

Введение в обучение с подкреплением



Введение

Описание задачи:

В обучении с подкреплением существует агент взаимодействующий с окружающей средой и предпринимающий некоторые действия (ходы).

В ответ окружающая среда дает награду за тот или иной ход и агент продолжает их выполнять.





Введение

Описание задачи:

При обучении с подкреплением, в отличие от обучения с учителем, не предоставляются верные пары "входные данные – ответ", а принятие субоптимальных решений не ограничивается явно.

Основываясь на взаимодействии с окружающей средой, агент должен выработать стратегию, которая максимизирует сумму вознаграждений.



Введение

Среда обычно формулируется как Марковский процесс принятия решений (МППР) с конечным множеством состояний.

Вероятности выигрышей и перехода состояний в МППР обычно являются величинами случайными, но стационарными в рамках задачи.

Введение

Кратко о Марковских процессах:

Марковский процесс принятия решений задается кортежем из 4-х значений:

S – конечное множество состояний

A – конечное множество действий (часто представляется в виде множества A_s , доступных из состояния s)

$P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ – вероятность, что действие a в состоянии s во время t приведет в состояние s' ко времени $t+1$

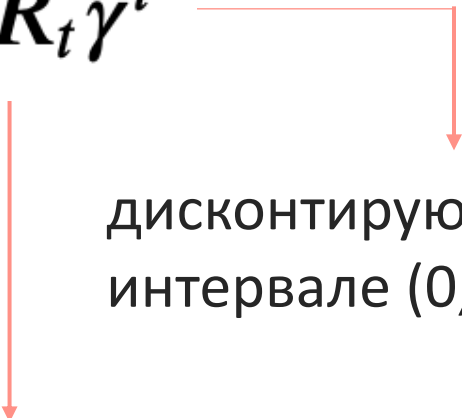
$R_a(s, s')$ – вознаграждение, получаемое после перехода в состояние s' из s с вероятностью перехода $P_a(s, s')$

Введение

Постановка задачи:

Цель агента – выработать стратегию $\pi : S \rightarrow A$, которая максимизирует величину R , где

$$R = \sum_t R_t \gamma^t$$



дисконтирующий множитель, лежит в интервале $(0, 1)$

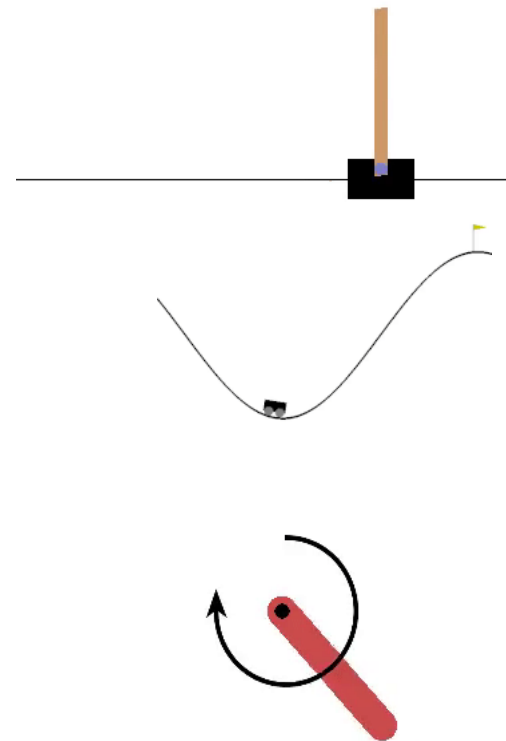
вознаграждение, получаемое после перехода в состояние s_{t+1} из состояния s_t в момент времени t

Проблема определения выигрыша

Одной из основных проблем при решении проблем реального мира в RL является проблема выбора выигрыша

Примеры:

- Сохранение наклона
- Увеличение механической энергии
- Ограничение сверху



Алгоритмы и типы задач

Эпизодические или непрерывные задачи

Эпизодические задачи

В этом случае у нас есть начальная и конечная точка. Это создает эпизод: список состояний, действий, наград и будущих состояний.



