

16. ЛЕКЦИЯ. Policy gradient подход (Подход градиентной политики) 2

Схемы «актёр-критик»

Введение критика

Мы хотели научиться оптимизировать параметры стратегии при помощи формулы градиента, не доигрывая эпизоды до конца. Мы уже поняли, что мат.ожидание по траекториям не представляет для нас проблемы. Тогда осталось лишь придумать, как, не доигрывая эпизоды до конца, проводить credit assignment (кредитное присвоение), то есть определять для каждой пары s, a оценку Advantage (Преимущество)-функции.

Раз знание функции Q^π позволит обучаться, не доигрывая эпизоды до конца, а функцию в готовом виде нам никто не даст, то возникает логичная идея — аппроксимировать её. Итак, введём вторую сетку, которая будет «оценивать» наши собственные решения — критика (critic). Нейросеть, моделирующую стратегию, соответственно будем называть актёром (actor), и такие алгоритмы, в которых обучается как модель критика, так и модель актёра, называются Actor-Critic (актёр-критик).

Здесь возникает принципиальный момент: мы не умеем обучать модели оценочных функций Q^π или V^π , так чтобы они оценивали будущую награду несмещённо («выдавали в среднем правильный ответ»). Что это влечёт? При смещённой оценке Q -функции любые гарантии на несмещённость градиентов мгновенно теряются. Единственный способ оценивать Q -функцию несмещённо — Монте-Карло, но она требует полных эпизодов и имеет более высокую дисперсию. Любое замешивание нейросетевой аппроксимации в оценку — и мы сразу теряем несмещённость оценок на градиент, и вместе с ними — любые гарантии на сходимость стохастической оптимизации к локальному оптимуму или даже вообще хоть куда-нибудь.

В машинном обучении в любых задачах оптимизации всегда необходимо оценивать градиент оптимизируемого функционала несмещённо, и важной необычной особенностью Policy Gradient (градиент политики) методов в RL (обучение с подкреплением) является тот факт, что в них внезапно используются именно смещённые оценки градиента. Надежда на работоспособность алгоритма после замены значения $Q^\pi(s, a)$ на смещённую оценку связана с тем, что формула градиента говорит проводить policy improvement (улучшение политики) во встречаемых состояниях. Коли градиент есть просто policy improvement, то мы знаем из общей идеи алгоритма Generalized Policy Iteration (итерация обобщенной политики), что для улучшения политики вовсе не обязательно

использовать идеальную Q^π , достаточно любой аппроксимации критика $Q(s, a) \approx Q^\pi(s, a)$, которая параллельным процессом движется в сторону идеальной Q^π для текущей стратегии π . Фактически, все Actor-Critic (актёр-критик) методы моделируют именно эту идею.

Следующий существенный момент заключается в том, что в качестве критика обычно учат именно V -функцию. Во-первых, если бы мы учили модель Q -функции и подставили бы её, то градиент

$$\nabla_{\theta} \log \pi_{\theta}(a|s) Q(s, a)$$

направил бы нашу стратегию в $\arg\max_a Q(s, a)$. Помимо того, что это выродило бы стратегию, это бы привело к тому, что качество обучения актёра выродилось бы в качество обучения критика: пока модель $Q(s, a)$ не похожа на истинную $Q^\pi(s, a)$, у нас нет надежды, что актёр выучит хорошую стратегию.

