Policy gradient
REINFORCE
Reparametrization trick

Подготовил:
Федоров Никита, БПМИ202

nmfedorov@edu.hse.ru

План

1. Вводные понятия, описание сеттинга задачи
   1.1. Детерминистические и стохастические политики
   1.2. Value- и policy-based алгоритмы
   1.3. Симулятор среды. Model-free алгоритмы при отсутствии симулятора
   1.4. Нейросеть в качестве политики
   1.5. Истинное значение V- и Q-функции для данной политики
2. Вывод формулы градиента целевого функционала по параметрам политики
   2.1. Описание целевого функционала (что максимизируем)
   2.2. Вывод формулы градиента
3. Алгоритм REINFORCE
   3.1. Описание алгоритма
   3.2. Большие проблемы
   3.3. Бейзлайны, решение проблем
4. Reparametrization trick
5. Сравнение двух подходов

Детерминистические и стохастические политики

Политика $\pi(s)$ по текущему состоянию s возвращает

- либо одно конкретное действие, которое наш агент выполняет при наступлении s (**детерминистическая политика**)

- либо распределение вероятности на действиях, возможных в состоянии s (**стохастическая политика**). При этом агент выбирает действие из этого распределения

Value- и policy-based алгоритмы

Value-based сразу пытаются найти V- и Q-функции оптимальной (самой лучшей) политики, а потом выводят итоговую политику, как $\pi(s) = argmax_a\ Q(a, s)$

Value iteration *Policy iteration* *Q-learning*

Policy-based напрямую ищут оптимальную политику, пытаясь максимизировать получаемый reward

Cross entropy method *REINFORCE*

Симулятор среды
Model-free алгоритмы при отсутствии симулятора

Знает ли агент

- переходные вероятности: $p(s'|s,a)$
- функцию награды: $p(r|s,a)$ (еще часто обозначают $r(s,a)$)

Если агент знает ответы на эти вопросы, то говорят, что у него есть **симулятор среды** (можно использовать value и policy iteration)

Далее мы будем считать, что *симулятора среды у нас нет*!

Нейросеть в качестве политики

Пусть политикой нашего агента будет нейросеть $\pi(s, \theta)$ и мы будем ее учить!

Множество действий конечно

$\pi(s, \theta)$ - классификатор, формирующий распределение вероятности на действиях

Множество действий континуально

$\pi(s, \theta)$ - регрессор, предсказывающий действие $\mu \in \mathbb{R}$

Конечное действия выбирается агентом из распределения $\mathcal{N}(\mu, \sigma^2)$

Истинное значение V- и Q-функции для данной политики

Каждой конкретной политике π соответствуют истинные для нее $V^{\pi}(s)$ и $Q^{\pi}(s,a)$

Policy gradient
Описание целевого функционала

Хотим максимизировать $J(\pi_\theta) = V^{\pi_\theta}(s_0)$

Policy gradient
Вывод градиента

$$\nabla_\theta V^{\pi_\theta}(s) = \nabla_\theta \mathbb{E}_{a\sim\pi_\theta(a|s)} Q^{\pi_\theta}(s,a) =$$

$$= \{\text{мат.ожидание — это интеграл}\} =$$

$$= \nabla_\theta \int_{\mathcal{A}} \pi_\theta(a \mid s) Q^{\pi_\theta}(s,a)\, \mathrm{d}a =$$

$$= \{\text{проносим градиент внутрь интеграла}\} =$$

$$= \int_{\mathcal{A}} \nabla_\theta \left[\pi_\theta(a \mid s) Q^{\pi_\theta}(s,a)\right] \mathrm{d}a =$$

$$= \{\text{правило градиента произведения}\} =$$

$$= \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) Q^{\pi}(s,a)\, \mathrm{d}a + \int_{\mathcal{A}} \pi(a \mid s) \nabla_\theta Q^{\pi_\theta}(s,a)\, \mathrm{d}a =$$

$$= \{\text{второе слагаемое — это мат.ожидание}\} =$$

$$= \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) Q^{\pi}(s,a)\, \mathrm{d}a + \mathbb{E}_a \nabla_\theta Q^{\pi_\theta}(s,a) =$$

$$= \{\text{log-derivative trick (2.7)}\} =$$

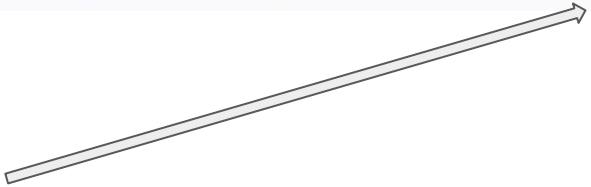
$$= \int_{\mathcal{A}} \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi}(s,a)\, \mathrm{d}a + \mathbb{E}_a \nabla_\theta Q^{\pi_\theta}(s,a) =$$

$$= \{\text{первое слагаемое тоже стало мат.ожиданием}\} =$$

$$= \mathbb{E}_a \left[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi}(s,a) + \nabla_\theta Q^{\pi_\theta}(s,a)\right]$$

