TLab Test Task
Going from Offline to Online

Илья Зисман

10 апреля 2023 г.

1 Алгоритмы

Прежде всего я ознакомился со всеми статьями и их результатами. Мне приглянулись два алгоритма, которые я впоследствии и реализовал. Выбор пал на **AWAC** и улучшенный **TD3-BC-RF**. Этот выбор, прежде всего, обусловлен небольшим количеством деталей для успешной реализации. Например, статьи **IQL, CQL** имеют куда больше теоретических обоснований (формулировка и докатательство теорем и лемм для анализа алгоритма), нежели **TD3-BC-RF**, но зная по собственному опыту, чем больше у алгоритма составных частей, тем проще в нем налажать и тем менее стабильным он будет. А так как время у меня все же ограничено, как и вычислительные ресурсы (их просто нет :)), то и выбор был весьма очевиден.

Кроме того мне захотелось сравнить два принципиально разных подхода. Статья и алгоритм в **AWAC** кажутся более строго сформулированными и более "сложными" , тогда как **TD3-BC-RF** больше похож на костыль. Соответственно, эта идейная разница побудила во мне желание проверить, что же все-таки круче: глубокая математическая мысль или попытки инженеров собрать велосипед из уже имеющихся методов.

Еще стоит добавить, что **TD3-BC-RF** показалась мне самой адекватной из инжереных. Так, например, в статье *Adaptive Behavior Cloning Regularization for Stable Offline-to-Online Reinforcement Learning* предлагается заменить один гиперпараметр α на два новых гиперпараметра, которые отвечают за коэффициенты в PI-контроллере. Подход, конечно, интересный, но вряд ли его можно назвать практичным.

2 Датасет

Проверять перфоманс Offline-to-Online алгоритмов разумно на тех задачах, где можно сначала обучить наш алгоритм на суб-оптимальной политике и затем в онлайн режиме дообучить и сравнить стало ли лучше. Для этой цели очень хорошо подойдут среды из `MuJoCo` из пакета `D4RL`, где можно подгружать датасеты, которые обучены на разных уровнях. В частности, я решил взять данные из среды `halfcheetah-medium`, где перфоманс составляет $\frac{1}{3}$ от производительности оптимального агента. Использовать среды по типу `AntMaze` смысла здесь особого не имеет, так как в этих задачах нет предобученных суб-оптимальных политик, а датасет там генерится при помощи алгоритмов планирования маршрута, который оптимален.

Читу я взял из-за ее простоты и достаточно быстрой скорости обучения. К сожалению, в отсутствии ресурсов (колаб постоянно обрубал сессии, от кагла после 4-х часов отваливался `wandb`) мне не удалось запустить несколько сред для проверки, однако если бы я обладал k карточками A100, думаю, проблем прогнать остальныеы бенчмарки бы не возникло. Мне бы еще хотелось, например, попробовать оценить перфоманс агентов на `-random` датасетах, чтобы посмотреть так ли важны апостериорные данные для достижения высокого результата при онлайн тренировке.

	AWAC (Nair, Gupta)	AWAC ours	TD3-BC-RF (Beeson)	TD3-BC-RF ours
Offline	37.4	48.81	55.3 ±0.8	57.84 ±0.42
Online	52.9	74.58 ±1.55	74.4 ±2.4	80.6 ±1.55

Таблица 1: Результаты обучения агентов в оффлайн и онлайн режимах на датасете halfcheetah-medium. Указан нормализованная награда от 0 до 1. Первое число – среднее, второе – стандартное отклонение.

3 Результаты

Стоит сразу обратить внимание, что прогнать обучение на разных сидах у меня не вышло по тем же причинам, что и прогнать несколько бенчмарков. Так что можно считать полученные результаты первым приближением. Опять же, будь у меня k карточек A100... В общем, прогнать эксперименты на n сидах не сложнее, чем дописать `-seed=0`, `-seed=1`, ... к параметрам запуска скрипта. Оценка качества алгоритма проводилась каждые 1000 итераций на 10-ти разных сидах. В таблицах указано среднее значение +- стандартное отклонение (там, где я его сохранил) по этим 10-ти тестовым сидам. В статье с **AWAC** авторы не указывают стандартное отклонение для онлайн обучения, а просто приводят среднее.

Установку сида и еще часть утилит я честно скопировал из библиотеки **CORL**, чтобы не набагать самому. Весь остальной код, включая сам алгоритм, я реализовал сам. Результаты для среды нормализованные, то есть расположены в промежутке от 0 до 1, где 0 – random policy, 1 – expert policy, эти политики любезно посчитали за нас друзья из Berkeley.

