ТЕХНОСФЕРА

Занятие № 13

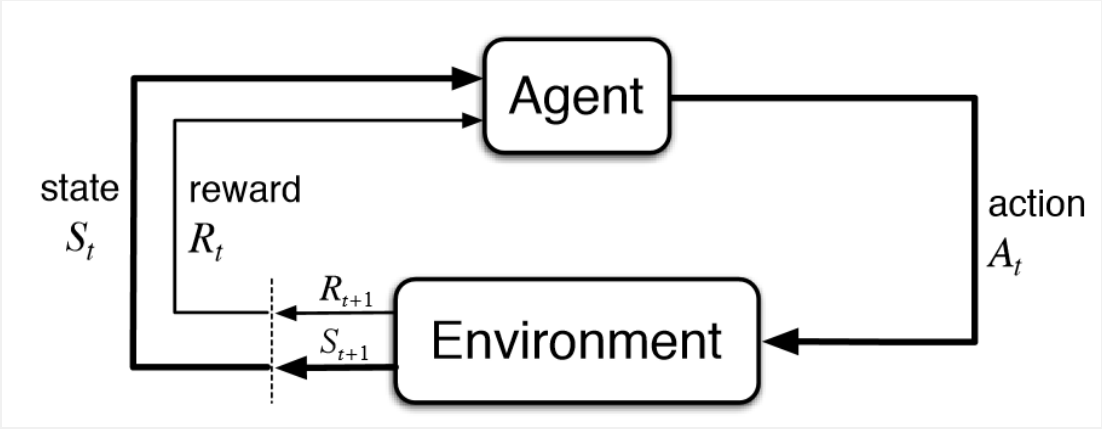
Обучение с подкреплением

Алексей Спасёнов, Евгений Некрасов

План занятия

1. Обучение с подкреплением
2. OpenAI Gym
3. Эволюционные стратегии

Обучение с подкреплением



Agent

- Получает reward R_t
- Получает состояние S_t
- Совершает действие A_t

Environment

- Получает действие A_t
- Генерирует состояние S_{t+1}
- Генерирует reward R_{t+1}

Обучение с подкреплением

Примеры решаемых задач

Управление портфелем ценных бумаг

Игры
Шахматы
Победа +1
Поражение -1

Робототехника
Движение по траектории

Обучение с подкреплением

Награды (Rewards)

- R_t- скаляр
- Задача агента – максимизировать среднюю сумму полученных R_t
- Любая задача может быть сформулирована в виде максимизации суммы R_t

Обучение с подкреплением

Состояние

В процессе взаимодействия со средой агент накапливает историю $H_t = R_1, O_1, A_1, \ldots, R_t, O_t, A_t$.

Для принятия решения хранение всей истории может быть крайне избыточно.

Игры: O_t - скриншот экрана

Робототехника: + информация со всех датчиков

Мы хотим иметь такое представление истории $S_t = f(H_t)$, которое было бы «достаточной статистикой» для будущего.

Обучение с подкреплением

Марковское свойство

Стохастический процесс обладает Марковским свойством, если условное распределение вероятностей будущих состояний процесса зависит только от нынешнего состояния, а не от последовательности событий, которые предшествовали этому.

Процесс, обладающий этим свойством называется Марковским процессом.

Пусть S_t - последовательность случайных величин (векторов, элементов). Последовательность обладает Марковским свойством, если

$$\Pr(S_{t+1}|S_t) = \Pr(S_{t+1}|S_t, S_{t-1}, \dots, S_1)$$

Т.е. S_t достаточно для предсказания будущих состояний.

Обучение с подкреплением

Матрица переходов (transition matrix)

Пусть S_t - последовательность дискретных состояний.