Возможность не обучать сложную Q^* является одним из преимуществ подхода прямой оптимизации $J(\theta)$. DQN было обязательным учить именно Q -функцию, так как, во-первых, мы хотели выводить из неё оптимальную стратегию, во-вторых, для уравнения оптимальности Беллмана для оптимальной V -функции фокус с регрессией не прокатил бы — там мат.ожидание стоит внутри оператора взятия максимума. Сейчас же мы из соображений эффективности алгоритма (желания не играть полные эпизоды и задачи снижения дисперсии оценок) не сможем обойтись совсем без обучения каких-либо оценочных функций, но нам хватит лишь $V_{\phi}(s) \approx V^\pi$, поскольку оценить Q -функцию мы можем хотя бы так:

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s'} V^\pi(s') \approx r(s, a) + \gamma V_{\phi}(s'), \quad s' \sim p(s' | s, a) \quad (1)$$

Иначе говоря, если у нас есть приближение V -функции, то мы можем использовать её как для бэйзлайна, так и для оценки Q -функции. Важно, что V -функция намного проще, чем Q -функция: ей не нужно дифференцировать между действиями, достаточно лишь понимать, какие области пространства состояний — хорошие, а какие плохие. И поэтому раз можно обойтись ей, то почему бы так не сделать и не упростить критику задачу/

Bias-variance trade-of (торговля со смещённой дисперсией)

Обсудим сначала, как критик $V_{\phi}(s) \approx V^\pi(s)$ с параметрами ϕ будет использоваться в формуле градиента по параметрам стратегии

$$\nabla_{\theta} J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{d_{\pi}(s)} \mathbb{E}_{\pi(a|s)} \nabla_{\theta} \log \pi_{\theta}(a | s) A^{\pi}(s, a) \quad (2)$$

Мы собираемся вместо частного advantage (преимущества) подставить некоторую его оценку (advantage estimator - оценщик преимуществ) и провести

таким образом credit assignment (кредитное присвоение):

$$\nabla_{\theta} J(\pi) \approx \frac{1}{1-\gamma} \mathbb{E}_{d_{\pi}(s)} \mathbb{E}_{a \sim \pi(a|s)} \nabla_{\theta} \log \pi_{\theta}(a | s) \underbrace{\Psi(s, a)}_{\approx A^{\pi}(s, a)}$$

Теперь рассмотрим алгоритм bias-variance trade-of (торговля со смещённой дисперсией), в контексте policy gradient (градиентной политики), речь напрямую идёт о дисперсии и смещении оценок градиента. Мы уже встречались с двумя самыми «крайними» вариантами выбора функции $\Psi(s, a)$: Монте-Карло и одношаговая оценка через V -функцию (2):

$\Psi(s, a)$	Дисперсия	Смещение
$R_t - V_{\varphi}(s)$	высокая	нет
$r(s, a) + \gamma V_{\varphi}(s') - V_{\varphi}(s)$	низкая	большое

В этой таблице речь идёт именно о дисперсии и смещении оценки градиента $J(\pi)$ при использованной аппроксимации. То есть, в первом случае оценка Q -функции несмещённая, оценка V -функции смещённая, но поскольку в качестве бэйзлайна в силу

$$\mathbb{E}_{a \sim \pi_{\theta}(a)} \nabla_{\theta} \log \pi_{\theta}(a) = 0$$

может использоваться совершенно произвольная функция от состояний, совершенно несущественно, насколько наша аппроксимация $V^{\pi}(s)$ вообще похожа на истинную оценочную функцию. Да, если аппроксимация V -функции будет неточной, мы, вероятно, не так сильно собьём дисперсию оценок градиента, как могли бы, но ровно на этом недостатки использования смещённой аппроксимации V -функции в качестве бэйзлайна заканчиваются.

Во втором же случае, аппроксимация $V^{\pi}(s')$ используется для оценки Q -функции и вызывает смещение в том числе в оценке градиента. Дисперсия же снижается, поскольку в Q -функции аккумулированы мат.ожидания по всему хвосту траектории; она в обоих случаях существенно ниже, чем до введения бэйзлайна, но замена Монте-Карло оценки на бутстрапированную оценку снижает её ещё сильнее.