Policy gradient
Вывод градиента

$$\nabla_\theta V^{\pi_\theta}(s) = \mathbb{E}_a \left[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s,a) + \nabla_\theta Q^{\pi_\theta}(s,a)\right]$$



$$\begin{aligned}\nabla_\theta Q^{\pi_\theta}(s,a) &= \{(3.5)\} = \nabla_\theta \left[r(s,a) + \gamma \mathbb{E}_{s'} V^{\pi_\theta}(s')\right] = \\ &= \{r(s,a) \text{ не зависит от } \theta,\ p(s' \mid s,a) \text{ тоже}\} = \\ &= \gamma \mathbb{E}_{s'} \nabla_\theta V^{\pi_\theta}(s')\end{aligned}$$

$$\nabla_\theta V^{\pi_\theta}(s) = \mathbb{E}_a \mathbb{E}_{s'} \left[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s,a) + \gamma \nabla_\theta V^{\pi_\theta}(s')\right]$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\mathcal{T} \sim \pi} \sum_{t \geq 0} \gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^\pi(s_t, a_t)$$

REINFORCE

Алгоритм 19: REINFORCE

Гиперпараметры: N — количество игр, $\pi(a \mid s, \theta)$ — стратегия с параметрами θ, SGD-оптимизатор.

Инициализировать θ произвольно
На очередном шаге t:

1. играем N игр $\mathcal{T}_1, \mathcal{T}_2 \dots \mathcal{T}_N \sim \pi$
2. для каждого t в каждой игре $\mathcal{T}$ считаем reward-to-go: $R_t(\mathcal{T}) := \sum_{\hat{t} \geqslant t} \gamma^{\hat{t}-t} r_{\hat{t}}$
3. считаем оценку градиента:

$$\nabla_\theta J(\pi) := \frac{1}{N} \sum_{\mathcal{T}} \sum_{t \geq 0} \gamma^t \nabla_\theta \log \pi(a_t \mid s_t, \theta) R_t(\mathcal{T})$$

4. делаем шаг градиентного подъёма по θ, используя $\nabla_\theta J(\pi)$

Беды с алгоритмом

У данного подхода есть 2 большие проблемы:

1. Очень высокая дисперсия оценки градиента. Хоть мы и имеем формально доказанную сходимость этого алгоритма, наши шаги настолько неточные и хаотичные, что обучение занимает слишком много времени
2. Каждую игру агент должен доигрывать до конца (проблемы с бесконечными играми)

Бейзлайны в REINFORCE

$$\nabla_\theta J(\pi) \approx \mathbb{E}_{\mathcal{T}\sim\pi} \sum_{t\geq 0} \nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^\pi(s_t, a_t)$$

Мы можем вычесть из Q любую функцию, зависящую от s, и при этом значение градиента не изменится! Эта функция называется **бэйзлайном**.

$$\nabla_\theta J(\pi_\theta) = \mathrm{E}_{\mathcal{T}} \sum_{t\geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)(Q^{\pi_\theta}(s_t, a_t) - b(s_t))$$

Бейзлайны в REINFORCE

Если в качестве бейзлайна использовать $b(s) = V^{\pi}(s)$, то дисперсия нашей оценки градиента значительно сократится!

Пусть наша нейросеть $\pi_\theta(s)$ помимо действия будет предсказывать $V^{\pi_\theta}(s)$

Тогда мы можем оценить $Q^{\pi_\theta}(s_t, a_t) \approx r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1})$

Бейзлайны в REINFORCE

$$\nabla J \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t \geq 0} \nabla \log \pi_\theta(a_t|s_t)(r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t))$$

$$Loss = \frac{1}{N} \sum_{i=1}^{N} \sum_{t \geq 0} ((r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1})) - V^{\pi_\theta}(s_t))^2$$

P.S. Это чинит 2 упомянутые проблемы REINFORCE

Reparametrization trick

Как дифференцирует REINFORCE

$$\begin{aligned}\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] &= \nabla_\theta \int f(x)p_\theta(x)dx \\ &= \int f(x)\nabla_\theta p_\theta(x)dx \\ &= \int f(x)p_\theta(x)\nabla_\theta \log p_\theta(x)dx \\ &= \mathbb{E}_{x\sim p_\theta(x)}[f(x)\nabla_\theta \log p_\theta(x)]\end{aligned}$$