Поскольку последовательность задаётся распределение $\Pr(S_{t+1}|S_t)$ естественно упорядочить его в матрицу.

$$P_{ss'} = \Pr(S_{t+1} = s'|S_t = s)$$

$$P = \begin{matrix} \\ from \end{matrix} \begin{matrix} to \\ \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \end{matrix}$$

Обучение с подкреплением

Марковский процесс

Марковский процесс (цепь) это кортеж (S, P), где

- S – принимает дискретные, конечные, значения
- P – матрица переходов (transition matrix)

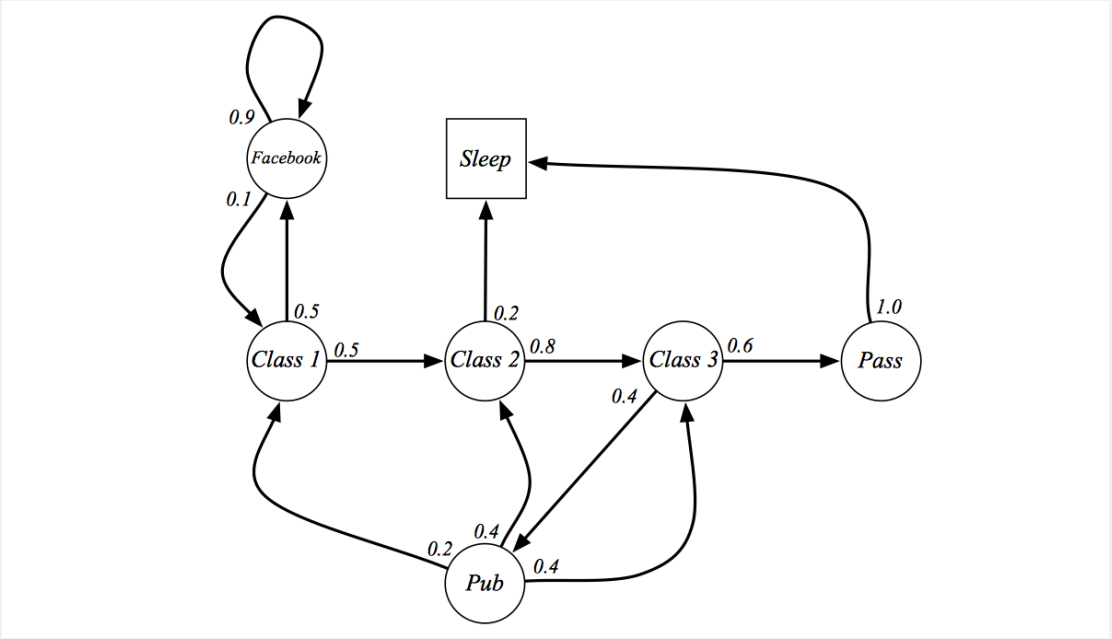
$$P_{ss'} = \Pr(S_{t+1} = s' | S_t = s)$$

Строго говоря, необходимо распределение начальных состояний (но мы предполагаем, что оно вырождено, т.е. мы знаем, где начинаем с вероятностью 1).

Марковский процесс – основа обучения с подкреплением (reinforcement learning)

