Обучение с подкреплением

On-policy Deep RL

JET BRAINS Research

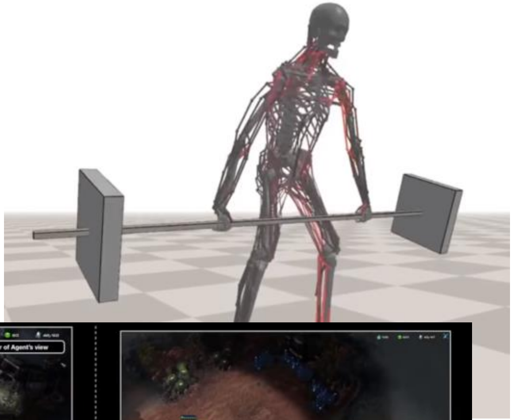
академия больших данных mail.ru group made

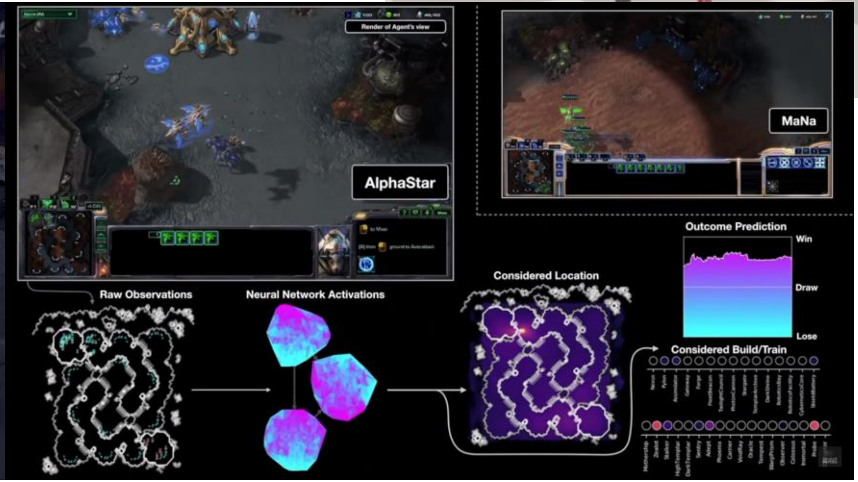
HIGHER SCHOOL OF ECONOMICS

NATIONAL RESEARCH UNIVERSITY

Напоминание: ограничения

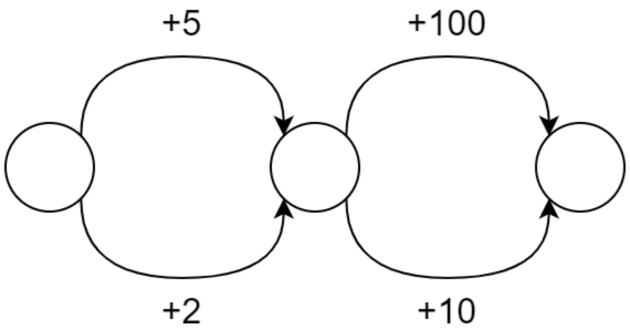
1) ~~Должны знать T и R~~
2) ~~Дискретное пространство состояний~~
3) Дискретное пространство действий



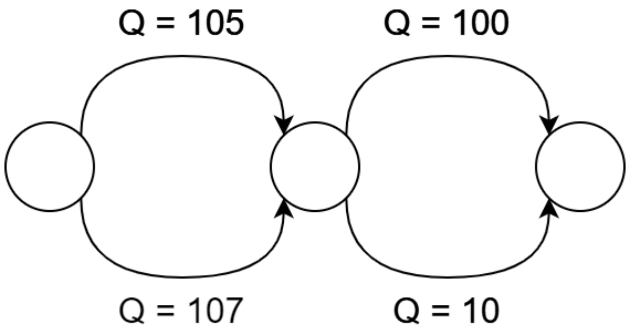


Мотивация

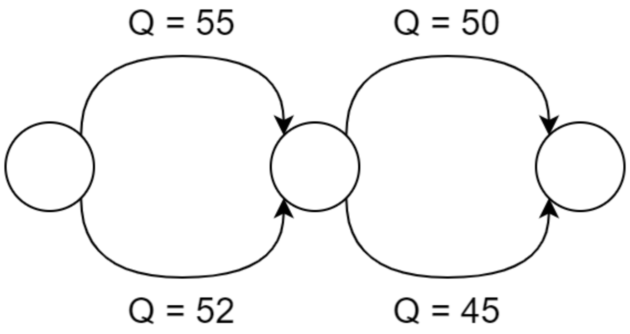
Какое из двух приближений Q-function лучше?



