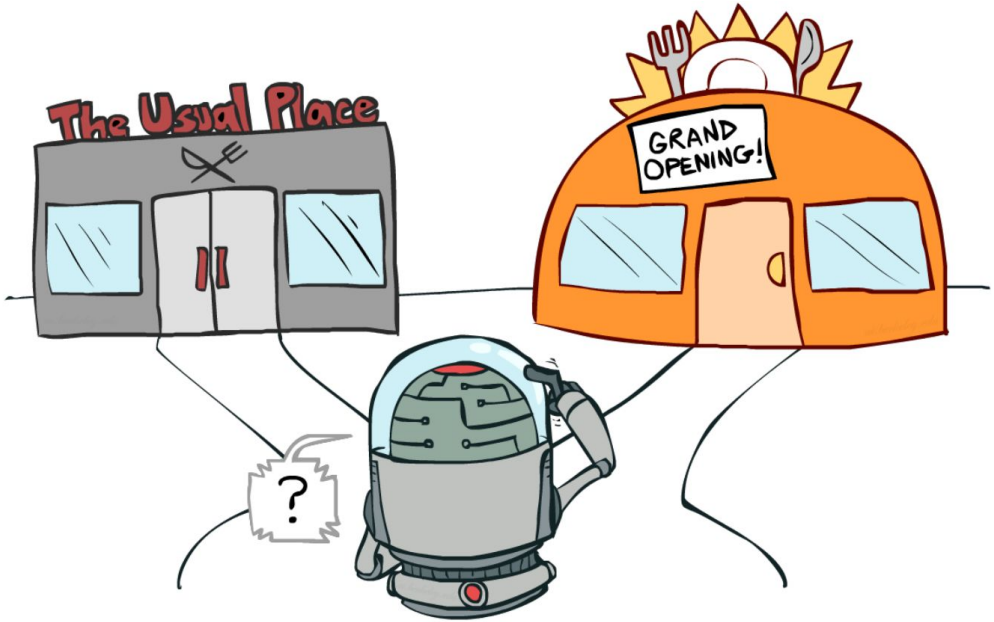
Exploration in deep RL

2 марта

Шошин Борис

В чем проблема?

Хотим, чтобы наша функция не останавливалась в локальном максимуме вознаграждения, а исследовала другие пути максимизации. Если все время действовать жадно, то можем так и не узнать о вариантах сильно лучше. Это приводит к известной проблеме [exploitation vs exploration].



Классические методы исследования.

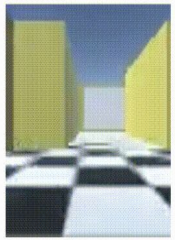
- ε-Greedy - Случайное действие с вероятностью ε
- Максимизируем $\widehat{Q}_t(a) + \widehat{U}_t(a)$, где $\widehat{U}_t(a)$ - функция, обратная частоте выбора действия a. То есть стараемся выбирать чаще то действие, которое до этого редко выбирали.

Основные проблемы исследования

The Hard-Exploration Problem. Проблема исследования, при котором пространство наград очень разреженное или обманчивое. В таком случае для достижения награды необходимо совершить много однонаправленных действий, что исключают наши базовые стратегии.

Noisy-TV Problem. Проблема шумного телевизора. Агент акцентирует внимание на незначительных деталях.

Agent in a maze with a noisy TV



Agent in a maze without a noisy TV

Внутренние награды

Идея, взятая из психологии. Будем добавлять агенту внутренней мотивации. Для этого введем внутренние награды.

$$R_t = R_t^e + \alpha R_t^i$$

R_t^e -внешняя награда, полученная из среды

R_t^i -внутренняя награда, которую мы сами определяем

Таким образом мы сможем мотивировать агента к исследованию при помощи дополнительной награды.

Intrinsic Curiosity Module

Будем обучать функцию f для предсказания следующего состояния $f\left(\phi(s_t), a_t\right) \rightarrow \phi(s_{t+1})$, где ϕ - функция кодирующая наше исходное пространство состояний. И тогда определим внутреннюю награду, как

$$R_t^i = \left\| f\left(\phi(s_t), a_t\right) - \phi(s_{t+1}) \right\|^2$$

Таким образом, чем больше мы знаем про состояние, тем лучше мы умеем его предсказывать и тем меньше наша награда. То есть мы мотивируем агента ходить в плохо изученные места и предпринимать новые действия.

Intrinsic Curiosity Module

Осталось определиться с тем, откуда брать хорошую ϕ(s).