Обучение с подкреплением

Марковский процесс, пример

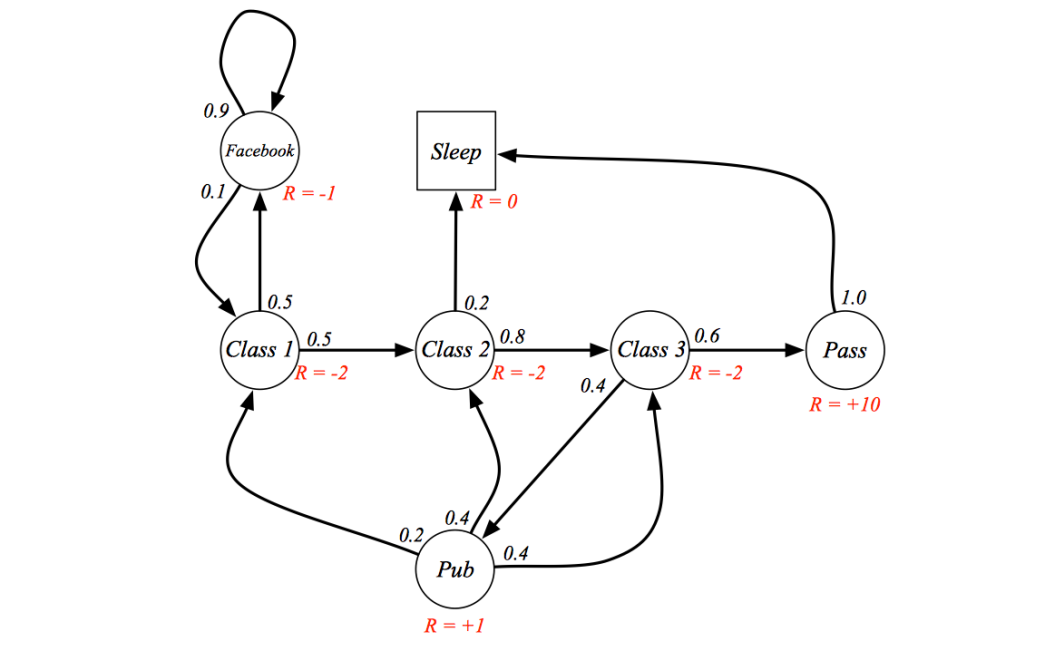


$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

https://www.52coding.com.cn/index.php?/Articles/single/70

Обучение с подкреплением

Марковский процесс, пример + rewards



$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

Обучение с подкреплением

Марковский процесс, пример + rewards

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{i=0}^{\infty} \gamma^i R_{t+1+i}$$

Под G_t понимаем дисконтированную сумму всех будущих rewards.

- $\gamma \in [0,1]$ - коэффициент дисконтирования (discount factor)
- Единая форма для «конечных» и «бесконечных» моделей
- Обеспечивает сходимость ряда ($\max|R_{t+1}| < C = const$)
- Нетерпимость (impatience) – насколько важно получить reward сейчас, чем потом.

По определению G_t - случайная величина (не привязанная ни к чему).

Обучение с подкреплением

Ценность состояния

$$v(s) = \mathrm{E}[G_t | S_t = s]$$

Ценность состояния – ожидаемая сумма всех полученных rewards, если стартовать из s

$$\begin{aligned} v(s) &= \mathrm{E}[G_t | S_t = s] \\ &= \mathrm{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s] \\ &= \mathrm{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) | S_t = s] \\ &= \mathrm{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned}$$

Уравнение Беллмана:

$$v(S) = \mathrm{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]$$

$$v = R + \gamma P v$$

Обучение с подкреплением

Markov reward process

MRP это кортеж (S, R, P, γ), где

- S – принимает дискретные (конечные значения)
- R – функция rewards
- P – матрица переходов
- γ – discount factor

Осталось добавить только actions.

Обучение с подкреплением

Markov decision process

MRP это кортеж (S, R, A, P, γ), где

- S – принимает дискретные (дискретное пространство)
- A – действия (дискретное пространство)
- R – функция rewards
- P – матрица переходов
- γ – discount factor

Policy, Q-function

Стратегия (policy)

$\pi(a|s) = \Pr(A_t = a|S_t = s)$ – то, как мы выбираем действия, оказавшись в состоянии s (стратегия фиксируется в начале каждой игры)

Ценность состояния (value-function)

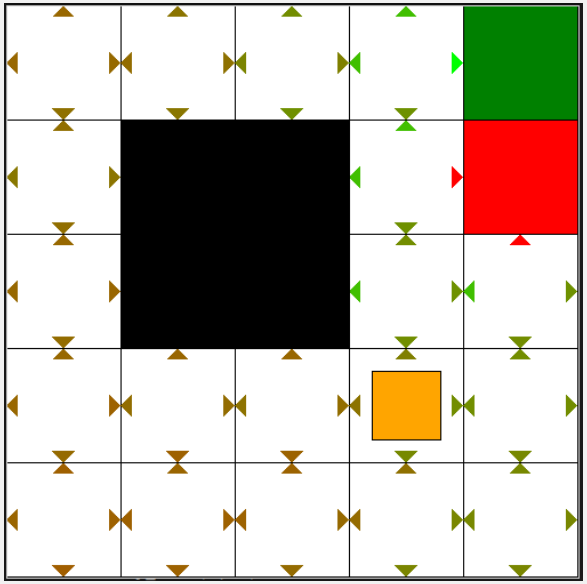
$v_\pi(s) = \mathrm{E}_\pi[G_t|S_t = s]$ – ценность состояния определяет ещё и стратегия