Вариант 1

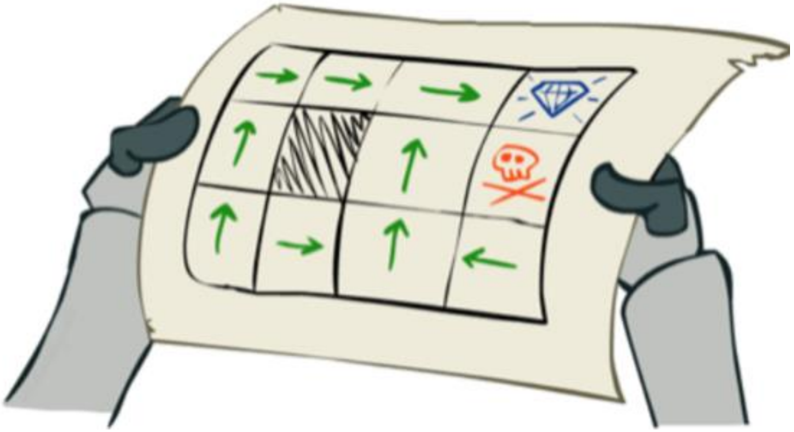


Вариант 2



Напоминание: Policy

π:



Политика/стратегия $\pi(s) : S \to A$ - правило, по которому агент принимает решения.

Оптимальная стратегия: $\pi^*(s) : \sum_{i=0}^{T} \gamma^i R(s_i, a_i, s_{i+1}) \underset{a}{\to} \max$

На самом деле хотим найти оптимальную политику!

Оптимизируем политику напрямую

Предположим, что политика $\pi_\theta(a|s)$ - стохастическая

Идея: будем итеративно максимизировать вероятность действий, которые привели к лучшему результату

Cross-entropy method:

1. Случайно инициализируем политику
2. В течении **T** итераций:
    1. Получить **N** эпизодов из среды
    2. Выбрать из них **M < N** лучших эпизодов
    3. Использовать лучшие эпизоды для оптимизации политики

Cross-entropy method

Алгоритм:

1. Случайно инициализируем политику
2. В течении **T** итераций:
    1. Получить **N** эпизодов из среды
    2. Выбрать из них **M < N** лучших эпизодов
    3. Использовать лучшие эпизоды для оптимизации политики

Вопросы:

1. Как именно оптимизировать политику в 2.3?
2. Что делать с exploration?
3. Можно ли использовать предыдущий опыт?

Cross-entropy method

Алгоритм:

1. Случайно инициализируем политику
2. В течении **T** итераций:
    1. Получить **N** эпизодов из среды
    2. Выбрать из них **M < N** лучших эпизодов
    3. Использовать лучшие эпизоды для оптимизации политики

Вопрос: Как именно оптимизировать политику в 2.3?

Ответ:
В дискретном случае подойдет любой способ классификации.
В континуальном случае можем приближать распределение и макс. *logprob*
Если мы считаем, что действия распределены нормально, то можно зафиксировать std и минимизировать MSE (эквивалентно максимизации *logprob*)

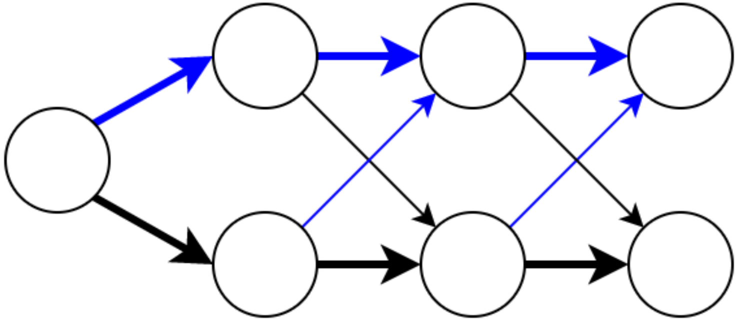
Cross-entropy method

Вопрос: Можно ли использовать предыдущий опыт?

