ideas & small research

Андрей

9 ноября 2025 г.

Содержание

1 Идеи

1.1 Исследование влияния ОДУ-солверов и траекторий в Flow Matching для онлайн RL

Идея: проверить, как выбор ОДУ-солвера и траектории в Conditional Flow Matching (CFM) влияет на сходимость и стабильность обучения flow-based агентов в онлайн RL[Lv+25]. Улучшение точности сэмплирования и выбор "хороших"траекторий (CFM) может увеличить устойчивость обучения RL-моделей без изменения их архитектуры[Lip+22]. Проблема лежит на стыке численных методов и RL, исследуя компромисс между точностью солвера и устойчивостью к шуму в градиентах. В области Flow-RL ещё не исследовалось влияние выбора солвера и траектории на эффективность обучения, хотя в других областях FM-моделей это давно стало фактором производительности.

1.2 QSS-chunking

Использовать в оригинальном QSS [Edw+20] или SAW [Lyu+22] метод аналогичный Q-chunking'у [LZL25], т.е. inverse dynamics model будет предсказывать chunk действий вместо одного действия. Интуитивно это должно улучшить идею QSS/SAW, так как мы сможем увеличить горизонт планирования (у нас будет $Q(s_t, s_{t+n})$). Это интересно, так как мы совмещаем два совершенно разных подхода в йерархическом RL. Насколько мне известно совместить эти две идеи никто не пытался. Также мне кажется интересным попробовать предсказывать вместо chunk'а эмбеддинг, который будет подаваться какой-нибудь рекурсивной нейронной модели. Ведь фиксированный chunk это странно (сам автор утверждает, что непонятно какой размер оптимален), и не масштабируемо (размер chunk'а растет $O(n)$ от количества действий).

1.3 Learnable dropout for continual learning (CL)

Идея использовать обучаемый dropout для CL (например, для задач Domain-Incremental Learning или Task-Incremental Learning) адаптивно перераспределяя регуляризацию по мере обучения новым задачам. В комбинации с парочкой других подходов (например, gradient projecting и какой-нибудь минимальный data replay [Wan+24]) это позволит нам сделать агента адаптивным к новым задачам за счёт регуляризации на низкую экспрессивность нейронов. Похожие методы были опробованы вне RL, интересно посмотреть, что RL агент наберет на бенчмарках для CL in RL. Сама по себе идея использовать разные виды dropout'ов в CL не нова (например, вот:[Abb+22]), но в контексте RL я не смог найти ничего похожего.

1.4 KAN for first layer

Вставить KAN (например SincKAN по совету [NWL25]) вместо первого линейного слоя Q. Нужно чтобы улучшить извлечение признаков из состояний, содержащих скрытую функциональную или периодическую структуру (например, в задачах с осцилляторной динамикой)., а тк при должном фич-инжиниринге KAN хорош в разложении функций, он сможет сходу выдавать нам высококачественные фичи. Интересно узнать, есть ли такая модификация KAN'а с качественно другим результатом по сравнению с MLP применительно к робототехнике. Хайп с KAN'ами продолжает идти на спад, но всё равно это остается по большей части неисследованная вселенная.

2 Эксперимент

2.1 Планирование

В настоящее время набирают популярность подходы к обучению RL, связанные с Flow Matching [Chi+25][Bla+], они всё чаще показывают SOTA-результаты. Политика в таких подходах может обучаться путем минимизации conditional flow matching objective (CFM)[Lip+22], а критики обучаются на лоссах, в которых действия сэмплированы политикой. В CFM нам важно правильно выбирать промежуточные значения действий между $a_0 \sim \mathcal{N}(0, I^2)$ и $a_1 \sim \mathcal{D}$ при заданном t (т.е. траекторию или путь), а при сэмплировании важен ОДУ-солвер. Как правило (насколько мне известно, если в RL, то везде), промежуточные значения выбирают линейно, а для сэмплинга (в RL тоже, насколько мне известно, всегда, в отличие от CV) используется базовый метод `midpoint`, в то время как в литературе по Flow Matching существуют более продвинутые методы численного интегрирования. `midpoint` — это компромисс между точностью и скоростью. Но что, если эта "экономия"на точности солвера на самом деле замедляет обучение, требуя от модели дольше адаптироваться к неточностям сэмплинга? Идея: исследовать влияние выбора ОДУ-солвера (ODE Solver) и траектории (path) на скорость обучения в задачах Online Reinforcement Learning.

Гипотеза 1: Использование более продвинутых и точных ОДУ-солверов (в нашем эксперименте мы пробуем метод Рунге-Кутты 4-го порядка [`rk4`] и неявный метод Адамса [`implicit_adams`]) приведет к более аккуратному сэмплингу действий. Это, в свою очередь, может ускорить сходимость (потребуется меньше шагов среды и/или суммарного времени) и повысить итоговую производительность политики.

