ИСПОЛЬЗОВАНИЕ ACTOR-CRITIC АЛГОРИТМОВ ПРИ ОБУЧЕНИИ АГЕНТОВ ДЛЯ ИГР НА ATARI 2600

Зязюлькин С. П., Нестеренков С. Н.
Факультет компьютерных систем и сетей, кафедра программного обеспечения информационных технологий, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: sergeyzyazyulkin@gmail.com, nsn@bsuir.by

Последние достижения в области машинного обучения с подкреплением позволяют обучать агентов, значительно превосходящих человека в решении некоторых задач. Одним из самых популярных бенчмарков для алгоритмов и моделей машинного обучения с подкреплением являются игры на Atari 2600. В данной статье рассматриваются actor-critic алгоритмы и их модификации при обучении агентов для игр на Atari 2600.

Введение

Агент представляет собой сущность, которая получает на вход наблюдение (состояние) окружающей среды s и выполняет некоторое действие a, определяемое политикой π, с целью максимизации суммарной награды. Существует два больших семейства алгоритмов машинного обучения с подкреплением: DQN [1] и Policy Gradient [2].

DQN алгоритмы предполагают приближение нейронной сетью Q-функции, определяющей ценность действия a при наблюдении s:

$$Q(s,a) = E_{s' \sim P(\cdot|s,a)} \left[R(s,a) + \gamma \max_{a' \in A} Q(s',a') \right],$$

где A – множество допустимых действий, R – награда за выполнение действия a в состоянии s, γ – дисконтирующий множитель.

Политика агента в этом случае является детерминированной и имеет следующий вид:

$$\pi(s) = \text{argmax}_a Q(s,a).$$

Policy Gradient алгоритмы, в отличие от DQN, предполагают приближение нейронной сетью непосредственно самой политики π. Во-первых, это позволяет избежать необходимости решать задачу оптимизации – находить действие, имеющее максимальную ценность в данном состоянии. Для сред с небольшим числом допустимых действий решение задачи оптимизации не является проблемой, однако задача оптимизации может стать очень сложной в случае, когда число допустимых действий большое или действие является непрерывным, т.е. имеет бесконечное число допустимых значений. Во-вторых, Policy Gradient позволяет обучать стохастические политики, когда в DQN предпринимаемое действие является детерминированным для каждого состояния.

Градиент политики определяется следующей формулой:

$$\nabla J \approx E \left[Q(s,a) \nabla \log \pi(a|s) \right].$$

С практической точки зрения оптимизация градиента политики может быть реализована как оптимизация функции потерь вида

$$L = -Q(s,a) \log \pi(a|s).$$

Значение функции $Q(s,a)$ может быть разбито на две части: ценность состояния $V(s)$ и преимущество $A(s,a)$, которое даёт действие a. Использование $A(s,a)$ вместо $Q(s,a)$ в градиенте политики позволяет существенно повысить стабильность обучения. Для вычисления $V(s)$ используется отдельная нейронная сеть. Нейронная сеть, отвечающая за вероятности действий, называется actor, отвечающая за ценность состояния – critic. Этот алгоритм имеет название advantage actor-critic или A2C.

Модификации actor-critic алгоритмов

Для улучшения исследования среды в Policy Gradient алгоритмах часто добавляется entropy loss:

$$L_H = -\sum_i \pi_\theta(s_i) \log \pi_\theta(s_i).$$

Entropy loss принимаем минимальное значение в случае равномерного распределения вероятностей. Использование entropy loss позволяет избегать политик, в которых одно действие слишком сильно доминирует над остальными.

Алгоритм A2C допускает эффективное распараллеливание. Создаётся несколько процессов, каждый из которых взаимодействует с одной или несколькими средами и генерирует переходы (s,a,r,s'). Все переходы собираются на одном процессе, который вычисляет функцию потерь и выполняет шаги стохастического градиентного спуска. Обновлённые веса нейронной сети передаются на все процессы, выполняющие взаимодействие со средой. Параллельный вариант алгоритма A2C с асинхронным взаимодействием со средами получил название asynchronous advantage actor-critic или A3C [3].

Дальнейшим улучшением A2C является алгоритм PPO, предложенный командой OpenAI в

2017-м году [4]. Основным предлагаемым алгоритмом улучшением является изменение формулы градиента политики. Градиент политики вычисляется как отношение новой и старой политик, масштабируемое значением преимущества действия:

$$J_\theta = E_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t\right].$$

Использование такого значения градиента политики может приводить к очень большим изменениям в весах нейронной сети, что может нарушить весь процесс обучения, т.к. данные для дальнейшего обучения генерируются текущей политикой, т.е. ухудшение политики приводит к снижению качества генерируемых данных. Для решения этой проблемы величина отношения новой и старой политик ограничивается интервалом $[1-\epsilon, 1+\epsilon]$. Варьирование величины ϵ позволяет ограничить величину изменения весов.