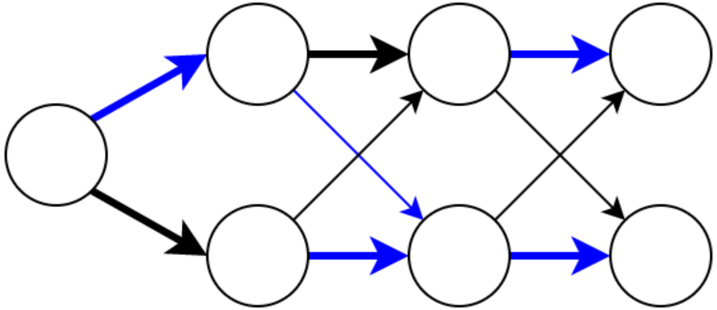
Мы хотим максимизировать $\mathbb{E}_{\tau \in \mathrm{Best}} \, p(\tau | \pi_\theta) \underset{\theta}{\to} \max$

Важно понимать, что распределение Best также зависит от текущей политики:
Синий цвет - решения, принимаемые политикой
Тонкие ребра - не оптимальные переходы в среде

Политика 1:



Политика 2:



Cross-entropy method

Алгоритм:

1. Случайно инициализируем политику
2. В течении **T** итераций:
    1. Получить **N** эпизодов из среды
    2. Выбрать из них **M < N** лучших эпизодов
    3. Использовать лучшие эпизоды для оптимизации политики

Проблема:

1. Никак не учитываем то, насколько хорош эпизод

1. (Пока) нет никакого способа исследовать среду

Переформулируем задачу

Предположим, что политика $\pi_\theta(a|s)$ - стохастическая и задана нейронной сетью

Также предположим, что умеем считать $\nabla_\theta \pi_\theta(a|s)$

Как и раньше, хотим оптимизировать математическое ожидание суммарной награды:

$$\mathbb{E}_{\tau \sim \pi_\theta} \sum_{i=0}^{\mathrm{len}(\tau)} r_i$$

Идея:
Поскольку политика задана нейронной сетью, можно напрямую оптимизировать награду, если посчитаем градиент

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta} \sum_{i=0}^{\mathrm{len}(\tau)} r_i$$

Log-derivative trick

Перепишем мат. ожидание как интеграл:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta} \sum_{i=0}^{\mathrm{len}(\tau)} r_i = \nabla_\theta \int p(\tau|\pi_\theta) \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau$$

Траектория не зависит от параметров дифференцирования, Можем внести дифференциал под интеграл:

$$\nabla_\theta \int p(\tau|\pi_\theta) \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau = \int [\nabla_\theta p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau$$

Умножим и поделим на вероятность траектории:

$$\int [\nabla_\theta p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau = \int \frac{p(\tau|\pi_\theta)}{p(\tau|\pi_\theta)} [\nabla_\theta p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau$$

Log-derivative trick

$$\int [\nabla_\theta p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau = \int \frac{p(\tau|\pi_\theta)}{p(\tau|\pi_\theta)} [\nabla_\theta p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau$$

По формуле сложной производной:

$$\int \frac{p(\tau|\pi_\theta)}{p(\tau|\pi_\theta)} [\nabla_\theta p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau = \int p(\tau|\pi_\theta) [\nabla_\theta \log p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau$$

Превратим получившейся интеграл в матожидание:

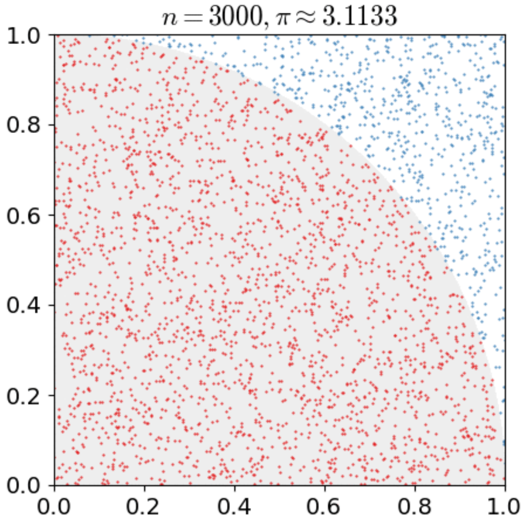
$$\int p(\tau|\pi_\theta) [\nabla_\theta \log p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i d\tau = \mathbb{E}_{\tau\sim\pi_\theta} [\nabla_\theta \log p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i$$

