5. ЛЕКЦИЯ. Классическая теория 3

Продолжаем рассматривать ***Табличные алгоритмы***

Exploration-exploitation дилемма (дилемма исследование-использование)

Так какой же стратегией играть со средой, чтобы получать траектории? Коли мы учим Q^*, у нас на очередном шаге есть текущее приближение $\pi_k(s) := arg\max_a Q_k(s,a)$, и мы можем использовать (exploit) эти знания.

Теорема: Собирая траектории при помощи жадной стратегии, есть риск не сойтись к оптимальной.

Действительно, детерминированные стратегии не позволяют получить свойство infinite visitation (бесконечное посещение): многие пары s, a просто приниципиально не встречаются в порождаемых ими траекториях. В частности, из-за неё нельзя ограничиваться рассмотрением только класса детерминированных стратегий, хоть и была рассмотрена теорема о существовании в нём оптимальной стратегии: мы показали, что для сбора данных — взаимодействия со средой — детерминированные стратегии не подходят.

Теорема: Любая стратегия, для которой $\forall s, a: \mu(a|s) > 0$, удовлетворяет условию infinite visitation (бесконечного посещения), то есть с отличной от нуля вероятностью в траектории, порождённой π, встретится любая пара s, a.

Если мы воспользуемся любой такой стохастической стратегией, мы попадём под действие теоремы о сходимости. Совершая случайные действия, мы рискуем творить ерунду, но занимаемся исследованием (exploration) — поиском новых возможностей в среде. Означает ли это, что нам достаточно взять полностью случайную стратегию, которая равновероятно выбирает из множества действий, отправить её в среду порождать переходики s, a, r, s', обучать на них Q-функцию, и на выходе в пределе мы получим Q^*.

Иначе говоря, исследование — корректный, но неэффективный способ генерации данных. Использование — куда более эффективный метод, позволяющий быстро добираться до «труднодоступных областей» в среде — такими областями обычно являются области с высокой наградой, иначе задача RL скорее всего не является особо интересной — и быстрее набирать сэмплы для обновлений пока ещё редко обновлённых ячеек $Q(s,a)$. Но это «некорректный» способ: детерминированная стратегия может застрять в локальном оптимуме и никогда не увидеть, что в каком-то месте другое действие даёт ещё большую награду. Обсудим базовые варианты, как можно решать этот exploration-exploitation trade-off (компромисс между исследованием-использованием) в контексте вычисления оптимальной Q-функции методом временных разностей.

Нам нужно взять нашу стратегию использования $\pi(s) := arg\max_a Q(s,a)$ и что-нибудь с ней поделать так, чтобы она стала формально стохастичной.

Определение: ε-жадной (ε-greedy) называется стратегия

$$\mu(s) := \begin{cases} \text{a} \sim \text{Uniform(A)} \text{ с вероятностью } \varepsilon; \\ arg\max_a Q(s,a) \qquad\qquad \text{иначе.} \end{cases}$$

Определение: Больцмановской с температурой τ называется стратегия

$$\mu(a|s) := soft\max_a \left(\frac{Q(s,a)}{\tau}\right)$$

Первый вариант никак не учитывает, насколько кажутся хорошими другие действия помимо жадного, и, если решает «исследовать», выбирает среди них случайно. Второй же вариант будет редко выбирать очень плохие действия (это может быть крайне полезным свойством), но чувствителен к масштабу приближения Q-функции, что обычно менее удобно и требует настройки температуры τ.

Реплей буфер (буфер воспроизведения)

Итак, теперь можно собрать классический алгоритм табличного RL под названием Q-learning (обучение). Это метод временных разностей для вычисления оптимальной Q-функции с ε-жадной стратегией исследования.

Алгоритм: Q-learning

Гиперпараметры: α — параметр экспоненциального сглаживания, ε — параметр исследований

Инициализируем $Q(s,a)$ произвольно для всех $s \in \mathcal{S}, a \in \mathcal{A}$
Наблюдаем s_0
На k-ом шаге:

1. с вероятностью ε играем $a_k \sim \text{Uniform}(\mathcal{A})$, иначе $a_k = \underset{a_k}{\text{argmax}}\, Q(s_k, a_k)$
2. наблюдаем r_k, s_{k+1}
3. обновляем $Q(s_k,a_k) \leftarrow Q(s_k,a_k) + \alpha\left(r_k + \gamma \max_{a_{k+1}} Q(s_{k+1},a_{k+1}) - Q(s_k,a_k)\right)$

Q-learning является типичным представителем off-policy (вне политики) алгоритмов: нам нигде не требовались сэмплы взаимодействия со средой конкретной стратегии. Наша стратегия сбора данных могла быть, вообще говоря, произвольной. Это крайне существенное свойство, потому что в таких алгоритмах возможно обучение с буфера: допустим, некоторый «эксперт» π^{expert} провёл много-много сессий взаимодействия со средой и собрал для нас кучу траекторий. Рассмотрим их как набор переходов. Тогда мы можем, вообще не взаимодействуя больше со средой, провести обучение Q^* с буфера: сэмплируем равномерно переход (s,a,r,s') и делаем обновление ячейки $Q(s,a)$.

Определение: Для данного буфера — набора переходов (s,a,r,s') —

будем называть эмпирическим MDP (empirical MDP – Марковских процессов принятия решений) MDP с тем же пространством состояний, действий и функций награды, где функция переходов задана следующим образом:

$$\hat{p}(s'|s,a) := \frac{N(s,a,s')}{N(s,a)}$$

где $N(s,a,s')$ — число троек s,a,s0, входящих в буфер, $N(s,a)$ — число пар s,a, входящих в буфер

Утверждение: При выполнении условий на learning rate (скорости обучения), Q-learning, запущенный с фиксированного буфера, выучит Q^* для эмпирического MDP.

Естественно, если буфер достаточно большой, то мы выучим очень близкую Q^* к настоящей. Ещё более интересно, что Q-learning как off-policy (вне политики) алгоритм может обучаться со своего собственного опыта — со своего же собственного буфера, составленного из порождённых очень разными стратегиями переходов.

Определение: Реплей буфер (replay buffer, experience replay) — это память со всеми собранными агентом переходами $(s,a,r,s0,done)$.