Вспомним, как можно интерполировать между этими крайними вариантами. Для начала, есть ещё целое семейство промежуточных вариантов — многошаговых (multi-step) оценок Q -функции, использующих N -шаговое уравнение Беллмана:

$$Q^{\pi}(s, a) \approx \sum_{t=0}^{N-1} \gamma^t r^{(t)} + \gamma^N V_{\varphi}(s^{(N)})$$

Тогда мы пользуемся для credit assingment (кредитное присвоение) N -

шаговой оценкой Advantage (преимущество), или N-шаговой временной разностью.

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

С ростом N дисперсия такой оценки увеличивается: всё больший фрагмент траектории мы оцениваем по Монте-Карло, нам становятся нужны сэмплы $a_{t+1} \sim \pi(a_{t+1}|s_{t+1}), s_{t+2} \sim \pi(s_{t+2}|s_{t+1}, a_{t+1}), \dots, s_{t+N} \sim \pi(s_{t+N}|s_{t+N-1}, a_{t+N-1})$, а при $N \rightarrow \infty$ оценка в пределе переходит в полную Монте-Карло оценку оставшейся награды, где дисперсия большая, но зато исчезает смещение в силу отсутствия смещённой аппроксимации награды за хвост траектории.

Также с ростом N мы начинаем всё меньше опираться на модель критика. Используя сэмплы наград, мы ценой увеличения дисперсии напрямую учим связь между хорошими действиями и высоким откликом от среды, меньше опираясь на промежуточный этап в виде выучивания оценочной функции. Это значит, что нам не нужно будет дожидаться, пока наш критик идеально обучиться: в оценку Advantage (преимущества) попадает сигнал в том числе из далёкого будущего при больших N , и актёр поймёт, что удачно совершённое действие надо совершать чаще. Математически это можно объяснить тем, что, увеличивая N , мы снижаем смещение наших оценок градиента. Значит, нам в целом даже и не потребуется, чтобы критик оценивал состояния с высокой точностью, поскольку главное, чтобы в итоге получалась более-менее адекватная оценка совершённых нашей стратегией действий.

Trade-off (компромисс) заключается в том, что чем дальше в будущее мы заглядываем, тем выше дисперсия этих оценок; помимо этого, для заглядывания в будущее на N шагов нужно же иметь это самое будущее, то есть из каждой параллельно запущенной среды понадобится собрать для очередного мини-батча не по одному сэмплу, а собрать целый длинный фрагмент траектории. Поэтому, регулируя длину оценок, мы «размениваем» смещение на дисперсию, и истина, как всегда, где-то посередине.

Возможность разрешать bias-variance trade-off и выбирать какую-то оценку с промежуточным смещением и дисперсией — чуть ли не главное преимущество on-policy (с политикой) режима обучения. Напомним, что сэмплы фрагментов траекторий из буфера получить не удастся (действия должны генерироваться из текущей стратегии), и использование многошаговых оценок в off-policy (без политики) режиме было невозможно.

Пока что в нашем алгоритме роллауты (развертывание) непрактично

делать сильно длинными. Дело в том, что пары s, a из длинных роллаутов будут сильно скоррелированы, что потребует перебивать числом параллельно запущенных сред, а тогда при увеличении длины роллаута начинает раздуваться размер мини-батча. Потом собранные переходы будут использованы всего для одного шага градиентного подъёма, и переиспользовать их будет нельзя; это расточительно и поэтому большой размер мини-батча невыгоден. Поэтому пока можно условно сказать, что самая «выгодная» оценка Advantage (преимущества) для не очень длинных роллаутов — оценка максимальной длины: для s_t мы можем построить N -шаговую оценку, её и возьмём; для s_{t+1} уже не более чем $N - 1$ -шаговую; наконец, для s_{t+N-1} нам доступна лишь одношаговая оценка, просто потому что никаких сэмплов после s_{t+N} мы ещё не получали.