Гипотеза 2: Мы также хотим сравнить два разных способа выбора траектории. Дефолтную линейную ($a_t = ta_1 + (1-t)a_0$) с траекторией с Preserving Linear Variance (PLV) ($a_t = ta_0 + \sqrt{1-t^2}a_1$). Выбор траектории, по которой элемент из шума «течет» к действию, может кардинально влиять на стабильность и скорость обучения CFM-модели.

Если гипотеза подтвердится, мы сможем предложить простую, но эффективную модификацию SOTA-метода, которая потенциально увеличит его sample efficiency. Мы хотим показать, что выбор численного метода для сэмплинга — это не второстепенная деталь реализации, а важный компонент, влияющий на производительность flow-based агентов.

Планирование эксперимента:

1. Подготовка: в качестве основы мы возьмём репозиторий FlowRL [Lv+25]. Мы заменим жестко прописанный step (`midpoint`) на гибкий интерфейс, совместимый с `ODESolver` из библиотеки flow_matching[Lip+24]. Это позволит нам легко переключать методы (`midpoint`, `rk4`, `implicit_adams` и т.д.), а также добавим из той же библиотеки разные траектории (возьмём линейную и с preserving linear variance). На самом деле это позволит нам не только переключаться между методами, которые подходят к нашему эксперименту, но и пробовать сэмплинг, например, на римановских многообразиях в случае, если новая задача будет того требовать.

   Мы сфокусируемся на трёх сложных задачах из DeepMind Control Suite: `dogrun`, `dogwalk`, `quadrupedwalk` (например, в `dogrun` $s \in \mathbb{R}^{233}$, а $a \in \mathbb{R}^{38}$).

2. Эксперименты: Запустить 12 прогонов (2 метода × 2 траектории × 3 задачи), каждый на ∼1M шагов, мониторя метрики сходимости (reward over steps). Использовать фиксированные сиды для воспроизводимости, гиперпараметры также фиксируем. Сравнить кривые обучения. Логи собираем в wandb.

3. Анализ: построить графики, вычислить средние/стандартные отклонения, проверить статистическую значимость (U-критерий Манна — Уитни). Оформить отчет с нарративом, графиками и выводами.

Что если изначальный план провалился? (например, `rk4` не дал прироста или оказался хуже `midpoint`). Возможно, более точный солвер `rk4` хуже работает с "шумным"градиентом или быстро меняющимся velocity field (самой моделью) во время обучения. `midpoint` может действовать как форма регуляризации, «сглаживая» процесс.

Убиваем или есть еще идеи?

Результат «простой `midpoint` работает лучше сложных солверов в задачах Flow RL» — это тоже не самый очевидный результат. Он бы сэкономил время будущим исследователям. Мы просто сместим фокус отчета с «Мы улучшили SOTA» на «Мы проанализировали и объяснили, почему SOTA-дизайн работает (и почему его наивная замена не улучшает его работу)».

2.2 Результаты

Я провел эксперимент с дефолтными для Flow RL настройками: солвер с методом `midpoint` и линейная траектория выбора промежуточных значений для CFM. Потом я запустил следующую сетку экспериментов: (`rk4`, `implicit_adams`) × (linear traj, traj with PLV) × (`dogrun`, `dogwalk`, `quadrupedwalk`). Код находится здесь.

Методы `rk4`, `implicit_adams` были выбраны как лучшие (и относительно быстрые) методы с фиксированным шагом. В противном случае, если бы мы выбрали, например, метод Дорманда-Принса с адаптивным шагом, мы бы уже не смогли использовать `@torch.compile`.

Траекторию с PLV мы взяли потому что она сохраняет линейную дисперсию (preserving linear variance), то есть обеспечивает более естественное сочетание шумовой и целевой составляющих траектории, что может улучшить стабильность и качество сэмплинга.

Задачи достаточно сложные и разнообразные, чтобы уменьшить влияние специфики отдельной среды и повысить обобщаемость результатов.

Мы получили графики reward/step, а также среднюю длительность одного шага обучения. (Приведены ниже)

Скорость тренировки с `implicit_adams` ($0.61 \pm 0.01 \frac{1000\text{steps}}{s}$) оказалась существенно медленнее, чем у `rk4` ($1.02 \pm 0.02 \frac{1000\text{steps}}{s}$) и `midpoint` ($1.29 \pm 0.01 \frac{1000\text{steps}}{s}$).

Вопреки ожиданиям, мы не нашли одного 'лучшего' метода. Оптимальный выбор солвера и траектории оказался сильно зависим от среды. Проведя U-тесты на AUC кривых train/reward(для $p < 0.05$), мы можем сказать, что для задачи `dogrun` лучшей оказалась комбинации (`rk4`+`PLV` и `implicit_adamsulinear`). Для `dogwalk` разные комбинации показывают примерно одинаковое кач-во. В `quadrupedwalk` выигрывает `midpoint` и `linear`.

Все эксперименты запускались на NVIDIA GeForce RTX 2080 Ti.