Определение: Реплей буфер (replay buffer, experience replay) — это память со всеми собранными агентом переходами $(s,a,r,s',done)$. (done – сделанный)

Если взаимодействие со средой продолжается, буфер расширяется, и распределение, из которого приходит s', становится всё больше похожим на настоящее $p(s'|s,a)$; на факт сходимости это не влияет.

Алгоритм: Q-learning with experience replay – Q-обучение с повторением опыта

Гиперпараметры: α — параметр экспоненциального сглаживания, ε — параметр исследований

Инициализируем $Q(s,a)$ произвольно для всех $s \in \mathcal{S}, a \in \mathcal{A}$
Наблюдаем s_0
На k-ом шаге:

1. с вероятностью ε играем $a_k \sim \text{Uniform}(\mathcal{A})$, иначе $a_k := \underset{a_k}{\text{argmax}}\, Q(s_k, a_k)$
2. наблюдаем r_k, s_{k+1}
3. кладём s_k, a_k, r_k, s_{k+1} в буфер
4. сэмплируем случайный переход s, a, r, s' из буфера
5. обновляем $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s',a')\right)$

Реплей буфер — ключевое преимущество off-policy (вне политики) алгоритмов: мы сможем с каждого перехода потенциально обучаться бесконечное количество раз. Это развязывает нам руки в плане соотношения числа собираемых данных и количества итераций обновления нашей модели,

поскольку здесь мы можем, в общем-то, сами решать, сколько переходов на один шаг обновления мы будем проводить. Проявляется это в том, что в Q-learning, как и в любом другом off-policy алгоритме, есть два независимых этапа: сбор данных (взаимодействие со средой при помощи ε-жадной стратегии и сохранение опыта в памяти — первые три шага), и непосредственно обучение (сэмплирование перехода из буфера и обновление ячейки — шаги 4-5). Можно проводить несколько шагов сбора данных на одно обновление, или наоборот: несколько обновлений на один шаг сбора данных, или даже проводить эти процессы независимо параллельно

SARSA

Рассмотрели off-policy (вне политики) алгоритм: мы умеем оценивать нашу текущую стратегию ($arg\max_a Q(s,a)$, неявно сидящий внутри формулы обновления), используя сэмплы другой стратегии. Иными словами, у нас в алгоритме различаются понятия целевой политики (target policy) — стратегии, которую алгоритм выдаст по итогам обучения, она же оцениваемая политика, то есть та политика, для которой мы хотим посчитать оценочную функцию — и политики взаимодействия (behaviorpolicy) — стратегии взаимодействия со средой, стратегии с подмешанным исследованием. Это различие принципиально: оптимальны детерминированные стратегии, а взаимодействовать со средой мы готовы лишь стохастичными стратегиями. У этого «несовпадения» есть следующий эффект.

Возможны ситуации, когда небезопасное поведение во время обучения не является проблемой, но для, например, реальных роботов сбивать пешеходов из-за случайных действий — не самая лучшая идея. Что, если мы попробуем как-то «учесть» тот факт, что мы обязаны всегда заниматься исследованиями? То есть внутри оценочной функции должно закладываться, что «оптимальное» поведение в будущем невозможно, а возможно только около-оптимальное поведение с подмешиванием исследования. Для примера будем рассматривать ε-жадную стратегию, пока что с константным ε.

Рассмотрим очень похожий на Q-learning алгоритм. Будем использовать пятёрки s, a, r, s', a' (отсюда и название) прямо из траекторий нашего взаимодействия со средой и апдейтить текущее приближение Q-функции по формуле

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r(s,a) + \gamma Q(s',a') - Q(s,a))$$

Алгоритм: SARSA

Гиперпараметры: α — параметр экспоненциального сглаживания, ε — параметр исследований

Инициализируем $Q(s, a)$ произвольно для всех $s \in \mathcal{S}, a \in \mathcal{A}$
Наблюдаем s_0, сэмплируем $a_0 \sim \mathbf{Uniform}(\mathcal{A})$
На k-ом шаге:

1. наблюдаем r_k, s_{k+1}
2. с вероятностью ε играем $a_{k+1} \sim \mathbf{Uniform}(\mathcal{A})$, иначе $a_{k+1} = \underset{a_{k+1}}{\operatorname{argmax}} Q(s_{k+1}, a_{k+1})$
3. обновляем $Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha\left(r_k + \gamma Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k)\right)$

Попробуем понять формальнее, что происходит в таком алгоритме. Можно считать, что он чередует два шага: обновление, которое учит Q^{π} для текущей π, и, неявно, некий аналог policy improvement-а: замены π на ε-greedy(жадный). Именно при помощи обновлённой стратегии мы будем взаимодействовать со средой на следующем шаге, то есть полагаем $\mu \equiv \pi$ («переходим в on-policy режим»). Является ли такое обновление policy improvement-ом (допустим, для идеально посчитанной Q^{π})? Вообще говоря, нет, но наши стратегии π, которые мы рассматриваем — не произвольные. Они все ε-жадные. Введём такое определение.

Определение: Будем говорить, что стратегия π — ε-мягкая (ε-soft), если $\forall s, a: \pi(a|s) \geq \frac{\varepsilon}{|\mathcal{A}|}$

Утверждение: Если π_1 — ε-мягкая, то $\pi_2 := \varepsilon - \text{жадный}(Q^{\pi_1})$ не хуже, чем π_1

Давайте изменим постановку задачи: скажем, что мы запрещаем к рассмотрению стратегии, «похожие на детерминированные». Наложим ограничение в нашу задачу оптимизации: скажем, что стратегия обязательно должна быть ε-мягкой. В такой задаче будут свои оптимальные оценочные функции.

Определение: Для данного MDP оптимальными ε-мягкими оценочными функциями назовём:

$$V^*_{\varepsilon-soft(s)} := \max_{\pi \in \varepsilon-soft} V^{\pi}(s)$$

$$Q^*_{\varepsilon-soft(s,a)} := \max_{\pi \in \varepsilon-soft} Q^{\pi}(s, a)$$

Как тогда будет выглядеть принцип оптимальности? Раньше нужно было выбирать самое хорошее действие, но теперь так делать нельзя. Проводя аналогичные рассуждения, можно показать, что теперь оптимально выбирать самое хорошее действие с вероятностью $1 - \varepsilon + \varepsilon|\mathcal{A}|$, а всем остальным действиям выдавать минимально разрешённую вероятность $\varepsilon|\mathcal{A}|$.

