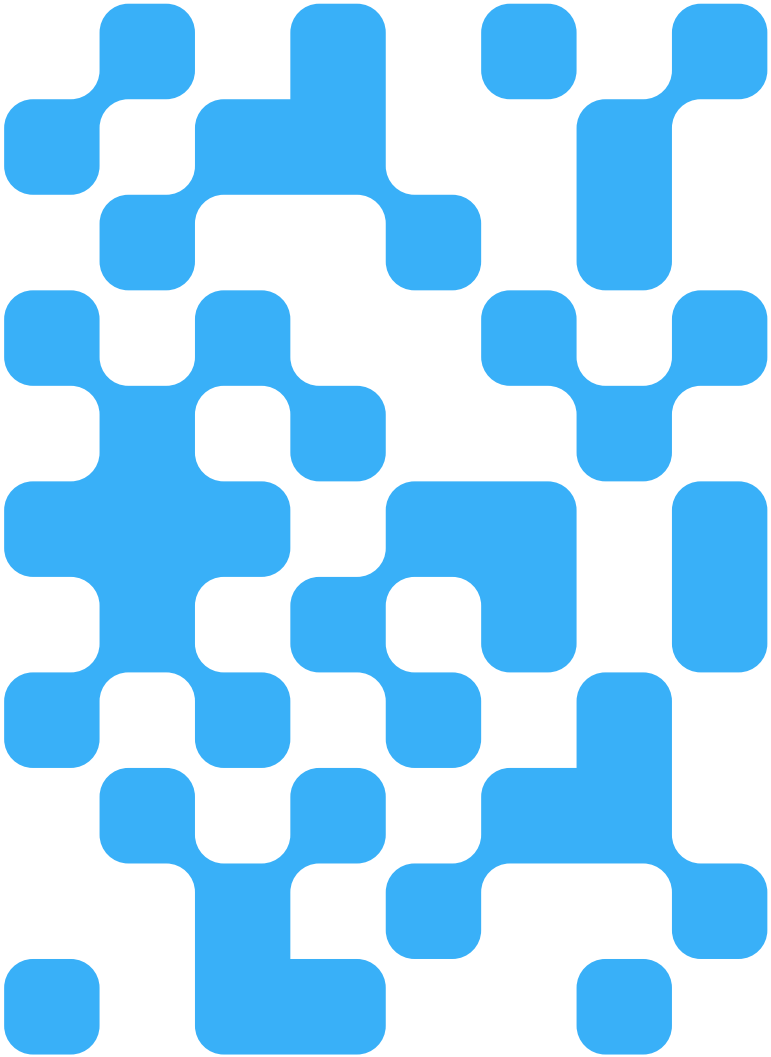
Machine Learning

2024 (ML-2024)

Lecture 12. Reinforcement Learning

by Alexei Valerievich Kornaev, Dr. habil. in Eng. Sc.,
Researcher at the RC for AI, Assoc. Prof. of the Robotics and CV Master's Program, Innopolis University
Researcher at the RC for AI, National RC for Oncology n.a. NN Blohin
Professor at the Dept. of Mechatronics, Mechanics, and Robotics, Orel State University