Метод Monte Carlo

Метод Monte Carlo заключается в том, что мы приближаем математическое ожидание с помощью семплов из распределения:

$$\mathbb{E}_X f(x) = \int p(x) f(x) dx \approx \frac{1}{n} \sum_{1}^{n} x_i, \quad x_i \sim X$$

Пример: приближаем площадь четверти круга



$$S = \int\limits_{x,y=0}^{1} 1_{x^2+y^2 \leq 1}(x, y) dx dy =$$

$$\mathbb{E}_{x,y \sim U[0,1]} 1_{x^2+y^2 \leq 1}(x, y) \approx \frac{1}{n} \sum_{1}^{n} 1_{x^2+y^2 \leq 1}(x, y)$$

Метод Monte Carlo

Приблизим математическое ожидание с помощью метода Монте-Карло:

$$\mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i \approx \frac{1}{n}\sum_{1}^{n}[\nabla_\theta \log p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i$$

Раскроем логарифм вероятности траектории:

$$\frac{1}{n}\sum_{1}^{n}[\nabla_\theta \log p(\tau|\pi_\theta)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i = \frac{1}{n}\sum_{1}^{n}[\nabla_\theta (C + \sum_{i=0}^{\mathrm{len}(\tau)} \log \pi_\theta(a_i|s_i))] \sum_{i=0}^{\mathrm{len}(\tau)} r_i$$

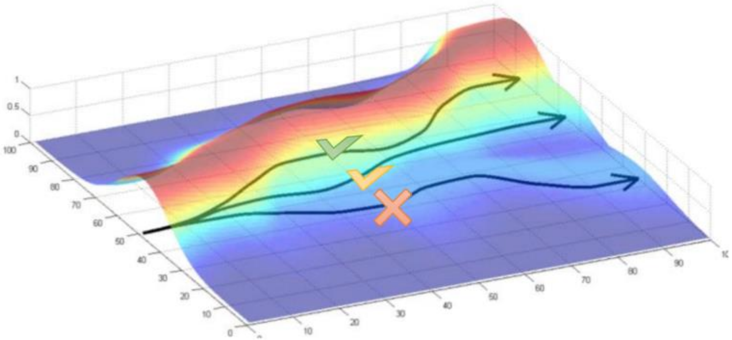
C не зависит от параметров сети, поэтому:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta} \sum_{i=0}^{\mathrm{len}(\tau)} r_i \approx \frac{1}{n}\sum_{1}^{n}[\nabla_\theta \sum_{i=0}^{\mathrm{len}(\tau)} \log \pi_\theta(a_i|s_i)] \sum_{i=0}^{\mathrm{len}(\tau)} r_i$$

Алгоритм REINFORCE

1. Инициализировать политику
2. В течении **T** итераций:
    1. Получить **N** траекторий из среды в соотвтетствии с текущей политикой
    2. Обновить политику с помощью градиентного спуска по математическому ожиданию суммарной награды:

$$\nabla_\theta - \mathbb{E}_{\tau \sim \pi_\theta} \sum_{i=0}^{\text{len}(\tau)} r_i \approx \frac{-1}{n} \sum_1^n [\nabla_\theta \sum_{i=0}^{\text{len}(\tau)} \log \pi_\theta(a_i|s_i)] \sum_{i=0}^{\text{len}(\tau)} r_i$$



Алгоритм REINFORCE

Замечания:

1. Алгоритм не зависит от того, какую функцию мы максимизируем. Это значит, что можно максимизировать любую функцию:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta} f(\tau) \approx \frac{1}{n}\sum_{1}^{n}[\nabla_\theta \sum_{i=0}^{\mathrm{len}(\tau)} \log \pi_\theta(a_i|s_i)] f(\tau)$$

