

Обучение с подкреплением

Сергей Николенко



Центр Речевых Технологий, 2012



Outline

- 1 **Агенты с одним состоянием**
 - Постановка задачи
 - Многорукие бандиты



Постановка задачи

- До сих пор задача ставилась так: есть набор «правильных ответов», и нужно его продолжить на всё пространство (supervised learning), или есть набор тестовых примеров без дополнительной информации, и нужно понять его структуру (unsupervised learning).
- Как работает обучение в реальной жизни? Мы далеко не всегда знаем набор правильных ответов, мы просто делаем то или иное действие и получаем результат.



Постановка задачи

- Отсюда и обучение с подкреплением (reinforcement learning).
- Агент взаимодействует с окружающей средой, предпринимая действия; окружающая среда его поощряет за эти действия, а агент продолжает их предпринимать.



Постановка задачи — формально

- На каждом шаге агент может находиться в состоянии $s \in \mathcal{S}$.
- На каждом шаге агент выбирает из имеющегося набора действий некоторое действие $a \in \mathcal{A}$.
- Окружающая среда сообщает агенту, какую награду r он за это получил и в каком состоянии s' после этого оказался.



Пример

- Диалог:

Среда: Агент, ты в состоянии 1; есть 5 возможных действий.

Агент: Делаю действие 2.

Среда: Даю тебе 2 единицы за это. Попал в состояние 5, есть 2 возможных действия.

Агент: Делаю действие 1.

Среда: Даю тебе за это -5 единиц. Попал в состояние 1, есть 5 возможных действий.

Агент: Делаю действие 4.

Среда: Даю тебе 14 единиц за это. Попал в состояние 3, есть 3 возможных действия...

- В этом примере агент успел вернуться в состояние 1 и исследовать ранее не пробовавшуюся опцию 4 (получив за это существенную награду).



Exploitation vs. exploration

- Каждый алгоритм должен и изучать окружающую среду, и пользоваться своими знаниями, чтобы максимизировать прибыль.
- Вопрос — как достичь оптимального соотношения? Та или иная стратегия может быть хороша, но вдруг она не оптимальная?
- Этот вопрос всегда присутствует в обучении с подкреплением.



Пример

- Пример: крестики-нолики.
- Как научить машину играть и *выигрывать* в крестики-нолики?
- Вариант: генетический алгоритм, пусть играют с противником, кто выиграл, тот выживает и даёт потомство.
- Но это очень медленно, не учитывается информация о собственно ходе игры, о том, какие ходы привели к победе; как это сделать?



Пример

- Состояния – позиции на доске.
- Для каждого состояния введём функцию $V(s)$ (value function).
- Подкрепление приходит только в самом конце, когда мы выиграли или проиграли; как его распространить на промежуточные позиции?



Пример

- Можно просто пропагировать обратно: если мы попадали из s в s' , апдейтим

$$V(s) := V(s) + \alpha [V(s') - V(s)] .$$

- Это называется TD-обучение (temporal difference learning), и оно очень хорошо работает на практике.



Агенты с одним состоянием

- Формально всё то же самое, но $|S| = 1$, т.е. состояние агента не меняется. У него фиксированный набор действий A и возможность выбора из этого набора действий.
- Модель: агент в комнате с несколькими игровыми автоматами. У каждого автомата своё ожидание выигрыша. Нужно за ограниченное количество попыток выбрать лучший автомат.



Жадный алгоритм

- Всегда выбирать стратегию, максимизирующую прибыль; прибыль оцениваем как среднее вознаграждение:

$$Q_t(a) = \frac{r_1 + r_2 + \dots + r_{k_a}}{k_a}.$$

- Что не так с таким алгоритмом?



Жадный алгоритм

- Оптимум легко проглядеть, если на начальной выборке не повезёт (что вполне возможно).
- Поэтому полезная эвристика — *оптимизм при неопределённости*. То есть выбирать жадно, но при этом прибыль ожидается весьма оптимистично, и нужны серьёзные отрицательные свидетельства, чтобы отклонить стратегию.



Случайные стратегии

- Стратегия: выбрать действие с наилучшей ожидаемой прибылью с вероятностью $1 - \epsilon$, а с вероятностью ϵ выбрать случайное действие.
- Обычно начинают с маленьких p , затем увеличивают.
- Но алгоритм не различает хорошую альтернативу от бесполезной.



Случайные стратегии

- Исследование по Больцману (Boltzmann exploration):

$$\pi_t(a) = \frac{e^{Q_t(a)/T}}{\sum_{a'} e^{Q_t(a')/T}},$$

где ER — ожидаемая прибыль, T — температура.

- Тоже обычно температура понижается со временем.



Алгоритм линейного вознаграждения–бездействия

- Алгоритм линейного вознаграждения–бездействия (linear reward-inaction) добавляет линейно к вероятности действия a_i , если оно успешно (в бинарном случае):

$$p_i := p_i + \alpha(1 - p_i),$$

$$p_j := p_j - \alpha p_j, \quad j \neq i,$$

а если оно безуспешно, то вероятности сохраняются.



Алгоритм линейного вознаграждения–бездействия

- Алгоритм сходится с вероятностью 1 к вектору из одной единицы и остальных нулей.
- Не всегда сходится к оптимальной стратегии; но вероятность ошибиться можно сделать сколь угодно малой, уменьшая α .
- Есть, соответственно, алгоритм линейного вознаграждения–наказания (linear reward-penalty): тот же самый апдейт, но всегда, даже при безуспешных действиях (тогда вознаграждаем другую ручку).



