Обучение с подкреплением

Урусов А.

Факультет компьютерных наук
Высшая школа экономики

14 ноября 2016 г.

Задача о многоруком бандите

Формулировка

Среда содержит

- A - множество действий
- p(a, r) - распределение наград для каждого действия

Цель - найти стратегию, максимизирующую прибыль.
$Q_t(a) = \frac{\sum r_i[a_i=a]}{\sum[a_i=a]}$ - средняя награда за действие a

$\lim_{t\to\infty} E[Q_t(a)] = Q^*(a)$ - ценность действия.
Играем со средой по следующему алгоритму:

- Инициализируется стратегия
- На каждом шаге:
- агент выбирает действие на основе стратегии
- среда возвращает reward
- агент корректирует стратегию

Задача о многоруком бандите

Жадная стратегия заключается в том, что мы выбираем действие с максимальной оценкой ценности, то есть:

$$A_t = \arg\max_{a \in A} (Q_t(a))$$

Недостаток этой стратегии в том, что мы почти не исследуем среду. Эвристика - используем ε-жадную стратегию, то есть будем выбирать действие согласно жадной стратегии с вероятностью $1 - \varepsilon$ и случайное - с вероятностью ε.
Эвристика: ε можно уменьшать со временем.

Задача о многоруком бандите

Метод UCB (upper confidence bound)

Метод UCB заключается в том, что мы выбираем действие с максимальной верхней оценкой ценности, а именно

$$A_t = \arg\max_{a \in A} \left(Q_t(a) + \delta \sqrt{\frac{2 \ln t}{k_t(a)}} \right)$$

Интерпретация:

- Чем меньше $k_i(a)$, тем менее стратегия исследована, соответственно, вероятность должна быть больше
- δ - параметр, чем он больше, тем стратегия более исследовательская

Эвристика: δ можно уменьшать со временем.

Задача о многоруком бандите

Метод Softmax (распределение Больцмана)

Мягкий вариант компромисса между исследованием и применением: выбираем действие случайно из распределения, в котором вероятность равна:

$$\pi_t(a) = \frac{e^{\frac{Q_t(a)}{\tau}}}{\sum\limits_{b \in A} e^{\frac{Q_t(b)}{\tau}}}$$

τ - параметр *температуры*

- При $\tau \to 0$ стратегия стремится к жадной
- При $\tau \to \infty$ стратегия стремится к равномерной, то есть полностью исследовательской

Задача о многоруком бандите

Оценка методов

Генерируется 2000 задач, в каждой из которых:

- $|A| = 10$
- $p_a(r) = N(Q^*(a), 1)$
- $Q^*(a)$ выбирается случайно из $N(0, 1)$.

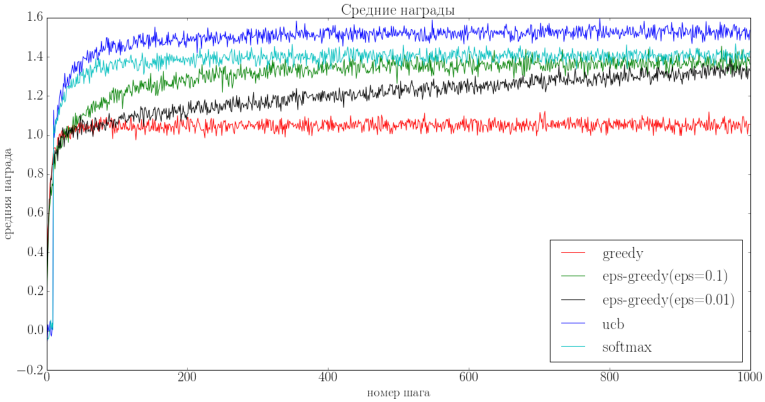
Строятся графики:

- Средняя награда
- Процент оптимальных действий

в зависимости от t, усредненное по всем 2000 задачам.

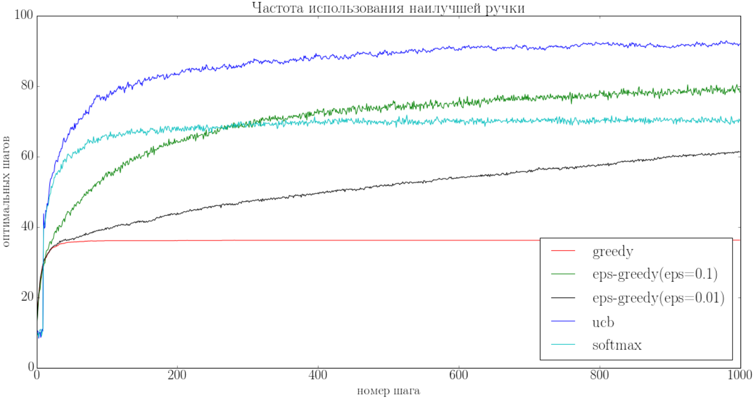
Задача о многоруком бандите

Графики



Задача о многоруком бандите

Графики



Общая задача

Формулировка

Добавляется множество состояний, для каждой пары (действие, состояние) есть распределение наград и распределение состояний.
Игра выглядит так:

- Инициализируется стратегия
- На каждом шаге:
- агент выбирает действие на основе стратегии и предыдущих состояний.
- среда возвращает награду и новое состояние
- агент корректирует стратегию

Общая задача

Приведённая выгода

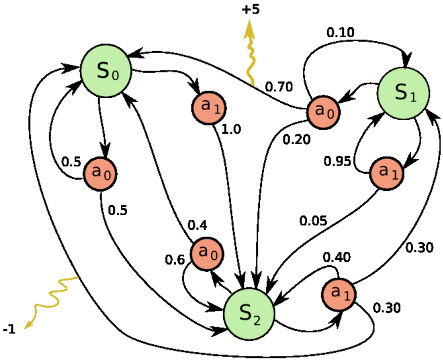
Давайте формализуем то, что мы пытаемся оптимизировать:

$$R_t = r_{t+1} + r_{t+2} * \gamma + r_{t+3} * \gamma^2 + ... = \sum_{i=1}^{+\infty} r_{t+i}\gamma^{i-1}$$

$0 < \gamma < 1$ - коэффициент приведения.
Аналогия: net present value и дефлятор.
При этом если процесс конечный, дополним его до бесконечного нулями.

Общая задача
МППР

Это *Марковский процесс принятия решений (МППР)*, что значит, что следующее состояние зависит только от одного предыдущего, а не от всех.



Общая задача

Ценность состояния

Довольно полезно иметь какую-то оценку того, насколько хорошо состояние, в котором мы находимся.

$$V_\pi(s) = E[R_t|s_t = s] = E\left[\sum_{i=1}^{+\infty} r_{t+i}\gamma^{i-1}|s_t = s\right]$$

Введем также аналогичную величину для действия:

$$Q_\pi(s, a) = E[R_t|s_t = s, a_t = a] = E\left[\sum_{i=1}^{+\infty} r_{t+i}\gamma^{i-1}|s_t = s, a_t = a\right]$$

Общая задача

Уравнение Беллмана

Рекуррентное соотношение для ценности состояния.

$$
\begin{aligned}
V_\pi(s) &= E[\sum_{i=1}^{+\infty} r_{t+i}\gamma^{i-1}|s_t = s] \\
&= E[r_{t+1} + \gamma \sum_{i=1}^{+\infty} r_{t+i+1}\gamma^{i-1}|s_t = s] \\
&= E[r_{t+1} + \gamma V_\pi(s_{t+1})|s_t = s]
\end{aligned}
$$

Из этой формулы очевиден следующий метод оценивания состояния: можно просто использовать в качестве оценки матожидания скользящее среднее:

$$V(s_t) = V(s_t) + \alpha_t(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

Утв. Если $\sum \alpha_t^2 < +\infty$, $\sum \alpha_t = +\infty$ и каждое состояние посещается бесконечное число раз, то оценки сходятся.

Общая задача

Метод SARSA

Метод SARSA (state-action-reward-state-action) основан на идее, похожей на метод TD, но теперь мы оцениваем уже действие, а не состояние, что позволяет нам выбирать наилучшее действие с точки зрения этой оценки.
Игра выглядит следующим образом:

- Инициализируем стратегию $\pi_1(a|s)$ и состояние среды s_1
- На каждом шаге:
- агент выбирает действие a_t из $\pi_t(a|s)$ (например, жадно)
- среда генерирует r_{t+1}, s_{t+1}
- агент разыгрывает еще один шаг a' из $\pi_t(a|s_{t+1})$
- обновляем $\Delta Q(s_t, a_t) = \alpha_t(r_{t+1} + \gamma Q(s_{t+1}, a') - Q(s_t, a_t))$

Общая задача

Q-learning

В методе TD есть проблема: мы считаем стратегию π константной, и на основе этого оцениваем ценность состояния. В реальности же стратегия постоянно меняется. Введем новую метрику оценки $Q^*(s, a)$, которая оценивает ценность *оптимального* действия:

$$Q^*(s, a) = E(r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a)$$

Далее будем оценивать эту величину точно так же, как и в TD - скользящим средним. Заметим, что это позволяет менять стратегию, так же как и SARSA.
Отличия от SARSA: убираем шаг 5 и меняем шаг 6, а именно, берем максимум по a'.