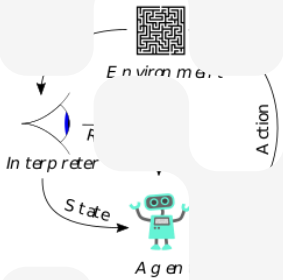


Agenda

I. INTRODUCTION TO REINFORCEMENT LEARNING

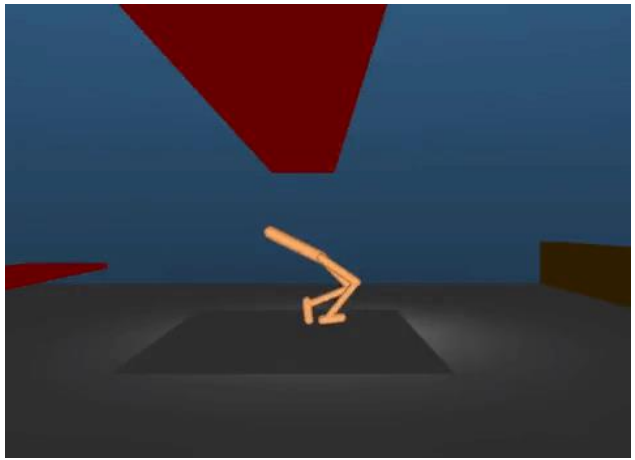
II. POLICY OPTIMIZATION

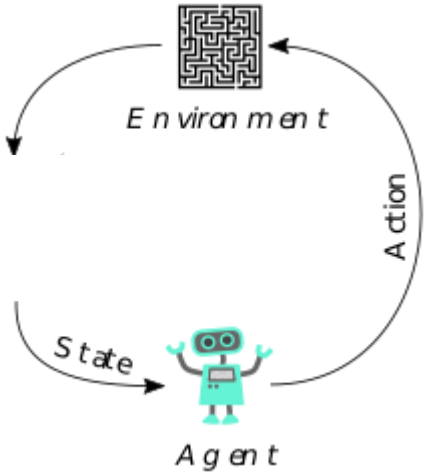
III. Q-LEARNING



Reinforcement learning

Control or decision process: at each time step, the controller (agent) receives feedback from the system (environment) in the form of a state signal, and takes an action in response. We supposed that current state completely characterizes the state of the system (Markov decision process).
The main problem is that the correct actions are unknown sometimes.
The main idea is to learn agent after the event, giving him higher reward for the better actions.

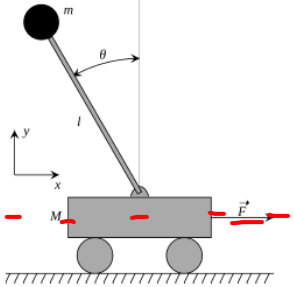




RL scenario

Reinforcement learning

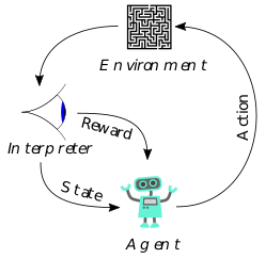
Control or decision process: at each time step, the controller (agent) receives feedback from the system (environment) in the form of a state signal, and takes an action in response. We supposed that current state completely characterizes the state of the system (Markov decision process).
The main problem is that the correct actions are unknown sometimes.
The main idea is to learn agent after the event, giving him higher reward for the better actions.



Objective: Balance a pole on top of a movable cart

State: angle, angular speed, position, horizontal velocity
Action: horizontal force applied on the cart
Reward: 1 at each time step if the pole is upright

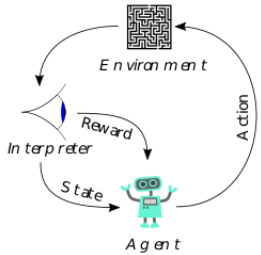
DRL course in Stanford University School of Engineering



RL scenario

Reinforcement learning

Характеристики модели RL:
S_t (s_t) – состояние среды (state) в момент времени t;
A_t (a_t) – действие агента (action) в момент времени t;
r_t – награда агента (reward) в момент времени t;
$g_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$ - будущая награда (return);
γ - дисконт (discount).