Определение 1: Для пар s_t, a_t из роллаута $s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_N$ длины N будем называть оценкой максимальной длины (max trace estimation) оценку с максимальным заглядыванием в будущее: для Q -функции

$$y^{MaxTrace}(s_t, a_t) := \sum_{\hat{t}=t}^{N-1} \gamma^{\hat{t}-t} r_{\hat{t}} + \gamma^{N-t} V^{\pi}(s_N) \quad (3)$$

для Advantage (преимущества) функции, соответственно:

$$\psi^{MaxTrace}(s_t, a_t) := y^{MaxTrace}(s_t, a_t) - V^{\pi}(s_t)$$

Заметим, что мы вовсе не обязаны использовать для всех пар s, a оценку одной и той же длины N . То есть мы не должны брать для s_t, a_t из N -шагового роллаута N -шаговую оценку, а остальные пары s, a из роллаута не использовать в обучении лишь потому, что для них N -шаговая оценка невозможна; вместо этого для них следует просто использовать ту многошаговую оценку, которая доступна. Это полностью корректно, поскольку любая N -шаговая оценка является оценкой Advantage (преимущества) для данного слагаемого в нашей формуле градиента. Именно это и говорит оценка максимальной длины: если мы собрали роллаут $s_0, a_0, s_1, a_1, \dots, s_5$ длины 5, то одну пару s_4, a_4 , самую последнюю, нам придётся оценить одношаговой оценкой (поскольку для неё известен сэмпл лишь следующего состояния s_5 и только, никаких альтернатив здесь придумать не получится), предпоследнюю пару s_3, a_3 — двухшаговой, и так далее. То есть «в среднем» длина оценок будет очень маленькой, такая оценка с точки зрения bias-variance (дисперсия смещения) скорее смещена, чем имеет большую дисперсию.

Generalized Advantage Estimation — GAE (Обобщенная оценка преимуществ)

Пока N не так велико, чтобы дисперсия раздувалась, оценка максимальной длины — самое разумное решение компромисс (trade-off), и о более умном решении думать не нужно. Однако, впоследствии мы столкнёмся с ситуацией,

что с политикой (on-policy) алгоритмы будут собирать достаточно длинные роллауты (порядка тысячи шагов), и тогда брать оценку наибольшей длины уже будет неразумно; с этой же проблемой можно столкнуться и в контексте обсуждаемой актёр-критик (Actor-Critic) схемы, если длина собираемых роллаутов достаточно большая.

Решение дилеммы компромисс между смещением и дисперсией (bias-variance trade-off)) подсказывает теория обновления временной разницы TD(λ) оценки. Нужно применить формулу

$$\Psi(s, a) := \sum_{t \geq 0} \gamma^t \lambda^t \Psi_{(1)}(s^{(t)}, a^{(t)}) = (1 - \lambda) \sum_{t \geq 0} \lambda^{t-1} \Psi_{(t)}(s, a) \quad (4)$$

и просто заансамблировать N -шаговые оценки разной длины:

Определение: GAE-оценкой Advantage-преимущество-функции называется ансамбль многошаговых оценок, где оценка длины N

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

берётся с весом λ^{N-1} , где $\lambda \in (0, 1)$ – гиперпараметр:

$$\Psi_{\text{GAE}}(s, a) := (1 - \lambda) \sum_{N > 0} \lambda^{N-1} \Psi_{(N)}(s, a)$$

При $\lambda \rightarrow 0$ такая GAE-оценка соответствует одношаговой оценке; при $\lambda = 1$ GAE-оценка соответствует Монте-Карло оценке Q-функции.