Q-function

$q_\pi(s, a) = \mathrm{E}_\pi[G_t|S_t = s, A_t = a]$ – ценность действия в состоянии s

Обучение с подкреплением

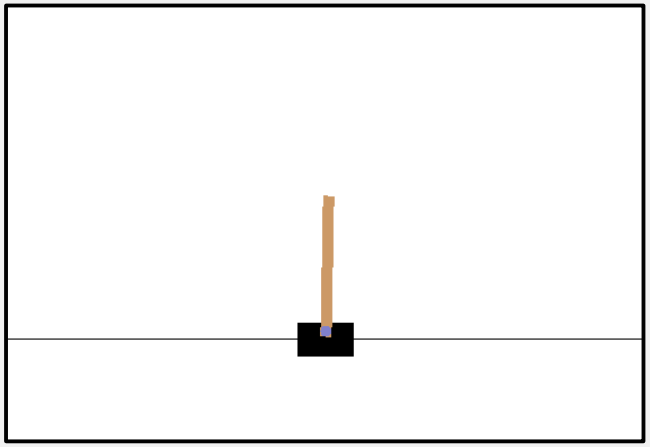
Пример 1

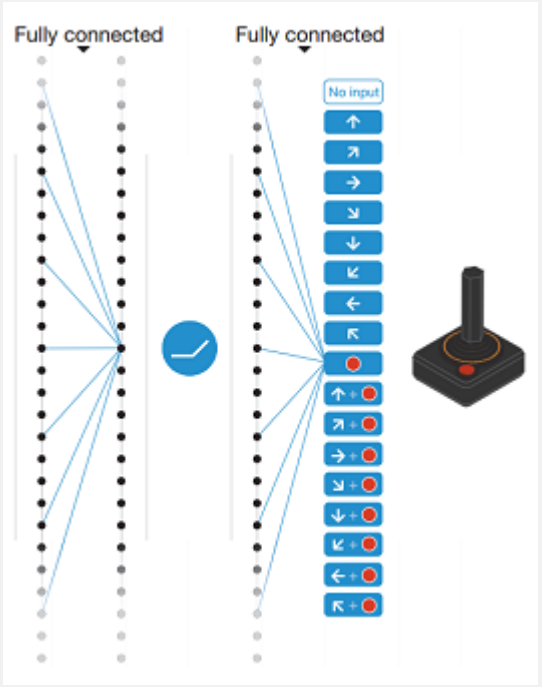


https://www.youtube.com/watch?v=A5eihauRQvo
https://github.com/llSourcell/q_learning_demo