Утверждение: Стратегия π оптимальна в классе ε-мягких стратегий тогда и только тогда, когда $\forall s, a$ таких, что $a \notin Argmax Q^{\pi}(s,a)$ верно $\pi(a|s) = \frac{\varepsilon}{|\mathcal{A}|}$

Утверждение: Уравнения оптимальности, соответственно, теперь выглядят так:

$$Q^*_{\varepsilon\,-soft(s,a)} = \mathrm{r} + \gamma \mathbb{E}_{s'}\left[(1-\varepsilon)\max_{a'} Q^*_{\varepsilon\,-soft}(s',a') + \frac{\varepsilon}{|\mathcal{A}|}\sum_{a'} Q^*_{\varepsilon\,-soft}(s',a')\right]$$

Мы теперь понимаем, что наша формула обновления — просто метод решения такого уравнения оптимальности: сэмплируя a' из ε-жадной стратегии, мы просто стохастически аппроксимируем по мат.ожиданию из ε-жадной стратегии. Мы могли бы, вообще говоря, взять это мат.ожидание полностью явно, сбив таким образом дисперсию:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r(s,a) + \gamma \mathbb{E}_{a' \sim \pi} Q(s',a') - Q(s,a))$$

где мат.ожидание по a' по определению π равно

$$\mathbb{E}_{a' \sim \pi} Q(s',a') = (1-\varepsilon)\max_{a'} Q(s',a') + \frac{\varepsilon}{|\mathcal{A}|}\sum_{a'} Q(s',a')$$

Такая схема называется Expected (ожидание) SARSA — «SARSA, в которой взяли мат.ожидание». Мы всё ещё учим Q-функцию текущей политики, но не используем сэмплиз текущей траектории, а вместо этого берём мат.ожидание по действиям из уравнения Беллмана. Вообще говоря, в такой схеме мы работаем не с пятёрками s, a, r, s', a', а с четвёрками s, a, r, s' (несмотря на название).

Алгоритм: Expected-SARSA

Гиперпараметры: α — параметр экспоненциального сглаживания, ε — параметр исследований

Инициализируем $Q(s,a)$ произвольно для всех $s \in \mathcal{S}, a \in \mathcal{A}$
Инициализируем π_0 произвольно
Наблюдаем s_0
На k-ом шаге:

1. играем $a_k \sim \pi_k(a_k \mid s_k)$
2. наблюдаем r_k, s_{k+1}
3. π_{k+1} есть с вероятностью ε выбрать $a_{k+1} \sim \mathbf{Uniform}(\mathcal{A})$, иначе $a_{k+1} \coloneqq \underset{a_{k+1}}{\mathbf{argmax}}\, Q(s_{k+1}, a_{k+1})$
4. обновляем $Q(s_k,a_k) \leftarrow Q(s_k,a_k) + \alpha\left(r_k + \gamma \mathbb{E}_{a_{k+1}\sim\pi_{k+1}} Q(s_{k+1},a_{k+1}) - Q(s_k,a_k)\right)$

Соответственно, и SARSA, и Expected SARSA будут сходиться уже не к обычной оптимальной оценочной функции, а к $Q^*_{\varepsilon\,-soft}(s,a)$ — оптимальной оценочной функции в семействе ε-мягких стратегий.

Принципиальное отличие схемы SARSA от Q-learning в том, что мы теперь учимся ровно на тех же сэмплах действий a', которые отправляем в среду. Наши

behavior (поведение) и target policy (целевая политика) теперь совпадают: для очередного шага алгоритма нужно сделать шаг в среде при помощи текущей π, и поэтому мы должны учиться онлайн, «в on-policy (с политикой) режиме».

Если попробовать запустить SARSA обучаться на собственной истории, то мы вообще творим полную ахинею: на каждом шаге мы движемся в сторону Q-функции для той стратегии π, которая породила засэмплированный переход (например, если переход был засэмплирован откуда-то из начала обучения — скорее всего случайной или хуже). Такой алгоритм не просто будет расходиться, но и не будет иметь никакого смысла. Поэтому SARSA нельзя (в таком виде, по крайней мере) запустить с реплей буфера.

Bias-VarianceTrade-Off (Компромисс смещения и дисперсии)

Дилемма смещения-разброса

Мы обсудили два вида бэкапов, доступных в model-free обучении: Монте-Карло бэкап и Temporal-Difference (временная разница) бэкап. На самом деле, они очень похожи, поскольку делают обновление вида

$$Q(s,a) \leftarrow Q(s,a) + \alpha(y_Q - Q(s,a))$$

и отличаются лишь выбором y_Q: Монте-Карло берёт reward-to-go (награда на вынос), а TD-backup — одношаговую оценку с использованием уже имеющейся аппроксимации Q-функции:

$$y_Q := r(s,a) + \gamma Q(s',a')$$

Какой из этих двух вариантов лучше? Мы уже обсуждали недостатки Монте-Карло оценок: высокая дисперсия, необходимость играть до конца эпизодов, игнорирование структуры получаемой награды и потеря информации о соединениях состояний. Но не то, чтобы одношаговые оценки сильно лучше: на самом деле, они обладают полностью противоположными свойствами и проблемами.

Да, одношаговые оценки аппроксимируют решение одношаговых уравнений Беллмана и приближают алгоритм динамического программирования: поэтому они не теряют информации о том, на каком шаге какой сигнал от среды был получен, и сохраняют информацию о сэмплах s' из функции переходов; в том числе, как мы видели, одношаговые алгоритмы могут использовать реплей буфер, по сути и хранящий собранную выборку таких сэмплов. Взамен в одношаговых алгоритмах возникает проблема распространения сигнала.