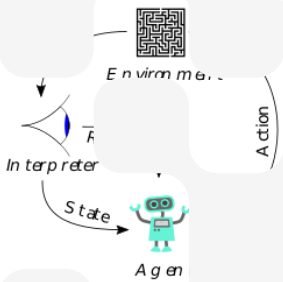
RL scenario

Agenda

I. INTRODUCTION TO REINFORCEMENT LEARNING

II. POLICY OPTIMIZATION

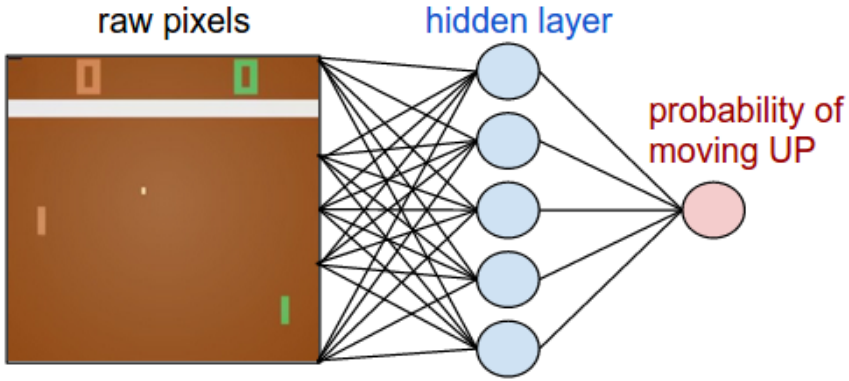
III. Q-LEARNING



Reinforcement learning

Пример простейшей реализации алгоритма PG: задача игры в пинг-понг.
State: матрица S_t размером [210, 160, 3], составленная из значений цветов пикселей цветного изображения состояния игры (точнее разница матриц для двух соседних моментов времени / difference frames). **Action:** бинарный выбор движения ракетки $a_t = \{0,1\}$ вниз или вверх (0 – «DOWN», 1 – «UP»). **Reward:** положительная $r_t = +1$, если соперник пропустил мяч; отрицательная $r_t = -1$, если агент пропустил.

$S_t \quad \rightarrow \quad 0 < p_t < 1, a_t = \{0,1\}.$



$i = 1 \ldots m$ #frame

$$L\left(\Theta^{(k)}\right) = \sum_{i=1}^{m} r_i \ln\left(p_i(a_i|s_i)\right) \Rightarrow max.$$

Andrej Karpathy blog: Pong from Pixels

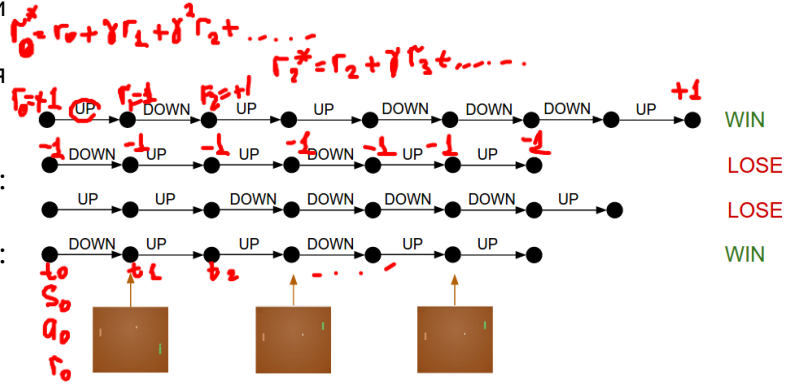
Какие действия в игре правильные?

Reinforcement learning

State: матрица S_t размером [210, 160, 3], составленная из значений цветов пикселей цветного изображения состояния игры (точнее разница матриц для двух соседних моментов времени / difference frames). **Action:** бинарный выбор движения ракетки $a_t = \{0,1\}$ вниз или вверх (0 – «DOWN», 1 – «UP»). **Reward:** положительная $r_t = +1$, если соперник пропустил мяч; отрицательная $r_t = -1$, если агент пропустил.

Алгоритм обучения / Training algorithm.