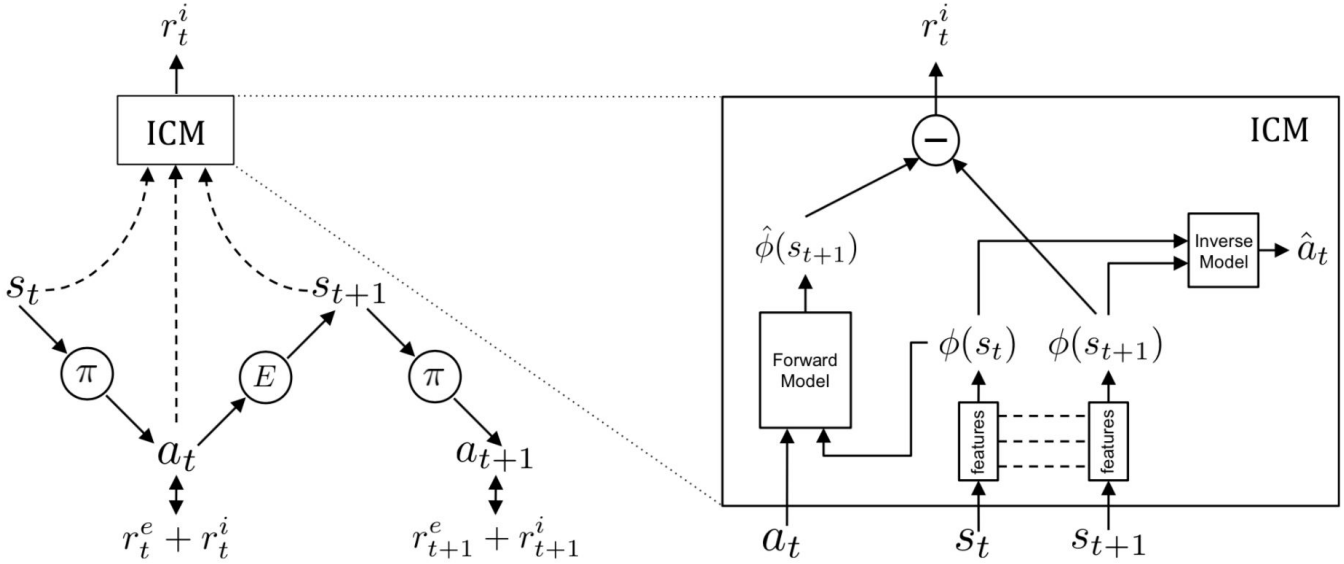
Для этого будем обучать g $$g\Big(\phi(s_t), \phi(s_{t+1})\Big) \rightarrow a_t$$

То есть мы стараемся научиться предсказывать действие исходя из текущего и последующего за ним состояния. Это хорошо, потому что так мы не будем брать во внимания случайности, которые агент не может учитывать.

Intrinsic Curiosity Module

Таким образом обучаем $g\Big(\phi(s_t), \phi(s_{t+1})\Big) \rightarrow a_t \;;\; f\Big(\phi(s_t), a_t\Big) \rightarrow \phi(s_{t+1})$

И считаем внутреннюю награду как $R_t^i = \Big\| f\Big(\phi(s_t), a_t\Big) - \phi(s_{t+1}) \Big\|^2$



DORA(Directed Outreaching Reinforcement Action-Selection)

Два параллельных марковских процесса:

- Основной с наградами R
- Побочный с наградами 0. Его Q-value будем называть E-value

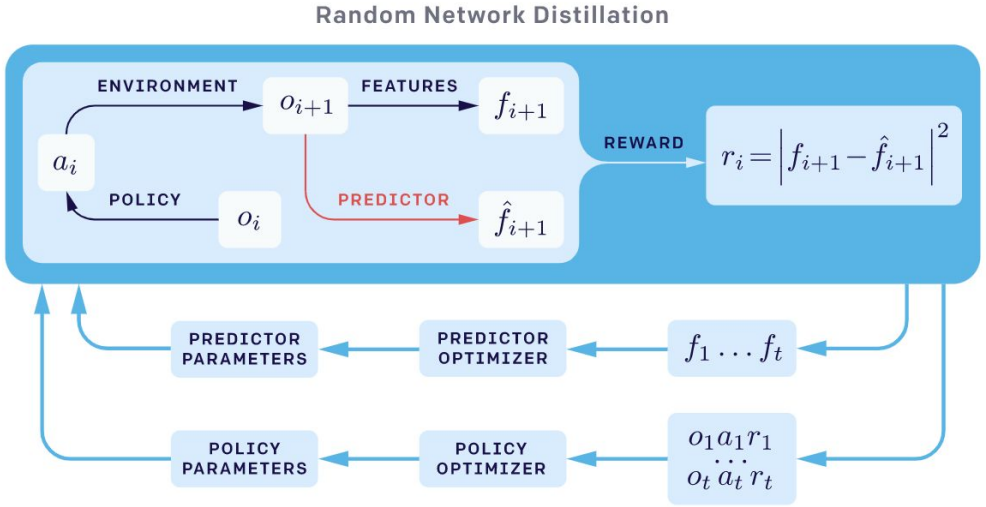
$$R_t^i = \frac{1}{\sqrt{-log\Big(E(s_t, a_t)\Big)}}$$

Чем больше раз бывали в определенном состоянии - тем меньше E-value и тем меньше награда.

Random Network Distillation

Возьмем какую-нибудь $f(s_t)$ и будем обучать $\widehat{f}(s_t;\theta) \rightarrow f$

То есть будем обучать новую функцию под какую-то исходно взятую функцию. И возьмем $R_t^i = ||\widehat{f}(s_{t+1};\theta) - f(s_{t+1})||^2$, тогда чем больше мы посетим состояние, тем лучше научимся предсказывать f и тем меньше будет награда.