Вместо высокой дисперсии Монте-Карло оценок в одношаговых оценках нас ждёт большое смещение (bias): если y_Q оценено через нашу же текущую

аппроксимацию через бутстрапирование (практический компьютерный метод исследования распределения статистик вероятностных распределений, основанный на многократной генерации выборок методом Монте-Карло на базе имеющейся выборки), то оно не является несмещённой оценкой искомой $Q^{\pi}(s,a)$ и может быть сколь угодно «неправильным». Как мы увидели, гарантии сходимости остаются, но естественно, что методы стохастической аппроксимации из-за смещения будут сходиться сильно дольше экспоненциального сглаживания, которому на вход поступают несмещённые оценки искомой величины. Но дисперсия y_Q в temporal difference (временная разница) обновлениях, например, в алгоритме Q-learning, конечно, сильно меньше дисперсии Монте-Карло оценок: внутри нашей аппроксимации Q-функции уже усреднены все будущие награды, то есть «взяты» все интегралы, относящиеся к будущим после первого шага наградам. Итого одношаговая оценка y_Q — случайная величина только от s',а не от всего хвоста траектории s', a', s'' ….Выбор между оценками с высокой дисперсией и отсутствием смещения (Монте-Карло) и оценками с низкой дисперсией и большим смещением (одношаговые оценки) — особая задача в обучении с подкреплением, называемая bias-variance trade-off (компромисс между смещением и дисперсией).

N-step Temporal Difference (N-шагов временной разницы)

Какие есть промежуточные варианты между одношаговыми оценками и Монте-Карло оценками? Давайте заглядывать в будущее не на один шаг и не до самого конца, а на N шагов. Итак, пусть у нас есть целый фрагмент траектории

Определение: Фрагмент траектории $s, a, r, s', a', r', s'', a'', r'', \dots s^{(N)}, a^{(N)}$, где $s^{(N)}, a^{(N)}$ — состояние и действие, которое агент встретил через N шагов, будем называть роллаутом (rollout) длины N.

Определение: N-шаговой оценкой (N-step estimation) для Qπ(s,a) назовём следующий таргет:

$$y_Q := r + \gamma r' + \gamma^2 r'' + \dots + \gamma^{N-1} r^{(N-1)} + \gamma^N Q(s^{(N)}, a^{(N)})$$

Хтобы выучить $Q^{\pi}(s,a)$, нужно, чтобы состояния приходили из функции переходов, а действия — из оцениваемой стратегии π. Другими словами, роллаут, использованный для построения таргета, должен быть порождён оцениваемой стратегией.

Понятно, что при $N \to \infty$ оценка переходит в Монте-Карло оценку. С увеличением N всё больше интегралов заменяется на Монте-Карло оценки, и растёт дисперсия; наше же смещённое приближение будущих наград

$Q(s^{(N)}, a^{(N)})$, которое может быть сколь угодно неверным, домножается на γ^N, и во столько же раз сбивается потенциальное смещение; с ростом N оно уходит в ноль. Замешивая в оценку слагаемое $r + \gamma r' + \gamma^2 r'' + \cdots + \gamma^{N-1} r^{(N-1)}$ мы теряем информацию о том, что из этого в какой момент было получено, но и начинаем распространять сигнал в N раз «быстрее» одношаговой оценки.

Сразу заметим, что при $N > 1$ нам необходимо иметь в N-шаговой оценке сэмплы $a' \sim \pi(a|s), s'' \sim p(s''|s', a')$. Это означает, что мы не можем получить такую оценку с буфера: действительно, в буфере для данной четвёрки s, a, r, s' не лежит сэмпл $a' \sim \pi(a|s)$, ведь в произвольном буфере a' генерируется стратегией сбора данных (старой версией стратегией или «экспертом»). Поэтому обучаться на многошаговые оценки с буфера не выйдет; по крайней мере, без какой-либо коррекции.

Также отметим, что все рассуждения одинаковы применимы как для обучения Q^π, таки V^π. Для V-функции общая формула обновления выглядит так:

$$V(s) \leftarrow V(s) + \alpha(y_V - V(s))$$

где y_V — reward-to-go (награда на вынос) при использовании Монте-Карло оценки, и $y_V := r(s, a) + \gamma V(s')$ для одношагового метода временных разностей, где $a \sim \pi(a|s)$ (сэмплирован из текущей оцениваемой стратегии), $s' \sim p(s'|s, a)$. Соответственно, для V-функции обучаться с буфера (без каких-либо коррекций) невозможно даже при $N = 1$, поскольку лежащий в буфере a сэмплирован из стратегии сбора данных, а не оцениваемой π. Для простоты и наглядности будем обсуждать обучение V^π. Также введём следующие обозначения:

Определение: Введём такое обозначение N-шаговой временной разности (N-step temporal difference) для пары s, a:

$$\Psi_{(N)}(s, a) := \sum_{t=0}^{N-1} \gamma^t r^{(t)} + \gamma^N V(s^{(N)}) - V(s)$$

где V — текущая аппроксимация V-функции, $s, a, \ldots s^{(N)}$ — роллаут, порождённый π.

Ранее мы обсуждали temporal difference (временная разница), в котором мы сдвигали нашу аппроксимацию на $\Psi_{(1)}(s, a)$:

$$V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s)) = \mathrm{V(s)} + \alpha\Psi_{(1)}(s, a)$$

Теперь же мы можем обобщить наш метод, заменив оценку V-функции на многошаговую оценку:

$$V(s) \leftarrow V(s) + \alpha\Psi_{(N)}(s, a)$$

Но какое N выбрать?

Интерпретация через Credit Assingment (переуступка кредита)

Вопрос, обучаться ли со смещённых оценок, или с тех, которые имеют большую дисперсию, имеет прямое отношение к одной из центральных проблем RL — credit assingment (переуступка кредита). На самом деле, это ровно та же самая проблема.

Рассмотрим проблему credit assingment-а: за какие будущие награды «в ответе» то действие, которое было выполнено в некотором состоянии s? «Идеальное» решение задачи — значение $A^{\pi}(s,a)$, но на практике у нас нет точных значений оценочных функций, а есть лишь аппроксимация, допустим, $V \approx V^{\pi}$. Положим, мы знаем сэмпл траектории $a, s', r, s'', a'', \ldots$ до конца эпизода. С одной стороны, мы можем выдавать кредит, полностью опираясь на аппроксимацию:

$$Q^{\pi}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} V^{\pi}(s') \approx r(s,a) + \gamma V(s')$$
$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \approx r(s,a) + \gamma V(s') - V(s)$$

Здесь проблема в том, что наша аппроксимация может быть сколь угодно неверна, и выдавать полную ерунду. Более «безопасный» с этой точки зрения способ — в приближении Q-функции не опираться на аппроксимацию и использовать reward-to-go (награда на вынос):

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \approx \sum_{t \geq 0} \gamma^t r^{(t)} - V(s)$$

У этого второго способа выдавать кредит есть важное свойство, которого нет у первого: в среднем такой кредит будет больше у тех действий, которые действительно приводят к более высокой награде. То есть, если a_1, a_2 таковы, что $Q^{\pi}(s,a_1) > Q^{\pi}(s,a_2)$, то при любой аппроксимации $V(s)$ среднее значение кредита для s, a_1 будет больше s, a_2. Это и есть свойство несмещённости Монте-Карло оценок в контексте проблемы выдачи кредита.