В текущем виде в формуле суммируются все N -шаговые оценки вплоть до конца эпизода. В реальности собранные роллауты могут прерваться в середине эпизода: допустим, для данной пары s, a через M шагов роллаут «обрывается». Тогда на практике используется чуть-чуть другим определением GAE-оценки: если мы знаем $s^{(M)}$, но после этого эпизод ещё не доигран до конца, мы пользуемся формулой (3) и оставляем от суммы только «доступные» слагаемые:

$$\Psi_{\text{GAE}}(s, a) := \sum_{t \geq 0}^{M-1} \gamma^t \lambda^t \Psi_{(1)}(s^t, a^t) \quad (5)$$

Напомним, что это корректно, поскольку соответствует просто занулению следа на M -ом шаге, или, что тоже самое, ансамблированию (взятию выпуклой комбинации) первых M многошаговых оценок, где веса «пропавших» слишком длинных оценок просто перекадываются в вес самой длинной доступной M -шаговой оценки:

Утверждение: Формула (5) эквивалентна следующему ансамблю N -шаговых оценок:

$$\Psi_{\text{GAE}}(s, a) := (1 - \lambda) \sum_{N > 0} \lambda^{N-1} \Psi_{(N)}(s, a) + \lambda^{M-1} \Psi_{(N)}(s, a)$$

В такой «обрезанной» оценке $\lambda = 1$ соответствует оценке максимальной

длины, а $\lambda = 0$ всё ещё даст одношаговую оценку.

В коде формула (5) очень удобна для рекурсивного подсчёта оценки; также для практического алгоритма осталось учесть флаги $done_t$. Формулы подсчёта GAE-оценки для всех пар (s, a) из роллаута $s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_N$ приобретают такой вид:

$$\begin{aligned}\Psi_{\text{GAE}}(s_{N-1}, a_{N-1}) &:= \Psi_{(1)}(s_{N-1}, a_{N-1}) \\ \Psi_{\text{GAE}}(s_{N-2}, a_{N-2}) &:= \Psi_{(1)}(s_{N-2}, a_{N-2}) + \gamma\lambda(1 - done_{N-2})\Psi_{\text{GAE}}(s_{N-1}, a_{N-1}) \\ &\dots \\ \Psi_{\text{GAE}}(s_0, a_0) &:= \Psi_{(1)}(s_0, a_0) + \gamma\lambda(1 - done_0)\Psi_{\text{GAE}}(s_1, a_1)\end{aligned}$$

Заметим, что эти формулы очень похожи на расчёт кумулятивной награды за эпизод, где «наградой за шаг» выступает $\Psi_{(1)}(s, a)$. В среднем наши оценки Advantage (преимущество) должны быть равны нулю, и, если наша аппроксимация этому не удовлетворяет, мы получаем направление корректировки стратегии.

Обучение критика

Как обучать критика $V_\phi(s) \approx V^\pi$? Воспользуемся идеей перехода к регрессии, которую мы обсуждали раньше в контексте DQN (Deep Q-learning – иллубокое Q-обучение). Нам нужно просто решать методом простой итерации уравнение Беллмана:

$$V_{\phi_{k+1}}(s) \leftarrow \mathbb{E}_\alpha[r + \gamma \mathbb{E}_{s'} V_{\phi_k}(s')]$$

Мы можем получить несмещённую оценку правой части $y := r + \gamma V_{\phi_k}(s')$ и с таким таргетом минимизировать MSE. Однако, давайте воспользуемся преимуществами on-policy (с политикой) режима и поймём, что мы можем поступить точно также, как с оценкой Q-функции в формуле градиента: решать многошаговое уравнение Беллмана вместо одношагового. Например, можно выбрать любое N-шаговое уравнение и строить целевую переменную как

$$y := r + \gamma r' + \gamma^2 r'' + \dots + \gamma^N V_{\phi_k}(s^N) \quad (6)$$

Конечно же, определение того, насколько далеко в будущее заглядывать при построении таргета — это снова всё тот же самый bias-variance trade-off (компромисс между смещением и дисперсией) и очередное ключевое преимущество on-policy (с политикой) подхода — разрешать его в том числе при обучении критика.