Обучение с подкреплением

Пример 2



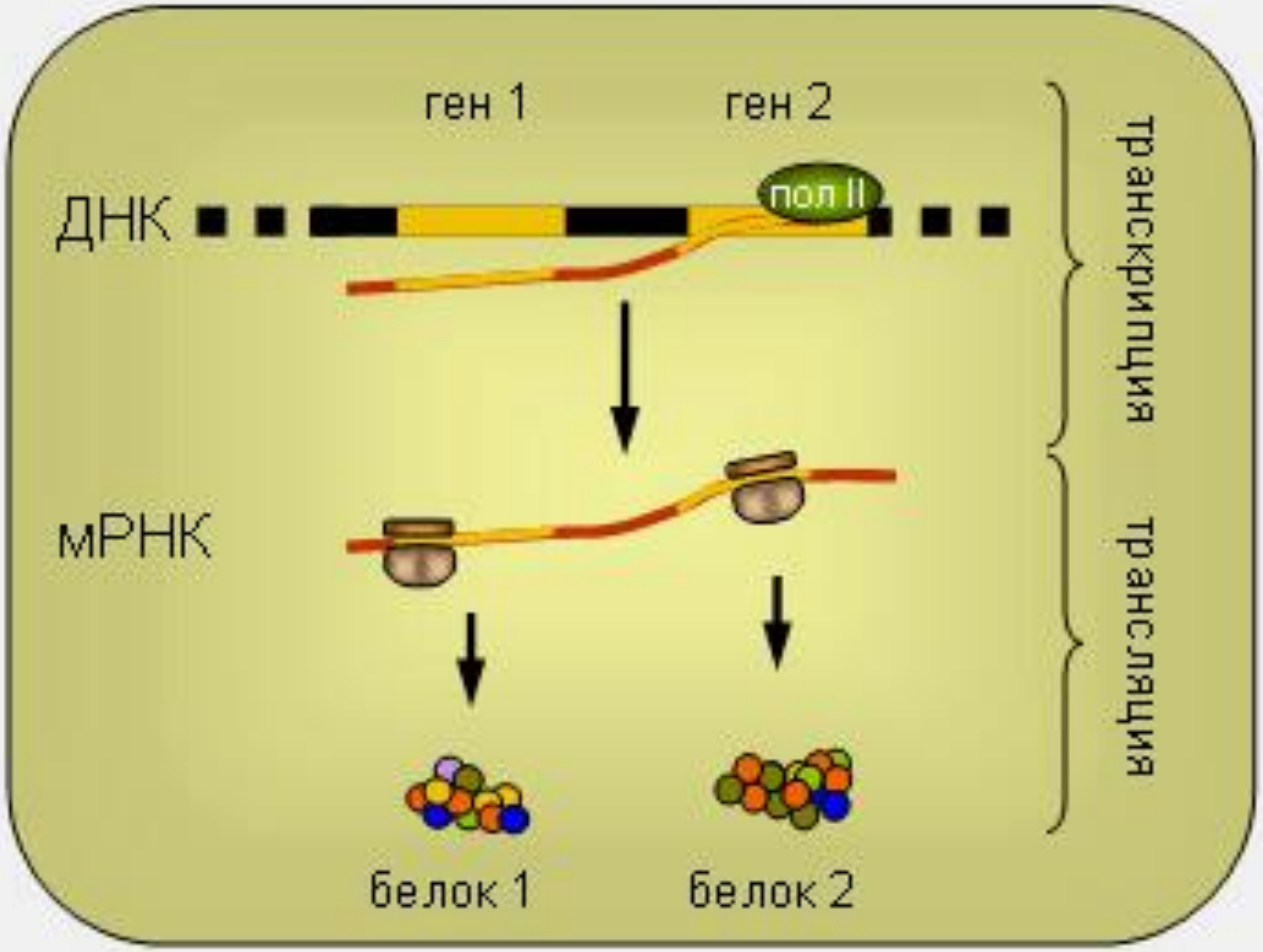


https://github.com/keras-rl/keras-rl/blob/master/examples/dqn_cartpole.py

Теория эволюции Ч. Дарвина (1859)

- Основной движущей силой эволюции является естественный отбор
- Отбор, действуя на особей, позволяет выживать и оставлять потомство тем организмам, которые лучше приспособлены для жизни в данном окружении
- Действие отбора приводит к распадению видов на части - дочерние виды, которые, в свою очередь, со временем расходятся до родов, семейств и всех более крупных таксонов

Молекулярно-генетические основы эволюции



Эволюционные стратегии

- Вдохновлены биологической эволюцией
- Не являются симуляцией биологической эволюции
- С математической точки зрения это black-box оптимизация

Эволюционные стратегии, постановка задачи

Пусть дана функция f(w)