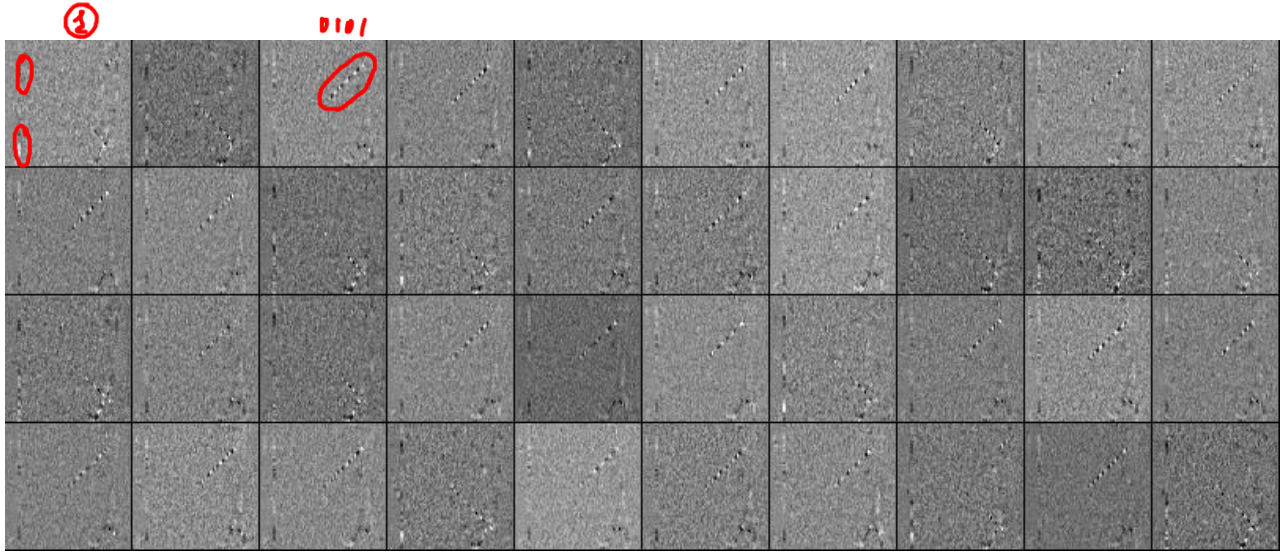
1. Случайным образом назначаются веса $\Theta^{(k)}$ ИНС.
2. Выполняется прогон (rollout) из 100 игровых партий. Все действия выигранных партий считаются правильными и поощряются наградой $r_t = +1$. И наоборот, для всех действий проигранных партий награда $r_t = -1$.
3. Для каждого действия партии рассчитывается дисконтированная награда: $r_t^* = r_{t+j}\gamma^j$ $(j = 0,1,\dots)$.
4. Формируются данные (dataset) прогона: S_i, p_i, r_i^*.
5. Рассчитывается для прогона функция качества: $L\left(\Theta^{(k)}\right) = \sum_{i=1}^{m} r_i^* \ln(p_i)$.
6. Рассчитывается градиент $\nabla L\left(\Theta^{(k)}\right)$ и новые значения весов: $\theta_{ij}^{(k)} = \theta_{ij}^{(k)} + \alpha \frac{\partial L}{\partial \theta_{ij}^{(k)}}$.
7. Повторяются пп. 2-6 до выполнения нек. условий.



Andrej Karpathy blog: Pong from Pixels

Reinforcement learning

State: матрица S_t размером [210, 160, 3], составленная из значений цветов пикселей цветного изображения состояния игры (точнее разница матриц для двух соседних моментов времени / difference frames). **Action:** бинарный выбор движения ракетки $a_t = \{0,1\}$ вниз или вверх (0 – «DOWN», 1 – «UP»). **Reward:** положительная $r_t = +1$, если соперник пропустил мяч; отрицательная $r_t = -1$, если агент пропустил.



Изображения весов 40 нейронов (из 200) скрытого слоя, которые визуализируют траекторию движения шарика. Белые пиксели означают положительные значения весов, черные – отрицательные.