Беда Монте-Карло в том, что в этот кредит закладывается награда не только за выбор a в s, но и за будущие решения. То есть: представьте, что через десять шагов агент выбрал действие, которое привело к +100. Эта награда +100 попадёт и в кредит всех предыдущих действий, хотя те не имели к нему отношения.

Видно, что эти два крайних способа выдачи кредита есть в точности «градиент» $y_V - V(s)$, по которому учится V-функция. Фактически, занимаясь обучением V-функции и используя формулу

$$V(s) \leftarrow V(s) + \alpha \Psi(s,a)$$

мы выбором функции $\Psi(s,a)$ по-разному проводим credit assingment (переуступка кредита).

Таким образом видна новая интерпретации (компромисс между смещением и дисперсией) bias-variance trade-off-а: и «дисперсия» с этой точки зрения имеет смысл возложения на действие ответственности за те будущие награды, к которым это действие отношения на самом деле не имеет. Одношаговая оценка $\Psi_{(1)}$ говорит: действие влияет только на награду, которую агент получит тут же, и весь остальной сигнал будет учтён в аппроксимации V-функции. Монте-Карло оценка $\Psi_{(\infty)}$ говорит, что действие влияет на все будущие награды. А N-шаговая оценка $\Psi_{(\mathrm{N})}$ говорит странную вещь: действие влияет на события, который происходят с агентом в течение следующих N шагов.

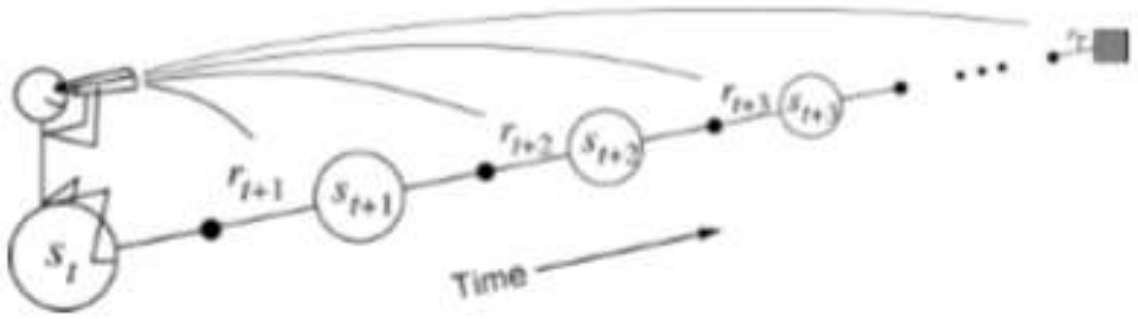

Рис. 5.1

Как и в любом trade-off (компромиса), истина лежит где-то по середине. Однако подбирать на практике «хорошее» N, чтобы получить оценки с промежуточным смещением и дисперсией, затруднительно. Но что ещё важнее, все N-шаговые оценки на самом деле неудачные. Во-первых, они плохи тем, что не используют всю доступную информацию: если мы знаем будущее на M шагов вперёд, то странно не использовать награды за шаг за все эти M шагов, и странно не учесть прогноз нашей аппроксимации оценочной функции $V(s^{(t)})$ для всех доступных t. Во-вторых, странно, что в стационарной задаче, где всё инвариантно относительно времени, у нас появляется гиперпараметр, имеющий смысл количества шагов — времени. Поэтому нас будет интересовать далее альтернативный способ «интерполировать» между Монте-Карло и одношаговым и оценками. Мы всё равно оттолкнёмся именно от N-шаговых оценок, поскольку понятно, что эти оценки «корректны»: они направляют нас к решению уравнений Беллмана, для которых искомая V^{π} является единственной неподвижной точкой.

Эти N-шаговые оценки, посмотрев на задачу под следующим углом: мы знаем сэмпл будущего (хвост траектории) и хотим выдать кредит «настоящему»: самому первому действию. Такой взгляд на оценки называется «forward view» (вид вперед): мы после выполнения a из s знаем «вперёд» своё будущее и можем обновить оценочную функцию для этой пары.

Backward View (Вид назад)

Оказывается, мы можем посмотреть на задачу немного по-другому: можно, используя настоящее, обновлять кредит для прошлого. Рассмотрим эту идею, развив пример с кафе

Пример: Сядем в кафе (s) и захотим вернуться домой. Текущая аппроксимация даёт $-V(s) = 30$ минут (езда до дома). Делаем один шаг в среде: тратим одну минуту $(-r)$ и обнаруживаем пробку (s'). Новая оценка времени возвращения даёт: $-V(s') = 40$ минут, соответственно с нами случилась одношаговая временная разность $\Psi_{(1)}(s,a) = 41 - 30 = 11$ минут, которая позволяет нам корректировать $V(s)$.

Давайте сделаем ещё один шаг в среде: тратим одну минуту$-r'$, видим пожар s'' и получаем новую оценку $-V(s'') = 60$ минут. Тогда мы можем посчитать как одношаговую временную разность для пары s', a', равную $\Psi_{(1)}(s'0, a') = 61 - 40 = 21$ минуте, и уточнить свою аппроксимацию V-функции для состояния с пробкой; а можем посчитать и двухшаговую временную разность для кафе: мы потратили две минуты $r + r'$ на два шагаинашанынешнееприближениеравно60минутам.Итогодвухшаговая временная разность равна $\Psi_{(2)}(s,a) = 62 - 30 = 32$ минуты. Forward view (вид вперед) говорит следующее: если мы хотим учиться на двухшаговые оценки вместо одношаговых, то нам не следовало на первом шаге использовать 11 минут для обновления $V(s)$, а нужно было дождаться второго шага, узнать двух шаговую ошибку в 32 минуты и воспользоваться ей.