Общая задача
TD(λ)

Идея: можно обновлять не только последнее состояние, но и предыдущие:

$$R_t^{(1)} = r_{t+1} + \gamma V(s_{t+1})$$

$$R_t^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2})$$

$$\ldots$$

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^n V(s_{t+n})$$

Общая задача

TD(γ)

Идея "следов приемлемости": будем корректировать $V(s)$ не только для текущего состояния, но и для недавно пройденных состояний с коэффицентом затухания λ. Обновление $V(S)$:

- $e(s_t) += 1$
- for $s \in S$ do
- $V(s) = V(s) + e(s) * \alpha_t(r_{t+1} + \gamma V(s_{t+1}) - V(s))$
- $e(s) = \lambda\gamma e(s)$

Real-world application

Общая идея

В реальном мире существует большая проблема: состояние - это очень сложный многомерный объект. Поэтому текущим алгоритмам необходимо очень много итераций на обучение: состояний много, и про каждое надо накопить какую-то статистику.
Идея: давайте попробуем оценивать $Q^*(s, a)$ с помощью deep convolutional network.
Более конкретно, будем использовать алгоритм Q-learning, но вместо накопления значений $Q^*(s, t)$ будем использовать $Q^*(s, t; \theta)$, где θ - вектор параметров нейронной сети.

Real-world application

Обновление параметров

Будем хранить историю событий, а именно, множество $D_T = \{e_t = (s_t, a_t, r_t, s_{t+1})\}, 1 \leq t \leq T$. Далее, при обновлении весов будем использовать следующий функционал ошибки:

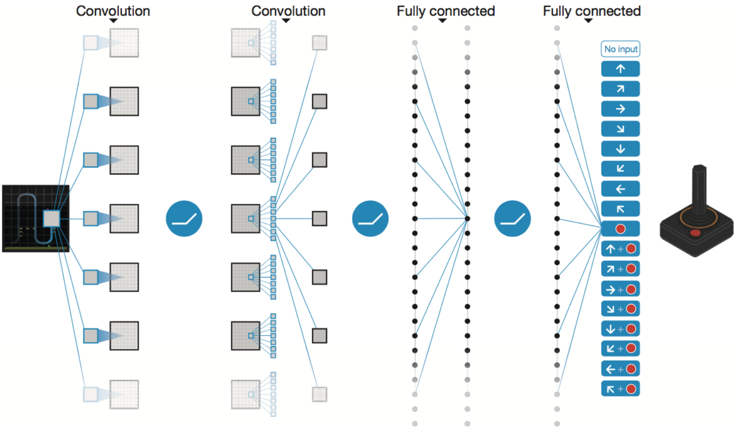
$$L_i(\theta) = \mathbb{E}_{(s,a,r,s')\in U(D)}\left[\left(r + y \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]$$

Здесь θ - новые параметры, а θ^- - старые. Поскольку пересчет - это довольно сложная операция, то будем это делать не на каждом шаге, а через C шагов.

Real-world application

Нейросеть

Сама нейросеть выглядит примерно вот так:



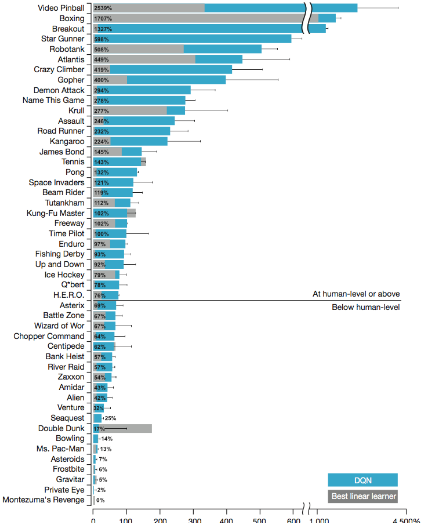
Real-world application

Results

Для тестирования этого алгоритма была использована платформа Atari 2600, которая состоит из 49 различных игр, которые были сделаны таким образом, чтобы человеку было неочевидно в них играть. Результаты оказались очень намного лучше, чем у стандартных алгоритмов обучения с подкреплением, что очень хорошо, поскольку этот агент изначально обладает только минимальной информацией (мы знаем заранее только то, что состояние дается в виде изображения фиксированного размера).

Real-world application

Results



Литература и ссылки

- Sutton, Richard S.; Andrew G. Barto Reinforcement Learning: An Introduction. — MIT Press. 1998, 1998.
- Лекция Воронцова "Обучение с подкреплением"
- Human-level control through deep reinforcement learning