Random Network Distillation

- Очень важен грамотный выбор α, так как внутренние награды могут иметь абсолютно произвольный масштаб
- Лучше накапливать знания про f между эпизодами и не обновлять её каждый раз

Never Give Up

- Для изучения признакового пространства используется та же идея, что в ICM
- Работает с чем-то эпизодическим. Внутренняя награда состоит из 2 частей: эпизодической и меж-эпизодической. $r_t^i = r_t^{episodic} * \max(\min(r_t^{cross-episodic}, L)); L - const$
- Для меж-эпизодической награды берем такую же награду, как в RND, но отнормированную к 1. $r_t^{cross-episodic} = 1 + \frac{e^{RND}(s_t) - \mu}{\sigma}$

Never Give Up

Эпизодическая награда

Для эпизодической награды поддерживаем эпизодическую память M.

Для этого после каждого шага добавляем $\phi(s)$ в M.

$$r_t^{episodic} = \frac{1}{\sqrt{n(\phi(s_t))}} \approx \frac{1}{\sqrt{\sum_{\phi_i \in N_k} K\left(\phi_i, \phi(s_t)\right)} + c}$$

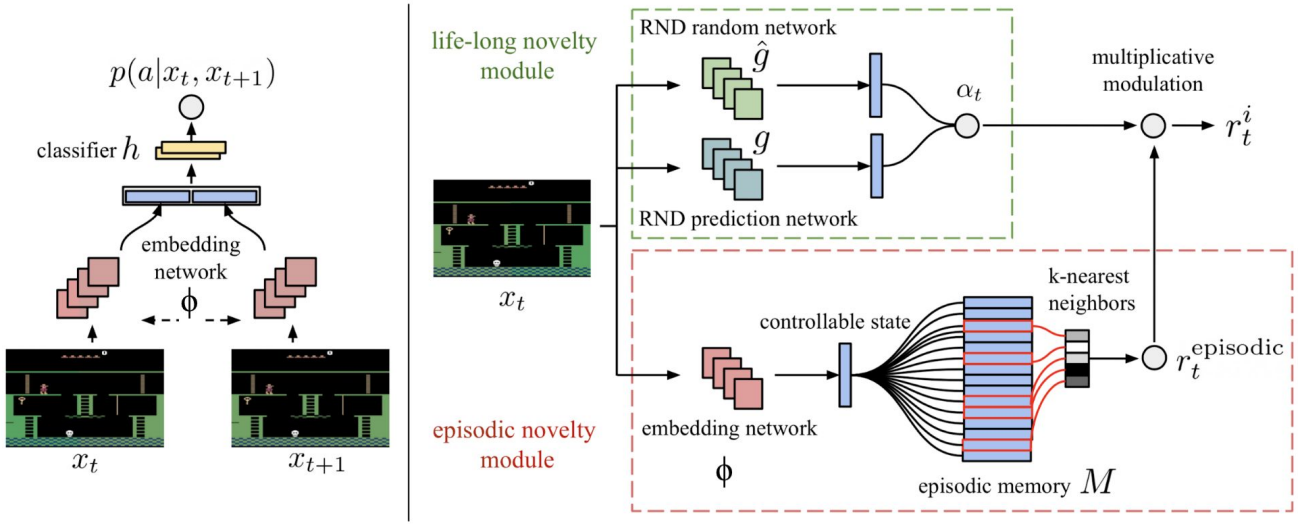
n(s) - количество посещений состояния s
N_k - k ближайших соседей.

$$K(x, y) = \frac{\epsilon_2}{\frac{d^2(x,y)}{d_m^2} + \epsilon_2}$$

d(x,y) - Евклидово расстояние
d_m - среднее расстояние до k ближайших соседей

Never Give Up

Основная идея - алгоритм быстро препятствует посещению одного и того же состояния в одном эпизоде при этом медленно препятствует посещению одного и того же состояния на протяжении разных эпизодов



Agent57

2 улучшения NGU

1. Используем семейство политик с параметрами (α_j, y_j). Выбираем политику с помощью мета-контроллера(sliding-window UCB bandit algorithm).
2. $Q(s, a; \theta_j) = Q(s, a; \theta_j^e) + \alpha_j Q(s, a; \theta_j^i)$.

   Учим по отдельности $Q(s, a; \theta_j^e)$ и $Q(s, a; \theta_j^i)$

Agent57 - первый алгоритм превзошедший человека во всех 57 Atari games

Источники

- https://lilianweng.github.io/lil-log/2020/06/07/exploration-strategies-in-deep-reinforcement-learning.html
- https://arxiv.org/abs/1705.05363
- https://arxiv.org/abs/1804.04012
- https://medium.com/data-from-the-trenches/curiosity-driven-learning-through-random-network-distillation-488ffd8e5938
- https://arxiv.org/abs/2002.06038
- https://arxiv.org/abs/2003.13350