Но понятно, что двухшаговая ошибка это сумма двух одношаговых ошибок. Наши 32 минуты ошибки — это 11 минут ошибки после выхода из кафе в пробку плюс 21 минута ошибки от выхода из пробки в пожар. Давайте после первого шага используем уже известную часть ошибки в 11 минут, а на втором шаге, если мы готовы обучаться с двух шаговых ошибок, возьмём и добавим недостающие 21 минуту.

Формализуем эту идею. Пусть мы взаимодействуем в среде при помощи стратегии π, которую хотим оценить; также будем считать (скорость обучения) learning rate α константным. После совершения первого шага «из кафе» мы можем, зная s, a, r, s' сразу же обновить нашу V-функцию:

$$V(s) \leftarrow V(s) + \alpha\Psi_{(1)}(s,a)$$

Затем мы делаем ещё один шаг в среде, узнавая a', r', s'' и временную разность за этот случившийся шаг $\Psi_{(1)}(s', a')$. Тогда мы можем просто ещё в нашу оценку V-функции добавить слагаемое:

$$V(s) \leftarrow V(s) + \alpha\gamma\Psi_{(1)}(s', a')$$

Непосредственной проверкой легко убедиться, что суммарное обновление V-функции получится эквивалентным двухшаговому обновлению:

Утверждение: Последовательное применение обновлений эквивалентно двухшаговому обновлению

$$V(s) \leftarrow V(s) + \alpha\Psi_{(2)}(s, a)$$

Понятно, что можно обобщить эту идею с двухшаговых ошибок на N-шаговые: действительно, ошибка за N шагов равна сумме одношаговых ошибок за эти шаги.

Теорема:

$$\Psi_{(N)}(s, a) = \sum_{t=0}^{N-1} \gamma^t \Psi_{(1)}(s^{(t)}, a^{(t)})$$

Это наблюдение открывает, что все наши формулы обновления выражаются через одношаговые ошибки — $\Psi_{(1)}(s, a)$. Это интересный факт, поскольку одношаговая временная разность

$$\Psi_{(1)}(s, a) = r + \gamma V(s') - V(s)$$

очень похожа на reward shaping (формирование вознаграждения), где в качестве потенциала выбрана наша текущая аппроксимация $V(s)$. Поэтому эти дельты, как их ещё иногда называют, можно интерпретировать как некие «новые награды», центрированные — которые в среднем должны быть равны нулю, если аппроксимация V-функции точная.

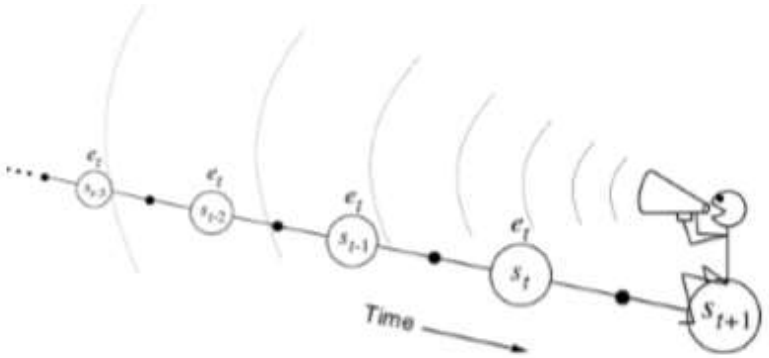

Рис. 5.2

Итого, оказывается, мы можем на каждом шаге добавлять к оценкам V-функции ранее встречавшихся в эпизоде пар s, a только что случившуюся одношаговую ошибку $\Psi_{(1)}(s, a)$ и таким образом получать N-шаговые обновления: достаточно пару s, a, посещённую K шагов назад, обновить с весом γ^K. То есть мы начинаем действовать по-другому: зная, что было в прошлом, мы правильным образом обновляем оценочную функцию посещённых состояний из

прошлого, используя временную разность за один последний шаг («настоящее»). Такой подход к обновлению оценочной функции называется, соответственно, «backward view», и он позволяет взглянуть на обучение оценочных функций под другим углом (вид назад).

Eligibility Trace (Отслеживание приемлемости)

Допустим, мы сделали шаг в среде и получили на этом одном шаге какую-то одношаговую ошибку $\Psi_{(1)}$. Рассмотрим какое-нибудь состояние s. С каким весом, помимо learning rate (скорости обучения), нужно добавить эту ошибку к нашей текущей аппроксимации? Это состояние в течение прошлого эпизода было, возможно, посещено несколько раз, и за каждое посещение вес увеличивается на γ^{K}, где K — число шагов с момента посещения. Такой счётчик можно сохранить в памяти: заведём вектор $e(s)$ размером с число состояний, проинициализируем его нулём и далее на t-ом шаге будем обновлять следующим образом:

$$e_{\mathrm{t}}(s) := \begin{cases} \gamma \mathrm{e}_{\mathrm{t}-1}(\mathrm{s}) + 1 & \text{если } \mathrm{s} = \mathrm{s}_{\mathrm{t}} \\ \gamma \mathrm{e}_{\mathrm{t}-1}(\mathrm{s}) & \text{иначе} \end{cases}$$

После этого на каждом шаге мы будем добавлять текущую одношаговую ошибку, временную разность $\Psi_{(1)}(s_t, a_t) = r_t + \gamma V(s_{t+1}) - V(s_t)$, ко всем состояниям с коэффициентом $e_t(s)$.

Определение 61: Будем называть $e_t(s)$ следом (eligibility trace) для состояния s в момент времени t эпизода коэффициент, с которым алгоритм обновляет оценочную функцию $V(s)$ на t-ом шаге при помощи текущей одношаговой ошибки $\Psi_{(1)}(s_t, a_t)$:

$$V(s) \leftarrow V(s) + \alpha e_t(s) \Psi_{(1)}(\mathrm{s_t}, \mathrm{a_t})$$

Одношаговое обновление будет превращено в двухшаговое, двухшаговое — в трёхшаговое и так далее до N-шагового, где N — количество шагов до конца эпизода. Таким образом (если в ходе таких обновлений learning rate и оцениваемая стратегия не меняется) наши обновления в точности соответствуют Монте-Карло.