В чём заключается компромисс между смещением и дисперсией при обучении критика? С ростом N таргет (6) в задаче регрессии становится всё более и более шумным, зато мы быстрее распространяем сигнал и меньше опираемся в таргете на свою же собственную аппроксимацию. Это позволяет бороться с проблемой накапливающейся ошибки, от которой страдают off-policy (без

политики) алгоритмы вроде DQN. В пределе — при $N = +\infty$ — мы решаем задачу регрессии, где целевая переменная есть reward-to-go (награда на вынос), и начинаем учить V -функцию просто по определению. Такая задача регрессии уже является самой обычной задачей регрессии из машинного обучения, целевая переменная будет являться именно теми значениями, среднее которых мы и хотим выучить. Но такая задача регрессии будет обладать очень шумными целевыми переменными, плюс для сбора таких данных понадобится, опять же, полные эпизоды играть.

Мы можем для оценки Q -функции для обучения политики и для построения целевой переменной для критика использовать разные подходы (скажем, оценки разной длины), но особого смысла в этом немного: хороший вариант для одного будет хорошим вариантом и для другого. Соответственно, можно считать решение trade-off (компромисс) одинаковым для актёра и критика. Тогда если мы оцениваем Advantage (преимущество) как

$$\Psi(s, a) = y - V_\phi(s),$$

где y — некоторая оценка Q -функции, то y же является и таргетом для V -функции, и наоборот. Используя функцию потерь MSE с таким таргетом, мы как раз и учим среднее значение наших оценок Q -функции, то есть бэйзлайн.

Конечно же, мы можем использовать и GAE-оценку (4) Advantage (преимущество), достаточно «убрать бэйзлайн»:

$$Q^\pi(s, a) = A^\pi(s, a) + V^\pi(s) \approx \Psi_{GAE}(s, a) + V_\phi(s)$$

При этом мы как бы будем решать «заансамблированные» N -шаговые уравнения Беллмана для V -функции:

Утверждение: Таргет $\Psi_{GAE}(s, a) + V_\phi(s)$ является несмещённой оценкой правой части «ансамбля» уравнений Беллмана:

$$V_\phi(s) = (1 - \lambda) \sum_{N \geq 0} \lambda^{N-1} [\mathfrak{B}^N V_\phi](s),$$

где \mathfrak{B} — оператор Беллмана для V -функции.

Брать для обучения критика набор выходных состояний s мы можем откуда угодно. Поэтому для удобства будем брать $s \sim \mu_\pi(s)$, чтобы можно было использовать для обучения критика и актёра один и тот же минибатч. Итого получаем следующее: делаем несколько шагов взаимодействия со средой, собирая таким образом роллаут некоторой длины N ; считаем для каждой пары s, a некоторую оценку Q -функции $y(s, a)$, например, оценку максимальной длины (3); оцениваем преимущество каждой пары как $\Psi(s, a) := y(s, a) - V_\phi(s)$; далее по Монте-Карло оцениваем градиент по параметрам стратегии

$$\nabla_{\theta} J(\pi) \approx \frac{1}{N} \sum_{s,a} \nabla_{\theta} \log \pi_{\theta}(a|s) \Psi(s, a)$$

и градиент для оптимизации критика (допустим, критик — Q -функция):

$$\text{Loss}^{\text{critic}}(\phi) = \frac{1}{N} \sum_{s,a} (y(s, a) - V_{\phi}(s))^2$$

Естественно, для декорреляции нужно собрать несколько независимых роллаутов из параллельно запущенных сред.

Advantage Actor-Critic (A2C – преимущество актёра-критика)

Мы собрали стандартную схему Advantage Actor Critic (A2C): алгоритма, который работает в чистом виде on-policy (с политикой) режиме. Из-за того, что роллауты в этом алгоритме не очень длинные, используются оценки максимальной длины (или, что тоже самое, GAE с $\lambda = 1$).