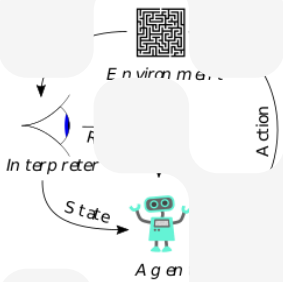
Andrej Karpathy blog: Pong from Pixels

Agenda

I. INTRODUCTION TO REINFORCEMENT LEARNING

II. POLICY OPTIMIZATION

III. Q-LEARNING



Reinforcement learning: Q-learning

Основная идея: на основании результатов исследования окружающей среды (environment) обучить модель (critic), которая при данных S_t , A_t предсказывает будущую награду g_t для любого возможного действия A_{t+1}. Тогда агенту (agent) при данных S_t, A_t из множества возможных дальнейших действий A_{t+1}, следует предпринимать то, которое приведет к наибольшей будущей награде g_t.

Формализация. В каждый момент времени t функция будущей награды имеет вид:

$$q_t(S_t, A_t) = r_t + \gamma \quad r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{t^{\bowtie}-1} r_{t^{\bowtie}}, \tag{1}$$

где γ – дисконт, $0 < \gamma \leq 1$, $t^{\bowtie}$- шаг по времени при достижении конечного состояния (terminal state) .

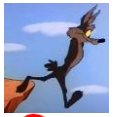
Функцию (1) можно представить в виде:

$$q_t(S_t, A_t) = r_t + \gamma\left(r_{t+1} + \gamma \quad r_{t+2} + \cdots + \gamma^{t^{\bowtie}-2} r_{t^{\bowtie}}\right) = r_t + \gamma\, q_{t+1}(S_{t+1}, A_{t+1}). \tag{2}$$

Тогда наилучшая действие q_t^* в момент времени t описывается уравнением Беллмана:

$$q_t^*(S_t, A_t) = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]\,. \tag{3}$$

Reinforcement learning: Q-learning

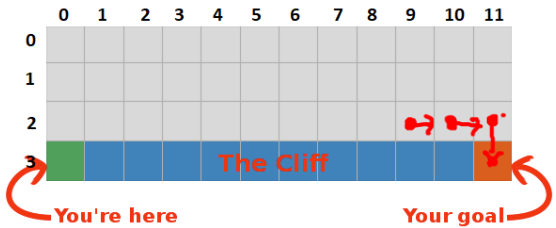


Формализация. Наилучшее действие q_t^* в момент времени t описывается уравнением Беллмана:

$$q_t^*(S_t, A_t) = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]. \quad (3)$$

Пример. Задача о прогулке по скале: необходимо найти кратчайший путь до цели, не упав с обрыва [https://habr.com/ru/post/443240/]. Награда за обычный шаг -1, за достижение цели +0, за срыв с обрыва -100. Дисконт 0.9.

$S = [\![s^{(i,j)}]\!]$, $A = [\![a^{(k)}]\!] \rightarrow Q = [\![q^{(i,j,k)}]\!]$. Q-?



Environment

	$a^{(1)}$ ↑	$a^{(2)}$ ↓	$a^{(3)}$ →	$a^{(4)}$ ←
$s^{(0,0)}$				
$s^{(0,1)}$				
…				
$s^{(2,10)}$	<-1	<-1	-1	<-1
$s^{(2,11)}$	<-1	<0	<-1	<-1
…				
$s^{(3,11)}$	0	0	0	0

The Q table aims to optimal

$q(S[2,11], \text{down}) = +0 + \gamma^{0.9} \max[0,0,0,0] = 0$

Алгоритм заполнения всей таблицы Q пока не ясен, но для оценки наград в обратном направлении от конечного состояния можно воспользоваться уравнением Беллмана (3):