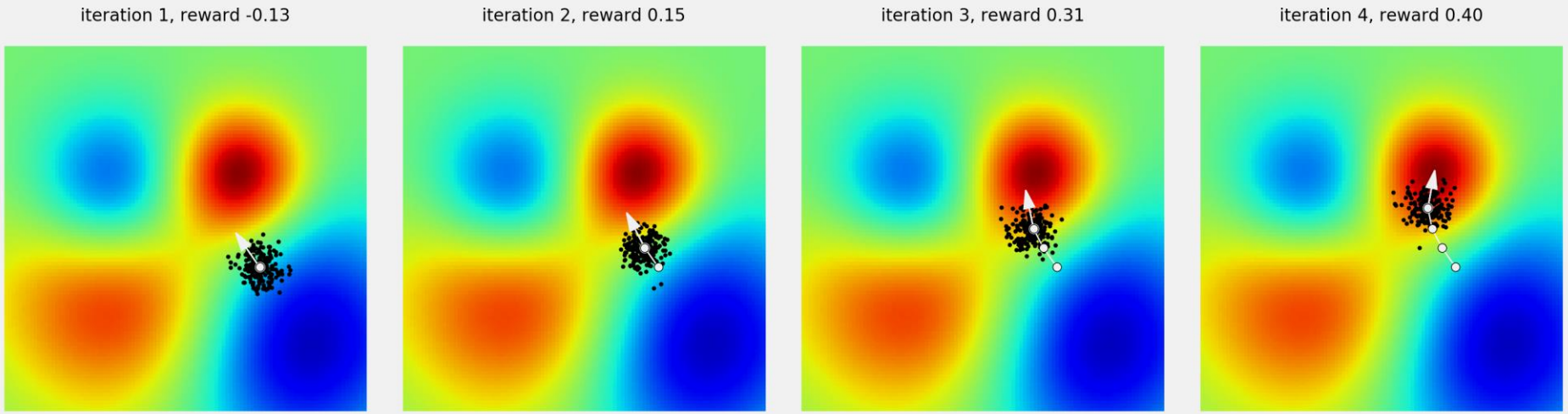
- w - параметры модели
- f(w) возвращает скаляр "награду" (reward)
- о внутреннем устройстве f(w) ничего не известно (black-box)
- возможно только вычисление или оценка f(w) для любого заданного w

Эволюционные стратегии, алгоритм

1. Инициализируем веса w случайными значениями
2. Создаем популяцию из N слегка отличающихся $w_1 \ldots w_N$
3. Вычисляем $f(w_1) \ldots f(w_N)$
4. Вычисляем новый w как взвешенную сумму $w_1 \ldots w_N$, где веса пропорциональны $f(w_1) \ldots f(w_N)$

Эволюционные стратегии, алгоритм

1. Инициализируем веса w случайными значениями
2. Численно оцениваем градиент f(w)
3. Смещаем w в направлении возрастания f(w)



Эволюционные стратегии, алгоритм

```python
# simple example: minimize a quadratic around some solution point
import numpy as np
solution = np.array([0.5, 0.1, -0.3])
def f(w): return -np.sum((w - solution)**2)

npop = 50      # population size
sigma = 0.1    # noise standard deviation
alpha = 0.001  # learning rate
w = np.random.randn(3) # initial guess
for i in range(300):
    N = np.random.randn(npop, 3)
    R = np.zeros(npop)
    for j in range(npop):
        w_try = w + sigma*N[j]
        R[j] = f(w_try)
    A = (R - np.mean(R)) / np.std(R)
    w = w + alpha/(npop*sigma) * np.dot(N.T, A)
```

Эволюционные стратегии, плюсы и минусы

Плюсы:

- Алгоритм прост
- Алгоритм не требует реализации обратного распространения ошибки
- Алгоритм очень эффективно параллелится

Минусы:

- Алгоритм требует большего количества шагов обучения
- Алгоритм требует больше вычислительных ресурсов

ТЕХНОСФЕРА

Спасибо за внимание!

Евгений Некрасов

e.nekrasov@corp.mail.ru