1. Можно ввести baseline-функцию, которая поможет алгоритму обучиться:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{i=0}^{\mathrm{len}(\tau)} r_i - b(\tau)] \approx \frac{1}{n}\sum_{1}^{n}[\nabla_\theta \sum_{i=0}^{\mathrm{len}(\tau)} \log \pi_\theta(a_i|s_i)][\sum_{i=0}^{\mathrm{len}(\tau)} r_i - b(\tau)]$$

1. Baseline-функцией может быть что угодно. Например, константа или value-функция.

Немного модификаций

Заметим, что

$$\frac{1}{n}\sum_{1}^{n}[\nabla_\theta \sum_{i=0}^{\text{len}(\tau)} \log \pi_\theta(a_i|s_i)][\sum_{i=0}^{\text{len}(\tau)} r_i] = \frac{1}{n}\sum_{1}^{n}[\sum_{i=0}^{\text{len}(\tau)} \nabla_\theta \log \pi_\theta(a_i|s_i)][\sum_{i=0}^{\text{len}(\tau)} r_i]$$

А также то, что награды не зависят от будущих действий, поэтому мы не хотим, чтобы они влияли на градиент:

$$\nabla_\theta L(i) = \nabla_\theta \log \pi_\theta(a_i|s_i)[\sum_{j=i}^{\text{len}(\tau)} r_j]$$

И добавим коэффициент дисконтирования:

$$\nabla_\theta L(i) = \nabla_\theta \log \pi_\theta(a_i|s_i)[\sum_{j=i}^{\text{len}(\tau)} \gamma^{j-i} r_j]$$

Алгоритм Actor-Critic

Раньше мы считали суммарную награду честно:

$$\nabla_\theta L(i) = \nabla_\theta \log \pi_\theta(a_i|s_i)[\sum_{j=i}^{\mathrm{len}(\tau)} \gamma^{j-i} r_j]$$

А теперь будем считать с помощью value-функции:

$$\nabla_\theta L(i) = \nabla_\theta \log \pi_\theta(a_i|s_i)(r_i + \gamma V_{\pi_\theta}(s_{i+1}))$$

или

$$\nabla_\theta L(i) = \nabla_\theta \log \pi_\theta(a_i|s_i)(r_i + \gamma V_{\pi_\theta}(s_{i+1}) - V_{\pi_\theta}(s_i)) = A_{\pi_\theta}(s_i, a_i)\nabla_\theta \log \pi_\theta(a_i|s_i)$$

А поскольку value-функцию мы не знаем, то будем приближать ее с помощью еще одной нейронной сети - критика

Алгоритм Actor-Critic

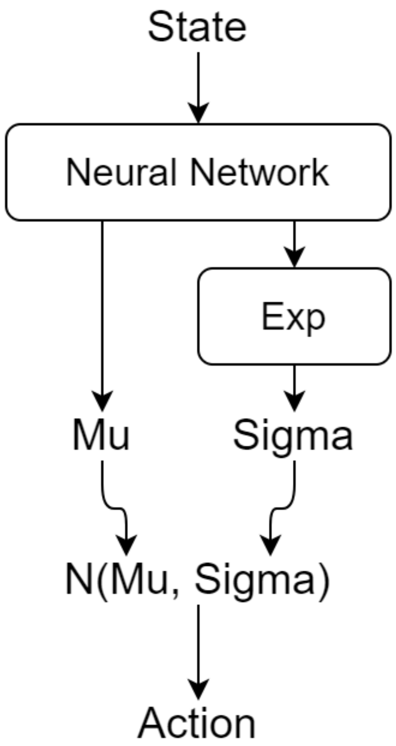
Актор - нейронная сеть, определяющая политику. Важно, чтобы можно было посчитать логарифм вероятности распределения действий, совершаемых политикой.

Критик - нейронная сеть, используемая для приближения value-функции.