Интервальные оценки

- Один из способов применить оптимистично-жадный метод.
- Для каждого действия мы храним статистику n и w , а потом вычисляем доверительный интервал для вероятности успеха (с границей $1 - \alpha$) и для выбора стратегии используем верхнюю границу этого интервала.



Интервальные оценки

- Пример: испытания Бернулли (монетка, например). Тогда с вероятностью .95 среднее лежит в интервале

$$\left(\bar{x} - 1.96 \frac{s}{\sqrt{n}}, \bar{x} + 1.96 \frac{s}{\sqrt{n}} \right),$$

где 1.96 берётся из распределения Стьюдента, n — количество испытаний, $s = \sqrt{\frac{\sum (x - \bar{x})^2}{n-1}}$.

- Отличный метод, если вероятностные предположения соответствуют действительности (часто непонятно).



Правило инкрементального обновления

- Как пересчитывать $Q_t(a) = \frac{r_1 + \dots + r_{k_a}}{k_a}$ при поступлении новой информации?
- Довольно просто:

$$\begin{aligned} Q_{k+1} &= \frac{1}{k+1} \sum_{i=1}^{k+1} r_i = \frac{1}{k+1} \left[r_{k+1} + \sum_{i=1}^k r_i \right] = \\ &= \frac{1}{k+1} (r_{k+1} + kQ_k) = Q_k + \frac{1}{k+1} (r_{k+1} - Q_k). \end{aligned}$$



Правило инкрементального обновления

- Это частный случай общего правила – сдвигаем оценку так, чтобы уменьшалась ошибка:

НоваяОценка := СтараяОценка + Шаг [Цель – СтараяОценка] .

- Заметим, что шаг у нас тут непостоянный: $\alpha_k(a) = \frac{1}{k_a}$.
- Изменяя последовательность шагов, можно добиться других эффектов.



Нестационарная задача

- Часто бывает, что выплаты из разных бандитов на самом деле нестационарны, т.е. меняются со временем.
- В такой ситуации имеет смысл давать большие веса недавней информации и маленькие веса – давней.
- Пример: у правила апдейта

$$Q_{k+1} = Q_k + \alpha [r_{k+1} - Q_k]$$

с постоянным α фактически веса затухают экспоненциально:

$$\begin{aligned} Q_k &= Q_{k-1} + \alpha [r_k - Q_{k-1}] = \alpha r_k + (1 - \alpha) Q_{k-1} = \\ &= \alpha r_k + (1 - \alpha) \alpha r_{k-1} + (1 - \alpha)^2 Q_{k-2} = (1 - \alpha)^k Q_0 + \sum_{i=1}^k \alpha (1 - \alpha)^{k-i} r_i \end{aligned}$$



Нестационарная задача

- Такое правило апдейта не обязательно сходится, но это и хорошо – мы хотим следовать за целью.
- Общий результат – правило апдейта сходится, если последовательность весов удовлетворяет

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty \quad \text{и} \quad \sum_{k=1}^{\infty} \alpha_k^2(a) < \infty.$$

- Например, для $\alpha_k(a) = \frac{1}{k_a}$ явно сходится.



ОПТИМИЗМ

- Можно ускорить и упростить поиск, если начать с оптимистичных значений средних.
- Давайте выставим $Q_0(a)$ такими большими, что любое реальное вознаграждение будет «разочаровывать», но не слишком большими – нам нужно, чтобы достаточно быстро Q_0 усреднилось с реальными r_i .
- Тогда даже тривиальная жадная стратегия достаточно быстро обучится.



Сравнение подкреплений

- Однако интуиция тут в том, что мы ищем «большие» вознаграждения. А что такое «большие»?
- Можно сравнивать со средним вознаграждением по всем ручкам; это называется *метод сравнения подкреплений* (reinforcement comparison).
- В таких методах обычно нет action values Q_k , есть предпочтения $p_t(a)$; вероятности можно получить, например, по Больцману:

$$\pi_t(a) = \frac{e^{p_t(a)}}{\sum_{a'} e^{p_t(a')}}.$$



Сравнение подкреплений

- И на каждом шаге мы обновляем среднее вознаграждение и предпочтения:

$$\begin{aligned}\bar{r}_{t+1} &= \bar{r}_t + \alpha (r_t - \bar{r}_t), \\ p_{t+1}(a) &= p_t(a_t) + \beta (r_t - \bar{r}_t).\end{aligned}$$



Методы погони

- Методы погони (pursuit methods) хранят и оценки ожидаемой выплаты, и предпочтения действий, и предпочтения «гонятся» за оценками.
- Например, $\pi_t(a)$ – вероятность выбрать a во время t ; после шага t мы ищем жадную стратегию

$$a_{t+1}^* = \arg \max_a Q_{t+1}(a)$$

и подправляем π в сторону жадной стратегии:

$$\begin{aligned}\pi_{t+1}(a_{t+1}^*) &= \pi_t(a_{t+1}^*) + \beta [1 - \pi_t(a_{t+1}^*)], \\ \pi_{t+1}(a) &= \pi_t(a) + \beta [0 - \pi_t(a)].\end{aligned}$$



Thank you!

Спасибо за внимание!