s_2, a_2), \dots\}$
- Вероятности действий в непосещенных состояниях заполняются равномерно.
- В посещенных хотя бы раз состояниях:

$$\pi(a|s) = \frac{\sum_{s,a \in Elites} [s = s'][a = a']}{\sum_{s,a \in Elites} [s = s']} =$$

$$= \frac{\text{Сколько раз действие } a \text{ сделано в состоянии } s}{\text{Сколько всего раз посещали } s}$$

Проблемы метода

Проблемы:

- Политика переобучается или теряет прогресс в обучении в редких состояниях.
- Элитные сессии – обычно лучшие 10-20% сессий, остальные данные отбрасываются.

Какие-никакие решения:

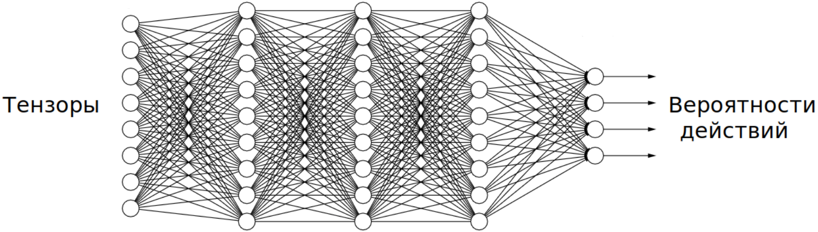
- Сглаживание политик:

$$policy_i = \alpha \cdot policy_i + (1 - \alpha) \cdot policy_{i-1}$$

- Использовать элитные сессии 2-3 предыдущих эпох.

Нейронки

Пусть теперь количество состояний континуально и описывается тензором. Можно адаптировать кросс-энтропийный метод на такой случай:



$$\pi(a|s)$$

Нейронки

Алгоритм обучения в таком случае совсем не изменится:

- nn = NNClassifier()
- Цикл:
  - Играем N игр
  - Elites $= \{(s_1, a_1), (s_2, a_2), \dots\}$
  - nn.fit(Elites)

Источники

- https://github.com/yandexdataschool/Practical_RL
- https://www.coursera.org/learn/practical-rl
- https://en.wikipedia.org/wiki/Reinforcement_learning