Алгоритмы и типы задач

Эпизодические или непрерывные задачи

Непрерывные задачи

Это задачи, которые продолжаются вечно. В этом случае система должна научиться выбирать оптимальные действия и одновременно взаимодействовать со средой.

Как пример можно привести систему, которая автоматически торгует акциями. Для этой задачи нет начальной точки и состояния терминала.

Алгоритмы и типы задач

Монте-Карло против Временной разницы

Метод Монте-Карло

Когда эпизод заканчивается, система смотрит на накопленное вознаграждение, чтобы понять насколько хорошо она выполнила свою задачу. В методе Монте-Карло награды получают только в конце игры.

Затем, мы начинаем новую игру с новыми знаниями. С каждым разом система проходит этот уровень все лучше и лучше.

$$\underline{V(S_t)} \leftarrow \underline{V(S_t)} + \alpha [G_t - \underline{V(S_t)}]$$

Maximum
expected future
reward starting at
that state

Former estimation of
maximum expected
future reward starting at
that state

learning
rate

Discounted
cumulative
rewards

Алгоритмы и типы задач



- Каждый раз мы будем начинать с одного и того же места.
- Мы проиграем, если кошка съедает нас или если мы сделаем более 20 шагов.
- В конце эпизода у нас есть список состояний, действий, наград и новых состояний.
- Система будет суммировать общее вознаграждение G_t .
- Затем она обновит $V(st)$, основываясь на приведенной выше формулы.
- Затем начнет игру заново, но уже с новыми знаниями.

Алгоритмы и типы задач

Монте-Карло против Временной разницы

Метод Монте-Карло

Данный метод не ждёт конца эпизода, чтобы обновить максимально возможное вознаграждение.

Он будет ждать лишь следующего временного шага, чтобы обновить значения.

В момент времени $t+1$ обновляются все значения, а именно вознаграждение меняется на R_{t+1} , а текущая оценка $V(S_t)$ на $V(S_{t+1})$.

TD Learning

$$V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Diagram illustrating the components of the TD Learning update:

- $V(S_t)$ (Previous estimate)
- R_{t+1} (Reward $t+1$)
- $\gamma V(S_{t+1})$ (Discounted value on the next step)
- The sum $R_{t+1} + \gamma V(S_{t+1})$ is labeled as the TD Target.

Алгоритмы и типы задач

Наивный подход:

- Опробовать все возможные стратегии
- Выбрать стратегию с наибольшим ожидаемым выигрышем

Проблемы:

- Огромное количество возможных стратегий
- Стохастические выигрыши

Алгоритмы и типы задач

Подход с использованием функции полезности:

- Оценивание ожидаемого выигрыша начиная с состояния s , при дальнейшем следовании стратегии π :

$$V(s) = E[R|s, \pi]$$

- Ожидаемый выигрыш, при принятии решения a в состоянии s и дальнейшем соблюдении π

$$Q(s, a) = E[R|s, \pi, a]$$

Алгоритмы и типы задач

Подход с использованием функции полезности:

- Если для выбора оптимальной стратегии используется функция полезности Q , то оптимальные действия всегда можно выбрать как действия, максимизирующие полезность
- Если же мы пользуемся функцией V , необходимо либо иметь модель окружения в виде вероятностей $P(s' | s, a)$ что позволяет построить функцию полезности вида $Q(s, a) = \sum_{s'} V(s')P(s' | s, a)$ либо применить метод исполнитель-критик

Алгоритмы и типы задач

Подход с использованием функции полезности:

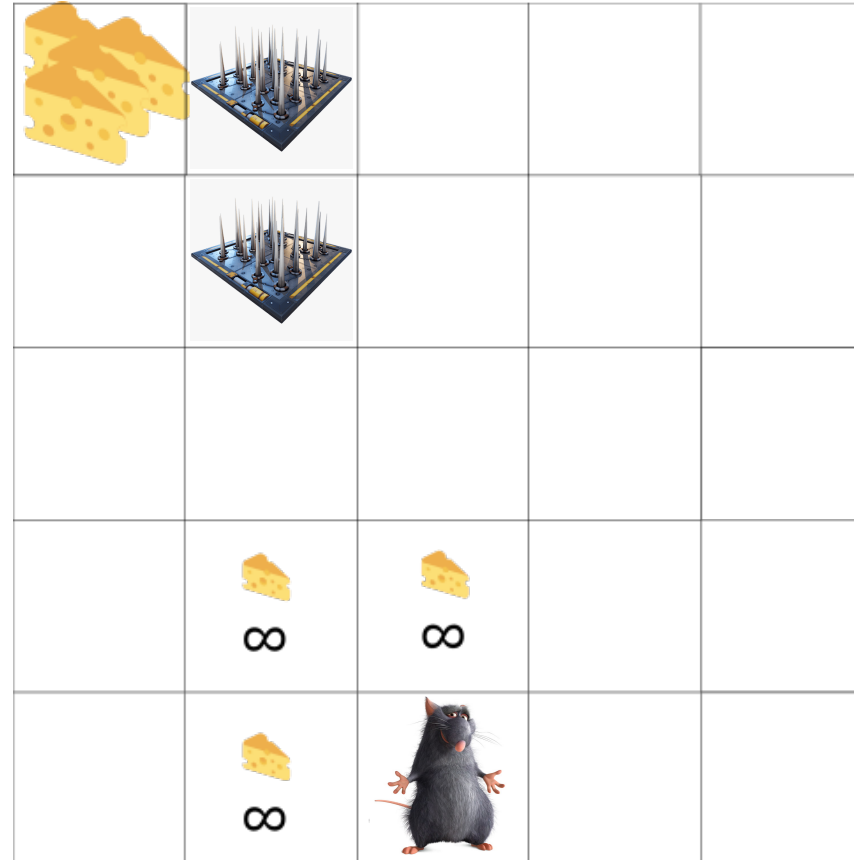
Если задача имеет терминальное состояние, то все хорошо, имея фиксированную стратегию можно оценить $E[R|\cdot]$ усреднив суммарный выигрыш после каждого состояния.

Однако если задача не оканчивается в определенный момент придется использовать алгоритм обучения с временными воздействиями.

Алгоритмы и типы задач

Exploitation-exploration:

- Исследование - это поиск дополнительной информации об окружающей среде
- Эксплуатация - это использование известной информации для получения максимального вознаграждения



Алгоритмы и типы задач

Многорукий бандит:

- A — множество возможных *действий* (ручек автомата)
- $p_a(r)$ — неизвестное распределение *награды* $r \in R \forall a \in A$
- $\pi_t(a)$ — *стратегия* агента в момент $t \forall a \in A$

Игра агента со средой:

- Инициализация стратегии $\pi_1(a)$
- Для всех $t = 1 \dots T$:
 - Агент выбирает действие (ручку) $a_t \sim \pi_t(a)$
 - Среда генерирует награду $r_t \sim p_{a_t}(r)$
 - Агент корректирует стратегию $\pi_{t+1}(a)$

$Q_t(a) = \frac{\sum_{i=1}^t r_i[a_i=a]}{\sum_{i=1}^t [a_i=a]} \rightarrow \max$ — средняя награда в t играх

$Q^*(a) = \lim_{t \rightarrow \infty} Q_t(a) \rightarrow \max$ — ценность действия a

Howdy, partner?