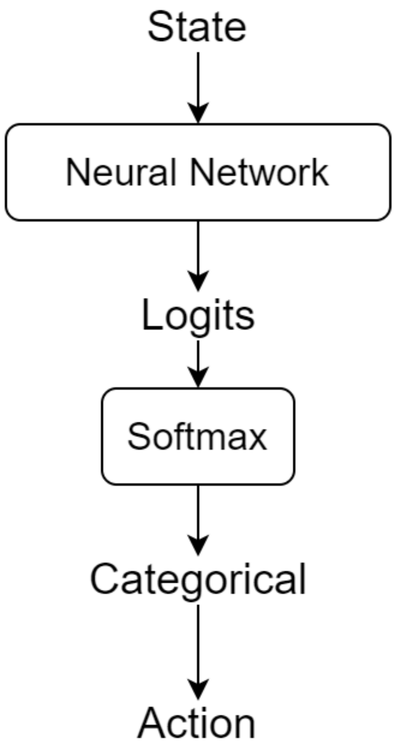
Алгоритм:
1. Инициализировать критика и актора
2. В течении **T** итераций:
    1. Получить **N** эпизодов из среды
    2. Посчитать приближение **R** суммарной награды или advantage
    1. Обновить сеть актора с помощью градиента:
    $$R(s_i, a_i)\nabla_\theta \log \pi_\theta(a_i|s_i)$$
    1. Обновить сеть критика

Алгоритм Actor-Critic

Континуальный Actor



Дискретный Actor



Алгоритм Actor-Critic

Проблема: Можем сойтись к жадной субоптимальной политике из-за того, что никак не мотивируем агента исследовать среду

Решения:

1. Добавим сглаживание к распределению. Например, можно ограничить минимальное значение sigma в нормальном распределении.

1. Добавить регуляризацию к функции ошибки. Для этого отлично подойдет энтропия.

   Напоминание: энтропия - это мера неопределенности

$$H(p(x)) = -\int p(x) \log p(x) dx$$

Алгоритм Actor-Critic

Для определения суммарной дисконтированной награды можно использовать разные функции:

1. One-step: $R_t = r_t + \gamma V(s_{t+1})$

1. N-step: $R_{t:t+n-1} = \sum_{i=t}^{t+n-1} \gamma^i r_i + \gamma^n V(s_{t+n})$

1. Lambda-return: $R_t^\lambda = (1-\lambda)[\sum_{i=1}^{T} \lambda^{i-1} R_{t:t+i}] + \lambda^T R_{t:t+T}$

Проблемы алгоритмов

REINFORCE:
1) Оценка Монте-Карло требует большого количества сэмплов для сходимости
2) Высокая чувствительность к learning rate

Actor-Critic:
1) Высокая чувствительность к learning rate
2) Оценка с помощью приближенной value-функции может иметь высокую ошибку. (Это частично исправляется с помощью lambda-return)

Алгоритм REINFORCE

Замечания:

1. Алгоритм не зависит от того, какую функцию мы максимизируем. Это значит, что можно максимизировать любую функцию:

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta} f(\tau) \approx \frac{1}{n} \sum_{1}^{n} [\nabla_\theta \sum_{i=0}^{\text{len}(\tau)} \log \pi_\theta(a_i|s_i)] f(\tau)$$

1. Можно ввести baseline-функцию, которая поможет алгоритму обучиться:

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta} [\sum_{i=0}^{\text{len}(\tau)} r_i - b(\tau)] \approx \frac{1}{n} \sum_{1}^{n} [\nabla_\theta \sum_{i=0}^{\text{len}(\tau)} \log \pi_\theta(a_i|s_i)] [\sum_{i=0}^{\text{len}(\tau)} r_i - b(\tau)]$$

1. Baseline-функцией может быть что угодно. Например, константа или value-функция.

Алгоритм REINFORCE

Алгоритм REINFORCE можно применять не только для решения задач RL, но и для решения любых задач, где метрика качества не дифференцируема. Например:

1. Bayesian models

1. Natural Language Processing

1. Рекомендательные системы и ранжирование

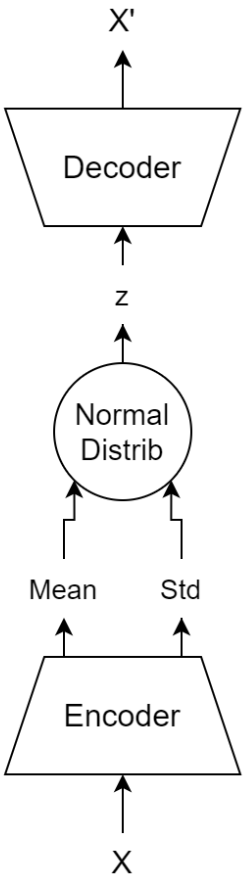
1. Любые задачи взаимодействия с пользователем, где в качестве основной метрики качества используется оценка пользователя