Важно, что eligibility trace (отслеживание приемлемости) имеет физический смысл «кредита», который выдан решениям, принятым в состоянии s: это та степень ответственности, с которой выбранное в этом состоянии действия влияют на события настоящего, на получаемые сейчас награды (временные разности).

TD(λ) (temporal difference - временная разница)

После момента посещения состояния s след для него вырастет на единичку

и оценочная функция обновится следующим образом:

$$V(s) \leftarrow V(s) + \alpha\Psi_{(1)}(\mathrm{s},\mathrm{a})$$

Допустим, на следующем шаге мы выбрали $\lambda_1 \in [0,1]$ — «коэффициент затухания» — и потушили след не с коэффициентом γ, а с коэффициентом $\gamma\lambda_1$. Тогда дальше мы добавим новую текущую временную разность $\Psi_{(1)}(s', a')$ с коэффициентом $\lambda\gamma$, получая суммарно следующее обновление:

$$V(s) \leftarrow V(s) + \alpha\big(\alpha\Psi_{(1)}(\mathrm{s},\mathrm{a}) + \lambda_1\gamma\Psi_{(1)}(s', a')\big)$$

которое в силу для $N = 2$ преобразуется в:

$$V(s) \leftarrow V(s) + \alpha\Big((1-\lambda_1)\Psi_{(1)}(\mathrm{s},\mathrm{a}) + \lambda_1\Psi_{(2)}(\mathrm{s},\mathrm{a})\Big)$$

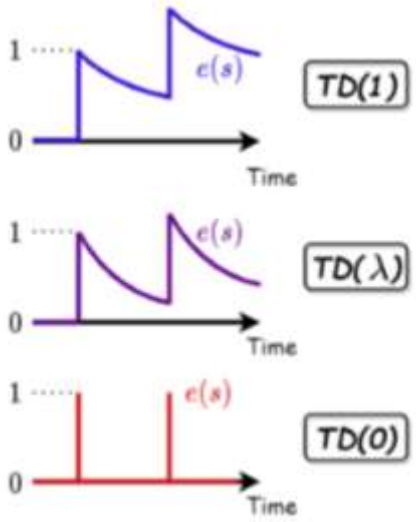

Рис. 5.3

Таким образом, мы заансамблировали одношаговое и двухшаговое обновление. Из этого видно, что такая процедура корректна: V^{π} является неподвижной точкой как одношагового, так и двухшагового уравнения Беллмана, и значит неподвижной точкой любой их выпуклой комбинации.

С точки зрения кредита, мы сказали, что решение, принятое в s, влияет на то, что случится через 2 шага, но не так сильно, как на то, что случится через 1 шаг: степень ответственности за один шаг затухла в λ_1 раз. Решения более вероятно влияют на ближайшее будущее, нежели чем на далёкое.

Определение: Будем называть (временная разница) temporal difference обновлением с параметром $\lambda \in [0,1]$ («обновление TD(λ)) обновление со следом $et(s)$, проинициализированным нулём и далее определённым следующим образом:

$$e_{\mathrm{t}}(s) := \begin{cases} \gamma \mathrm{e}_{\mathrm{t}-1}(\mathrm{s}) + 1 & \text{если } \mathrm{s} = \mathrm{s}_{\mathrm{t}} \\ \gamma \mathrm{e}_{\mathrm{t}-1}(\mathrm{s}) & \text{иначе} \end{cases}$$

Формула следа задаёт алгоритм в парадигме backward view (вид назад). Естественно, что любая оценка, придуманная в терминах backward view (вид

назад), то есть записанная в терминах следа (в явном виде хранящего «кредит» ответственности решений для каждого состояния), переделывается в парадигме forward view (вид вперед как и наоборот), когда мы, используя сэмплы будущего, строим некоторую оценку Advantage (преимущество) $\Psi(s,a) \approx A^{\pi}(s,a)$ и просто сдвигаем по нему значение V-функции:

$$V(s) \leftarrow V(s) + \alpha\Psi(s,a)$$

Теорема — Эквивалентные формы TD(λ): Обновление TD(λ) эквивалентно с оценкой

$$\Psi(s,a) := \sum_{t\geq 0} \gamma^t \lambda^{\lambda} \Psi_{(1)}(s^{(t)}, a^{(t)}) = (1-\lambda) \sum_{t\geq 0} \lambda^{t-1} \Psi_{(t)}(s,a)$$

Итак, полученная формула обновления имеет две интерпретации. Получается, что подобный ансамбль многошаговых оценок эквивалентен дисконтированной сумме будущих одношаговых ошибок («модифицированных наград» с потенциалом $V(s)$), где коэффициент дисконтирования равен $\gamma\lambda$; исходный коэффициент γ отвечает за затухание ценности наград со временем из исходной постановки задачи, а λ соответствует затуханию кредита ответственности действия за полученные в будущем награды.

А с другой стороны можно интерпретировать TD(λ) как ансамбль многошаговых оценок разной длины. Мы взяли одношаговую оценку с весом λ, двухшаговую с весом λ^2, N-шаговую с весом λ^{N} и так далее. Сумма весов в ансамбле, как водится, должна равняться единице, отсюда в формуле домножение на $1-\lambda$.

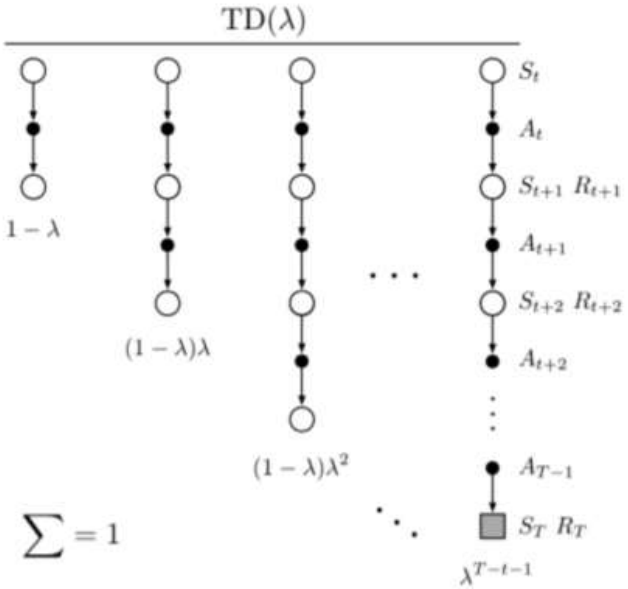

Рис. 5.4

Если через N шагов после оцениваемого состояния s эпизод закончился, то все многошаговые оценки длины больше N, понятно, совпадают с N-шаговой и

равны reward-to-go (награда на вынос). Следовательно, в такой ситуации reward-to-go по определению замешивается в оценку с весом $(1-\lambda)(\lambda^{N-1}+\lambda^{N}+\dots)=\lambda^{N-1}$.