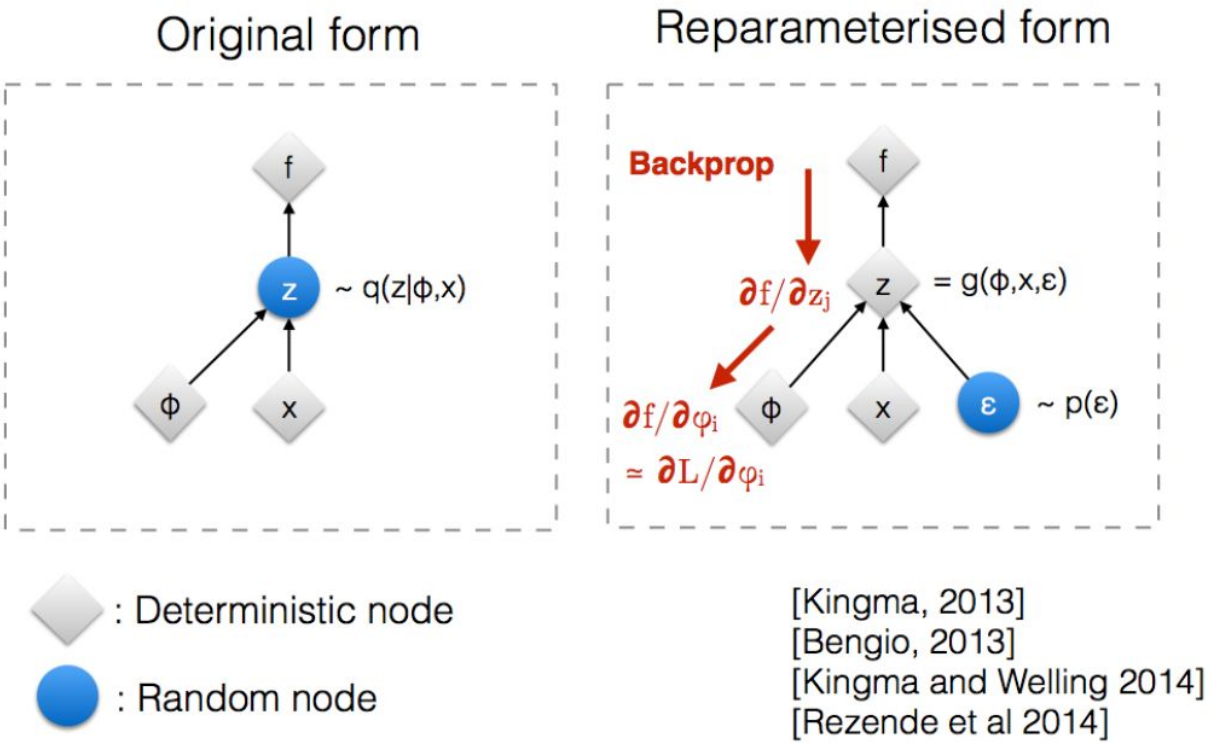
Как дифференцирует reparamerization trick

$$\begin{aligned}\varepsilon &\sim q(\varepsilon) \\ x &= g_\theta(\varepsilon) \\ \nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] &= \nabla_\theta \mathbb{E}_{\varepsilon\sim q(\varepsilon)}[f(g_\theta(\varepsilon))] \\ &= \mathbb{E}_{\varepsilon\sim q(\varepsilon)}[\nabla_\theta f(g_\theta(\varepsilon))]\end{aligned}$$

Пример: если $x \sim \mathcal{N}(\mu, \sigma^2)$, то мы можем перепараметризовать так:

$$g_\theta(\varepsilon) = \mu_\theta + \varepsilon\sigma_\theta \qquad \varepsilon \sim \mathcal{N}(0, 1)$$

Reparametrization trick



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

REINFORCE VS Reparametrization

Properties	REINFORCE	Reparameterization
Differentiability requirements	Can work with a non-differentiable model	Needs a differentiable model
Gradient variance	High variance; needs variance reduction techniques	Low variance due to implicit modeling of dependencies
Type of distribution	Works for both discrete and continuous distributions	In the current form, only valid for continuous distributions
Family of distribution	Works for a large class of distributions of x	It should be possible to reparameterize x as done above

Источники

О policy gradient и REINFORCE:

- https://arxiv.org/pdf/2201.09746.pdf
- https://towardsdatascience.com/policy-gradient-methods-104c783251e0
- https://github.com/yandexdataschool/Practical_RL/tree/master/week06_policy_based

О reparametrization trick:

- http://stillbreeze.github.io/REINFORCE-vs-Reparameterization-trick/
- https://deepganteam.medium.com/basic-policy-gradients-with-the-reparameterization-trick-24312c7dbcd
- https://stats.stackexchange.com/questions/199605/how-does-the-reparameterization-trick-for-vaes-work-and-why-is-it-important#:~:text=Reparameterization%20trick%20is%20a%20way,(%CE%B8%2B%CF%B5)2%5D