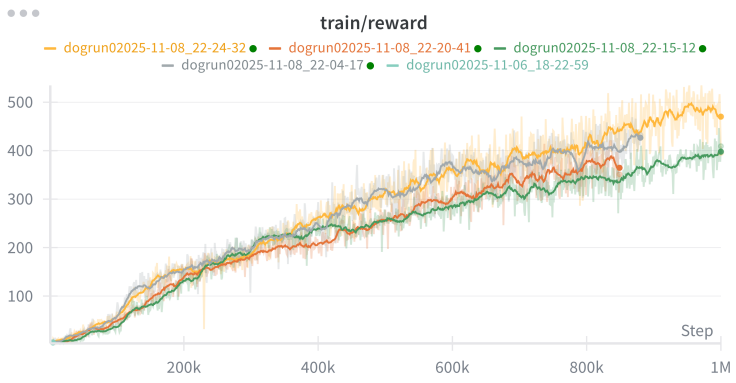

Рис. 1: Сравнение методов сэмплинга в среде DogRun

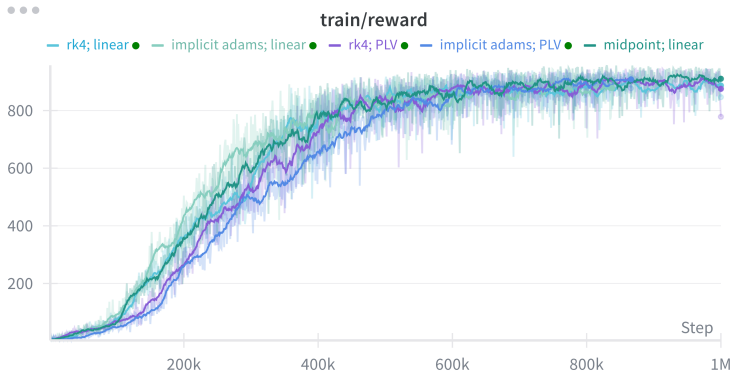

Рис. 2: Сравнение методов сэмплинга в среде DogWalk

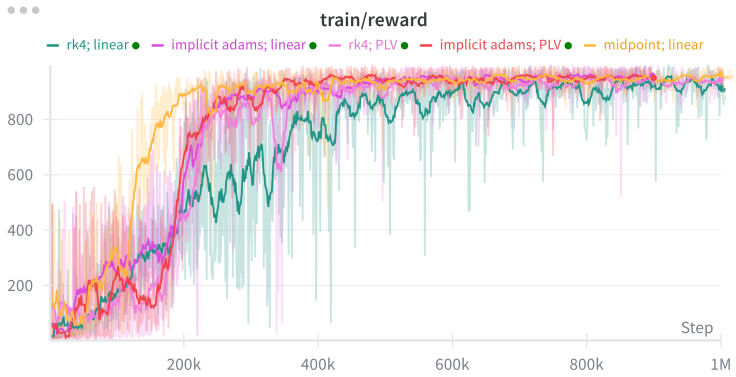

Рис. 3: Сравнение методов сэмплинга в среде QuadrupedWalk

Список литературы

[Edw+20] Ashley Edwards и др. «Estimating q (s, s') with deep deterministic dynamics gradients». В: *International Conference on Machine Learning*. PMLR. 2020, с. 2825—2835.

[Abb+22] Ali Abbasi и др. «Sparsity and heterogeneous dropout for continual learning in the null space of neural activations». В: *Conference on Lifelong Learning Agents*. PMLR. 2022, с. 617—628.

[Lip+22] Yaron Lipman и др. «Flow matching for generative modeling». В: *arXiv preprint arXiv:2210.02747* (2022).

[Lyu+22] Jiafei Lyu и др. «State advantage weighting for offline rl». В: *arXiv preprint arXiv:2210.04251* (2022).

[Lip+24] Yaron Lipman и др. «Flow matching guide and code». В: *arXiv preprint arXiv:2412.06264* (2024).

[Wan+24] Liyuan Wang и др. «A comprehensive survey of continual learning: Theory, method and application». В: *IEEE transactions on pattern analysis and machine intelligence* 46.8 (2024), с. 5362—5383.

[Chi+25] Cheng Chi и др. «Diffusion policy: Visuomotor policy learning via action diffusion». В: *The International Journal of Robotics Research* 44.10-11 (2025), с. 1684—1704.

[LZL25] Qiyang Li, Zhiyuan Zhou и Sergey Levine. «Reinforcement learning with action chunking». В: *arXiv preprint arXiv:2507.07969* (2025).

[Lv+25] Lei Lv и др. «Flow-Based Policy for Online Reinforcement Learning». В: *arXiv preprint arXiv:2506.12811* (2025).

[NWL25] Amir Noorizadegan, Sifan Wang и Leevan Ling. «A Practitioner's Guide to Kolmogorov-Arnold Networks». В: *arXiv preprint arXiv:2510.25781* (2025).

[Bla+] Kevin Black и др. «π0: A vision-language-action flow model for general robot control. CoRR, abs/2410.24164, 2024. doi: 10.48550». В: *arXiv preprint ARXIV.2410.24164* ().