Мы привыкли, что любая формула обновления для нас — стохастическая аппроксимация решения какого-то уравнения. TD(λ) не исключение. Если N-шаговая оценка направляет нас в сторону решения N-шагового уравнения Беллмана $V^{\pi}=\mathfrak{B}^{N}V^{\pi}$, то ансамбль из оценок направляет нас в сторону решения ансамбля N-шаговых уравнений Беллмана:

$$V^{\pi}=(1-\lambda)(\mathfrak{B}V^{\pi}+\lambda\mathfrak{B}^{2}V^{\pi}+\lambda^{2}\mathfrak{B}^{3}V^{\pi}+\dots)=(1-\lambda)\sum_{N>0}\lambda^{N-1}\mathfrak{B}^{N}V^{\pi}$$

Поскольку V^{π} является неподвижной точкой для операторов $\mathfrak{B}^{N}$ для всех N, то и для их выпуклой комбинации, «ансамбля», он тоже будет неподвижной точкой. Итого TD(λ) дало нам прикольную идею: мы не могли выбрать одну из многошаговых оценок, и поэтому взяли их все сразу.

Итак, мы получили алгоритм TD(λ) оценивания стратегии, или temporal difference (временная разница) с параметром λ, который при $\lambda=1$ эквивалентен Монте-Карло алгоритму (с постоянным обновлением V-функции «по ходу эпизода»), а при $\lambda=0$ эквивалентен одношаговому temporal difference методу, который мы, в частности, применяли в Q-learning и SARSA для оценивания стратегии.

Алгоритм: TD(λ)

Вход: π — стратегия
Гиперпараметры: α — параметр экспоненциального сглаживания, $\lambda\in[0,1]$ — степень затухания следа

Инициализируем $V(s)$ произвольно для всех $s\in\mathcal{S}$
Инициализируем $e(s)$ нулём для всех $s\in\mathcal{S}$
Наблюдаем s_0
На k-ом шаге:

1. выбираем $a_k\sim\pi(a_k\mid s_k)$
2. играем a_k и наблюдаем r_k, s_{k+1}
3. обновляем след $e(s_k)\leftarrow e(s_k)+1$
4. считаем одношаговую ошибку $\Psi_{(1)}:=r_k+\gamma V(s_{k+1})-V(s_k)$
5. для всех s обновляем $V(s)\leftarrow V(s)+\alpha e(s)\Psi_{(1)}$
6. для всех s обновляем $e(s)\leftarrow\gamma\lambda e(s)$

Выход: $V(s)$

Retrace(λ) (восстановить)

Возможность в принципе обучаться off-policy (вне политики) в Q-learning обеспечивалась тем, что, когда мы учим Q функцию с одношаговых оценок, нам для любых s, a, которые можно взять из буфера, достаточно лишь сэмпла s'. При

этом сэмпл $a' \sim \pi(a'|s')$ мы всегда сможем сгенерировать «онлайн», прогнав на взятом из буфера s' оцениваемую стратегию. Обычно в off-policy режиме учат именно Q-функцию, что важно в том числе тем, что по крайней мере одношаговая оценка будет доступна всегда. Если же a' из буфера такого, что $\pi(a'|s') = 0$, то любые наши коррекции схлопнутся в одношаговую оценку (или же будут теоретически некорректны), но по крайней мере хоть что-то мы сможем сделать.

Итак, попробуем разрешить (компромисс между смещением и дисперсией) bias-variance trade-off-а для Q-функции

Теорема — Retrace: Пусть число состояний и действий конечно, таблица $Q_0(s,a)$ проинициализирована произвольно. Пусть на k-ом шаге алгоритма для каждой пары s, a ячейка таблицы обновляется по формуле

$$Q_{\mathrm{k}+1}(s,a) \leftarrow Q_{\mathrm{k}}(s,a) + \alpha_{\mathrm{k}}(s,a) \sum_{t \geq 0} \gamma^t \left(\prod_{i=1}^{i=t} c_i \right) \Psi_{(1)}(s^{(t)}, a^{(t)})$$

$$\Psi_{(1)}\left(s^{(t)}, a^{(t)}\right) = r^{(t)} + \mathbb{E}_{a_\pi^{(t+1)} \sim \pi_k} Q_k(s^{(t+1)}, a_\pi^{(t+1)}) - Q(s^{(t)}, a^{(t)})$$

где $\mathcal{T} \sim \mu_k | s_0 = s, a_0 = a$ сгенерирована произвольной стратегией сбора данных μ_k (причём не обязательно стационарной), learning rate (скорость обучения) $\alpha_k(s,a)$ — случайно, π_k произвольно, и коэффициенты следа — любые в диапазоне

$$c_i \in \left[0, \frac{\pi_k(a^{(i)}|s^{(i)})}{\mu_k(a^{(i)}|s^{(i)})}\right]$$

Тогда, если с вероятностью 1 скорость обучения удовлетворяет условиям Роббинса-Монро, а стратегия π_k с вероятностью 1 становится жадной по отношению к Q_k в пределе $k \to \infty$, то при некоторых технических ограничениях с вероятностью 1 Q_k сходится к оптимальной Q-функции Q^*, а π_{k}, соответственно, к оптимальной стратегии.

Пользуясь этой теоремой, мы можем в полной аналогии с TD(λ) выбрать гиперпараметр $\lambda \in [0,1]$, и считать след по формуле

$$c_i = \lambda \frac{\pi(a^{(i)}|s^{(i)})}{\mu(a^{(i)}|s^{(i)})}$$