VAE

Идея: Хотим научиться семплировать из *p(x)*
используя скрытые переменные *z*

Для этого обучаем Encoder, который приближает распределение *q(z | x)* и Decoder, который восстанавливает распределение *p(x | z)*.

При этом мы хотим, чтобы распределение *q(z | x)* было близко к некоторому априорному *p(z)*, чтобы потом мы могли брать *z ~ p(z)* и генерировать *x*.



VAE

Хотим приблизить распределение $p(x)$ используя некоторое латентное распределение. Для этого мы выводим оценку снизу на логарифм вероятности:

$$\ln p(x) = \ln \int p(x,z)dz = \ln \int p(x,z)\frac{q_\theta(z|x)}{q_\theta(z|x)}dz$$

$$\geq \mathbb{E}_{z\sim q_\theta(z|x)} \ln \frac{p(x,z)}{q(z|x)} = \mathbb{E}_{z\sim q_\theta(z|x)} \ln \frac{p_\phi(x|z)p(z)}{q_\theta(z|x)}$$

$$= \mathbb{E}_{z\sim q_\theta(z|x)} \ln p_\phi(x|z) - \mathbb{E}_{z\sim q_\theta(z|x)} \ln \frac{q_\theta(z|x)}{p(z)}$$

Полученную нижнюю оценку мы хотим максимизировать, поэтому нашей функцией потерь будет:

$$L = \mathrm{D}_{\mathrm{KL}}(q_\theta(z|x_i)\|p(z)) - \mathbb{E}_{z\sim q_\theta(z|x_i)} \log p_\phi(x_i|z)$$

Semi-supervised VAE

Теперь предположим, что для некоторых объектов в обучающей выборке есть метки классов, но при этом не все данные размечены.

Теперь хотим максимизировать $\ln p(x) + \ln p(x, y)$

Неразмеченные данные Размеченные данные

Построив ELBO для левой и правой части отдельно, получим:

$$\mathbb{E}_{y \sim q_\chi(y|x), z \sim q_\theta(z|x,y)} \log \frac{p(x, y|z)p(z)}{q_\theta(z|x, y)q_\chi(y|x)} + \mathbb{E}_{z \sim q_\theta(z|x,y)} \log \frac{p(x, y|z)p(z)}{q_\theta(z|x, y)}$$

Поскольку в такой постановке метки y не несут в себе никакой смысловой нагрузки, мы также добавим ограничение на правильную классификацию *x* с помощью $q_\chi(y|x)$

А где тут REINFORCE?

Будем использовать REINFORCE для обновления $q_\chi(y|x)$ в случае неразмеченных данных:

$$\nabla_\chi \mathbb{E}_{y\sim q_\chi(y|x), z\sim q_\theta(z|x,y)} \log \frac{p(x,y|z)p(z)}{q_\theta(z|x,y)q_\chi(y|x)}$$

$$= \mathbb{E}_{y\sim q_\chi(y|x), z\sim q_\theta(z|x,y)} [\nabla_\chi \log q_\chi(y|x)] \log \frac{p(x,y|z)p(z)}{q_\theta(z|x,y)q_\chi(y|x)}$$

В частности, если *y* - это метки классов, то теперь вместо того, чтобы считать честное мат. ожидание, мы можем просто семплировать метки из классификатора

REINFORCE в NLP

В NLP очень много не дифференцируемых метрик:

- ROGUE - метрика для суммаризации текстов, которая считается на основе пересечения множества n-грамм из ground truth и из предсказания модели.
- BLEU - метрика, часто используемая для машинного перевода. Также позволяет оценить похожесть полученного моделью текста на набор референсов.
- И другие

Кроме того, в диалоговых системах важно принимать решения последовательно, поэтому RL в целом тут подходит еще лучше