Алгоритмы и типы задач

Жадная и Epsilon-жадная стратегия

Жадная стратегия:

- $P_a = 0 \forall a \in \{1 \dots N\}$ – сколько раз было выбрано действие a
- $Q_a = 0 \forall a \in \{1 \dots N\}$ – текущая оценка математического ожидания награды для действия a

На каждом шаге t :

- Выбираем действие с максимальной оценкой математического ожидания: $a_t = \operatorname{argmax}_{a \in A} Q_a$
- Выполняем действие a_t и получаем награду $R(a_t)$
- Обновляем оценку математического ожидания для действия a_t :

$$P_{a_t} = P_{a_t} + 1, \quad Q_{a_t} = Q_{a_t} + \frac{1}{P_{a_t}}(R(a_t) - Q_{a_t}).$$

Алгоритмы и типы задач

Epsilon-жадная стратегия:

Введем параметр $\epsilon \in (0, 1)$

На каждом шаге t

- Получим значение α – случайной величины равномерно распределенной на отрезке $(0,1)$
- Если $\alpha \in (0, \epsilon)$, то выберем действие $a_t \in A$ случайно и равновероятно, иначе как в жадной стратегии выберем действие с максимальной оценкой математического ожидания
- Обновляем оценки так же как в жадной стратегии

Если $\epsilon > 0$, то в отличие от жадной стратегии на каждом шаге с вероятностью ϵ происходит "исследование" случайных действий.

Алгоритмы и типы задач

Табличные методы

Применимы в случае, когда пространство полей и действий достаточно малы и могут быть записаны в виде векторов и таблиц.

Поскольку найти оптимальную стратегию аналитическим образом не представляется возможным используются итеративные методы.

Алгоритмы и типы задач

Метод кросс-энтропии

- Инициализируем стратегию случайным образом
- Повторяем следующий цикл:
 - Сэмплируем испытания N раз
 - Выбираем $M < N$ наиболее удачных – «элитных» испытаний
 - Меняем стратегию приоритезируя действия из элитных сессий



Алгоритмы и типы задач

Метод кросс-энтропии

- Стратегия задается матрицей:

$$\pi(a|s) = A_{s,a}$$

- М элитных испытаний из N сэмплов:

$$Elite = [(s_0, a_0), (s_1, a_1), \dots, (s_k, a_k)]$$

- Обновляем стратегию в соответствии с действиями предпринятыми в элитных испытаниях

$$\pi(a|s) = \frac{\sum_{s_t, a_t \in Elite} [s_t = s][a_t = a]}{\sum_{s_t \in Elite} [s_t, a_t = s]}$$

Алгоритмы и типы задач

Метод кросс-энтропии

Проблемы:

- Редкие состояния
 - Решение – сглаживание
- $$\pi(a|s) = \frac{\sum_{s_t, a_t \in Elite} [s_t = s][a_t = a] + \lambda}{\sum_{s_t \in Elite} [s_t, a_t = s] + \lambda N_{actions}}$$
- (Опять) стохастические выигрыш
 - Решение – сэмплировать действие для каждого состояния и запустить несколько симуляций с данными парами «состояние – действие». Усреднить результат.



Вопросы

- Как ставка дисконтирования влияет на приоритеты (долгосрочное и краткосрочное вознаграждение)?
- В чем заключается проблема жадной стратегии в случае многорукого бандита?
- Опишите задачу максимизации машинного обучения с подкреплением и опишите все элементы.
- Опишите проблему наивного подхода решения задачи (перебор всех возможных стратегий и выбор стратегии с наибольшим ожидаемым выигрышем)

ИСТОЧНИКИ

- [«Обучение с подкреплением» – статья на neerc.ifmo.ru](http://neerc.ifmo.ru)
- [«Введение в обучение с подкреплением для начинающих» – статья на proglib.io](http://proglib.io)
- [«Марковский процесс принятия решений» – статья на википедии](#)
- [Видеолекция «Crossentropy method» на coursera.org](https://coursera.org)
- [Статья «Summary of Tabular Methods in Reinforcement Learning» towardsdatascience.com](http://towardsdatascience.com)