Алгоритм: Advantage Actor-Critic (A2C)

Гиперпараметры: M — количество параллельных сред, N — длина роллаутов, $V_{\phi}(s)$ — нейросеть с параметрами ϕ , $\pi_{\theta}(a | s)$ — нейросеть для стратегии с параметрами θ , α — коэф. масштабирования лосса критика, SGD оптимизатор.

Инициализировать θ, ϕ

На каждом шаге:

1. в каждой параллельной среде собрать роллаут длины N , используя стратегию π_{θ} :

$$s_0, a_0, r_0, s_1, \dots, s_N$$

2. для каждой пары s_t, a_t из каждого роллаута посчитать оценку Q -функции максимальной длины, игнорируя зависимость оценки от ϕ :

$$Q(s_t, a_t) := \sum_{i=t}^{N-1} \gamma^{i-t} r_i + \gamma^{N-t} V_{\phi}(s_N)$$

3. вычислить лосс критика:

$$\text{Loss}^{\text{critic}}(\phi) := \frac{1}{MN} \sum_{s_t, a_t} (Q(s_t, a_t) - V_{\phi}(s_t))^2$$

4. делаем шаг градиентного спуска по ϕ , используя $\nabla_{\phi} \text{Loss}^{\text{critic}}(\phi)$

5. вычислить градиент для актёра:

$$\nabla_{\theta}^{\text{actor}} := \frac{1}{MN} \sum_{s_t, a_t} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (Q(s_t, a_t) - V_{\phi}(s_t))$$

6. сделать шаг градиентного подъёма по θ , используя $\nabla_{\theta}^{\text{actor}}$

Сравним A2C с value-based (основанный на ценности) алгоритмами на основе DQN. Когда мы моделировали Value Iteration (итерации значения), мы целиком и полностью «полагались» на этап оценивания стратегии, то есть на обучение критика: модели актёра в явном виде даже не было в алгоритме, поскольку текущая стратегия всё время считалась жадной по отношению к

текущему критику. Это означало, что качество стратегии упиралось в качество критика: пока Q -функция не научится адекватно приближать истинную Q^* , стратегия хорошей не будет.

Достаточно интересно, что идея model-free (без модели) алгоритмов, отказывающихся обучать модель функции переходов в том числе из соображений end-to-end (концы с концами) схемы, пришла к тому, что мы иногда сводимся к обучению оценочных функций — промежуточных величин, вместо того, чтобы напрямую матчить состояния и действия, понимать, какие действия стоит выбирать чаще других. Вообще, интуиция подсказывает, что обучать актёра проще, чем оценочную функцию: в реальных задачах человек редко когда думает в терминах будущей награды.

Пример: Представьте, что вы стоите перед кофеваркой и у вас есть два действия: получить кофе и не получить. Вы прекрасно знаете, какое действие лучше другого, но при этом не оцениваете, сколько кофе вы сможете выпить в будущем при условии, например, что сейчас вы выберете первый или второй вариант: то есть не строите в явном виде прогноз значения оценочной функции.

И формула градиента, идея policy gradient (политика градиентов) методов, как раз предоставляет возможность обучать актёра напрямую, минуя промежуточный этап в виде критика: так, в алгоритме REINFORCE (УСИЛЕНИЕ) мы могли использовать Монте-Карло оценки Q -функции и обходиться без обучения в явном виде модели критика, то есть полностью опираться на этап policy improvement (улучшение политики). Поэтому policy gradient алгоритмы ещё иногда называют policy-based (основанный на политике).

Таким образом, DQN и REINFORCE — это два «крайних случая», первый алгоритм полностью опирается на критика, а второй алгоритм — на актёра. Только у первого недостатки компенсируются возможностью обучаться в off-policy (вне политики) режиме и использовать реплей буфер, а вот недостатки алгоритма REINFORCE (УСИЛЕНИЕ) — высокая дисперсия и необходимость играть целые эпизоды — не имеют аналогичного противовеса.