В целом можно сказать, что алгоритмы показали приблизительно равные результаты (если верить в то, что в среднем по сидам оно так и учится). **TD3-BC-RF** показал качество на оффлай обучении лучше **AWAC**'а, но по картинке можно заметить, что без улучшения (то есть без **-RF**) результат у них равный. Вероятно, это достигается как раз через constraint relaxation, когда влияние BC лосса сокращается, в результате обученное распределение становится менее близким к π_β, но более генерализированным. В онлайн обучении **TD3-BC-RF** показал себя несколько лучше **AWAC**. Такое поведение можно попробовать объяснить тем, что **TD3-BC-RF** менее консервативен, тогда как имплицитная регуляризация в **AWAC** (KL-дивергенция) сохраняется на весь период обучения.

Еще стоит отметить, что **AWAC** показал куда более сильные результаты по сравнению с теми, которые зафиксированы в статье. Это не удивительно, учитывая, что в **CORL** уже был зафиксирован подобный перфоманс. Онлайн обучение так же вышло лучше, чем в оригинальной статье. **TD3-BC-RF** тоже слегка перешагнул уровень статьи, может удачный сид, может из-за немного иной реализации алгоритма. Я бы еще погонял, да времени уже нет. Графики представлены в конце .pdf. Обратите внимание, что TD3 и AWAC обучались разное количество эпох. Вышло немного криво для сравнения, но я хотел реализовать алгоритмы как можно ближе к статье, поэтому в offline графике появляется "AWAC Projected". Немножко навалил кринжа .

В реализации **AWAC** я пробовал не ограничивать advantage, а клипать градиент, однако это сильно замедляет обучение, поэтому остановился на варианте клипинга из **CORL**.

В реализации **TD3-BC-RF** я позволил себе в некоторых местах отклониться от реализации, предложенной в статье, но они настолько минорны, что вряд ли оказали какое-либо влияние на результат. Так, для апдейта политики я использую $\min_{i=1,2} Q_{\phi_i}$, а не Q_{ϕ_1}, хотя я и допускаю, что это просто опечатка в тексте статьи. Также я не стал реализовывать более частое обновление критика по отношению к актору. Еще в немногочисленных экспериментах было замечено, что несколько параллельных сред для сбора траекторий в буффер значительно ускоряют процесс схождения, поэтому в финальных версиях при сборе траекторий работают 8 сред.

Гиперпараметры для **TD3-BC-RF** были взяты из статьи, для **AWAC** из библиотеки **CORL**.

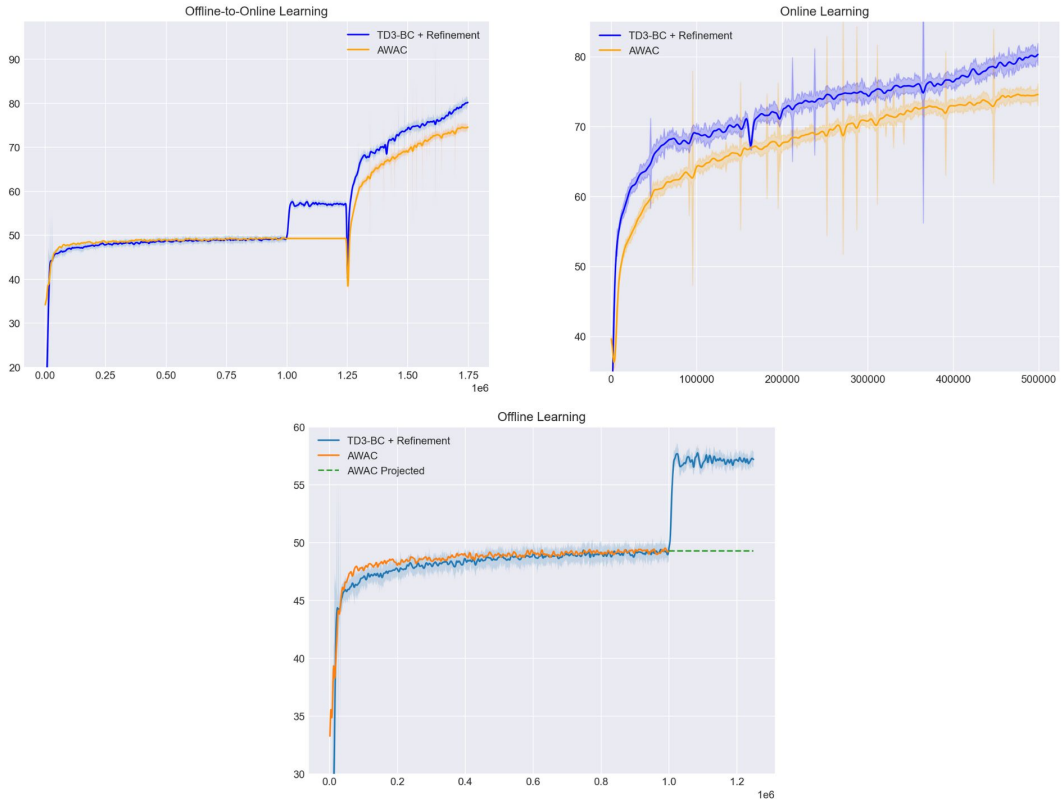

Рис. 1: Графики обучения