Ещё одно отличие от A2C – способ приближения величины преимущества действия. В A2C преимущество вычисляется по формуле

$$A_t = -V(s_t) + r_t + \ldots + \gamma^{T-t-1}r_{T-1} + \gamma^{T-t}V(s_T).$$

В PPO используется более общее приближение преимущества:

$$A_t = \sigma_t + (\gamma\lambda)\sigma_{t+1} + \ldots + (\gamma\lambda)^{T-t-1}\sigma_{T-1},$$
$$\text{где } \sigma_t = r_t + \gamma V(s_{t+1}) - V(s_t).$$

Приближение преимущества в A2C является частным случаем приближения преимущества в PPO при $\lambda = 1$.

Также стоит отметить, что в PPO имеются отличия в процессе обучения: генерируется длинная последовательность переходов, после чего подсчитываются преимущества действий для всей последовательности и выполняется несколько эпох обучения.

Одной из больших проблем машинного обучения с подкреплением является процесс исследования среды во время обучения. Этой проблеме посвящена работа [5]. В ней предложено использовать дополнительную награду за «любопытство». Это позволяет улучшить процесс исследования среды в процессе обучения и, как следствие, улучшить качество итоговых агентов, когда награда является разреженной. Награда за «любопытство» вычисляется при помощи двух нейронных сетей со схожей архитектурой. Одна из сетей, именуемая целевой, инициализируется случайными весами. Эти веса остаются неизменными на всём процессе обучения. Другая сеть, именуемая предсказывающей, обучается предсказывать выходы целевой нейронной сети. На вход сетям подаётся итоговое состояние среды на текущем шаге. Для вычисления величины награды за «любопытство» используется следующая формула:

$$r_t^i = \left\|\hat{f}(s_{t+1}) - f(s_{t+1})\right\|_2^2,$$

где $\hat{f}$ и f – предсказывающая и целевая сети соответственно. Данный подход получил название RND (Random Network Distillation).

Рассмотрим способы повышения скорости обучения при обучении агентов для игр на Atari 2600. На вход игровому агенту подаётся несколько цветных кадров игрового пространства с разрешением 210x160. Преобразование игровых кадров в чёрно-белые изображения меньшего разрешения позволяет значительно уменьшить размер кадра. Так, например, сжатие кадра до чёрно-белого изображения с разрешением 80x80 позволяет уменьшить размер почти в 16 раз, что значительно сказывается на скорости обучения, при этом обычно отсутствует заметное влияние на качество итогового игрового агента. Другой возможной оптимизацией является применение выбранного игровым агентом действия для нескольких последовательных взаимодействий со средой. Если частота взаимодействий со средой велика и агент часто предпринимает одно и то же действие для последовательных взаимодействий, имеется возможность «сэкономить» на числе взаимодействий агента без потери качества, особенно в условиях, когда обработка состояния среды игровым агентом является «тяжёлой» операцией.

Заключение

Рассмотренные actor-critic алгоритмы и их модификации позволяют обучать игровых агентов, превосходящих человека в большом числе игр на Atari 2600. Изучение влияния отдельных модификаций и их комбинаций на обучение игровых агентов для конкретных игр на Atari 2600, а также поиск иных модификаций остаются открытыми вопросами для дальнейшего изучения.

1. Mnih, V. Playing Atari with Deep Reinforcement Learning / V. Mnih, K. Kavukcuoglu, D. Silver, [и др.] // arXiv:1312.5602.
2. Thomas, P. Policy Gradient Methods for Reinforcement Learning with Function Approximation and Action-Dependent Baseline / P. Thomas, E. Brunskill // arXiv:1706.06643.
3. Mnih, V. Asynchronous Methods for Deep Reinforcement Learning / V. Mnih, M. Mehdi, A. Graves, [и др.] // arXiv:1602.01783.
4. Schulman, J. Proximal Police Optimization Algorithms / J. Schulman, F. Wolski, P. Dhariwal, [и др.] // arXiv:1707.06347.
5. Burda, Y. Exploration by Random Network Distillation / Y. Burda, H. Edwards, A. Storkey, O. Klimov // arXiv:1810.12894.