Важно, что в Policy Gradient алгоритмах мы за счёт использования метода REINFORCE при расчёте градиента можем заменять значение Q -функции (необходимую для улучшения политики) на какую-то заглядывающую в будущее оценку. И насколько сильно при обучении актёра опираться на критика («насколько далеко в будущее заглядывать») — это и есть bias-variance trade-off ((торговля со смещённой дисперсией), который в on-policy алгоритмах возможно разрешать. За счёт этого A2C в отличие от DQN может использовать и в качестве таргета для обучения критика, и в качестве оценки для обучения Q -функции

GAE-оценку (5). Да, пока что в A2C роллауты (зачастую) получаются слишком короткими, и GAE ансамбль «бедный», приходится $\lambda = 1$ использовать, но главное, что есть такая возможность технически, и даже короткие роллауты уже позволяют существенно справиться с проблемой распространения сигнала: это помогает и критику лучше учиться, и актёр может не полностью на критика опираться. В частности, нам не нужна модель Q -функции, достаточно более простой V -функции. Всё это делает Policy Gradient алгоритмы куда более эффективными в средах с сильно отложенным сигналом.

Наконец, ещё одно небольшое, но важное преимущество Policy Gradient — обучение стохастичной политики. Хотя мы знаем, что оптимальная политика детерминирована, обучение стохастичной политики позволяет использовать её для сбора данных, и стохастичность — ненулевая вероятность засэмплировать любое действие — частично решает проблему исследования. Да, это, конечно, далёкое от идеала решение, но намного лучшее, чем, например, ε -жадная стратегия: можно рассчитывать на то, что если градиент в какой-то области пространства состояний постоянно указывает на то, что одни действия хорошие, а другие плохие, актёр выучит с вероятностью, близкой к единице, выбирать хорошие действия; если же о действиях в каких-то состояниях приходит противоречивая информация, или же такие состояния являются для агента новыми, то можно надеяться, что стратегия будет близка к равномерной, и агент будет пробовать все действия.

Все эти преимущества неразрывно связаны с on-policy режимом. За счёт «свежести» данных, можно делать, так сказать, наилучшие возможные шаги обучения стратегия, практически идти по градиенту оптимизируемого функционала. Но из него проистекает и главный недостаток подхода: неэффективность по количеству затрачиваемых сэмплов. Нам всё время нужны роллауты, сгенерированные при помощи текущей политики с параметрами θ , чтобы посчитать оценку градиента в точке θ и сделать шаг оптимизации. После этого будут нужны уже сэмплы из новой стратегии, поэтому единственное, что мы можем сделать с уже собранными данными — почистить оперативную память.

Понятно, что это полное безобразие: допустим, агент долго вёл себя псевдослучайно и наконец наткнулся на какой-то хороший исход с существенным откликом среды. Схема сделает по ценному роллауту всего один градиентный шаг, что скорее всего не поможет модели выучить удачное поведение или значение отклика. После этого ценная информация никак не может быть использована и придётся дожидаться следующей удачи, что может

случиться нескоро. Это очень *sample inefficient* (образец неэффективен) и основная причина, почему от любого on-policy обучения в чистом виде нужно пытаться отойти.

Пример: Допустим, вы играете в видео-игру, и в начале обучения мало что умеете, всё время действуя примерно случайно и падая в первую же яму. Среда выдаёт вам всё время ноль, и вы продолжаете вести себя случайно. Вдруг в силу стохастичности стратегии вы перепрыгиваете первую яму и получаете монетку +1. В DQN этот ценнейший опыт будет сохранён в буфере, и постепенно критик выучит, какие действия привели к награде. В A2C же агент сделает один маленький шаг изменения весов моделей и тут же выкинет все собранные данные в мусорку, потому что на следующих итерациях он никак не может переиспользовать их. Агенту придётся ждать ещё много-много сессий в самой игре, пока он не перепрыгнет яму снова, чтобы сделать следующий шаг обучения перепрыгиванию ям.