$q_t^{*(2,11,2)} = 0 + 0.9 \max_{a_{t+1}}[0,0,0,0] = 0;$

$q_t^{*(2,10,3)} = -1 + 0.9 \max_{a_{t+1}}[<-1, \mathbf{0}, <-1, <-1] = -1;$

$q_t^{*(2,9,3)} = -1 + 0.9 \max_{a_{t+1}}[<-1, <-1, \mathbf{-1}, <-1] = -1.9;$

…

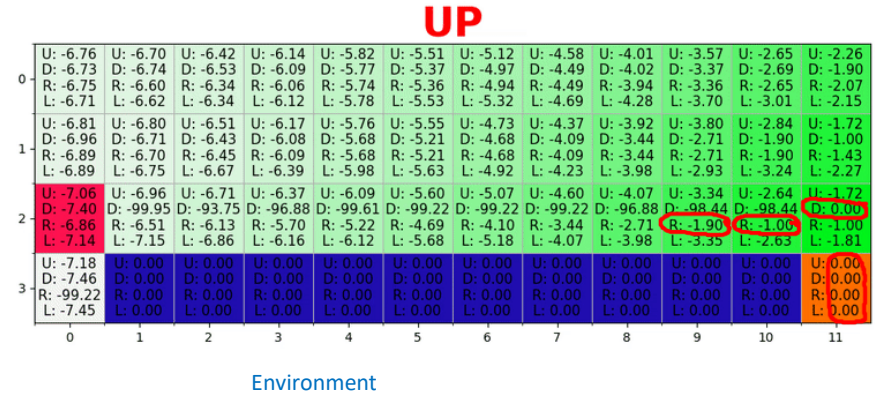
Reinforcement learning: Q-learning

Формализация. Наилучшее действие q_t^* в момент времени t описывается уравнением Беллмана:

$$q_t^*(S_t, A_t) = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]. \quad (3)$$

Пример. Задача о прогулке по скале: необходимо найти кратчайший путь до цели, не упав с обрыва [https://habr.com/ru/post/443240/]. Награда за обычный шаг -1, за достижение цели +0, за срыв с обрыва -100. Дисконт 0.9.

$S = [\![s^{(i,j)}]\!]$, $A = [\![a^{(k)}]\!] \rightarrow Q = [\![q^{(i,j,k)}]\!]$. Q-?



Environment

Алгоритм заполнения всей таблицы Q пока не ясен, но для оценки наград в обратном направлении от конечного состояния можно воспользоваться уравнением Беллмана (3):

$$q_t^{*(2,11,2)} = 0 + 0.9 \max_{a_{t+1}} [0,0,0,0] = 0;$$

$$q_t^{*(2,10,3)} = -1 + 0.9 \max_{a_{t+1}} [< -1, \mathbf{0}, < -1, < -1] = -1;$$

$$q_t^{*(2,9,3)} = -1 + 0.9 \max_{a_{t+1}} [< -1, < -1, -\mathbf{1}, < -1] = -1.9;$$

…

Reinforcement learning: Q-learning

Формализация. Наилучшее действие q_t^* в момент времени t описывается уравнением Беллмана:

$$q_t^*(S_t, A_t) = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]. \quad (3)$$

Процесс накопления опыта состоит в проигрывании эпизодов. В процессе обучения необходимо достичь минимизации ошибки между обучаемой функцией $Q(S, A)$ и оптимальной:

$$Q(S, A) - Q^*(S, A) \Rightarrow \min.$$

Тогда в результате обучения агент будет способен производить действия с максимальной наградой.

Алгоритм обучения [MATLAB Documentation]/ Training algorithm. Инициализировать $Q(S, A)$ случайными значениями или нулями. Задать гиперпараметры: вероятность ϵ, скорость обучения α.
Для каждого эпизода обучения:

1. Получить данные о начальном состоянии S_t $(t = 1)$
2. Повторять для каждого шага t до достижения terminal state:
   2.1 Для текущего состояния S_t выбрать случайное действие A_t с вероятностью ϵ (может уменьшаться в процессе обучения), иначе действие, для которого значение наибольшее: $A_t = \max_A [Q_t(S_t, A_t)]$
   2.2 Выполнить действие A_t, получить награду r_t и данные о новом состоянии S_{t+1}.
   2.3 Если новое состояние S_{t+1} терминальное, то установить значение целевой функции $y_t = r_t$. Иначе $y_t = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]$
   2.4 Обновить компоненту матрицы $Q(S, A)$:

$$q(S_t, A_t) = q(S_t, A_t) + \propto [y_t - q(S_t, A_t)].$$

Robotics – DDPG

Reinforcement learning: Q-learning

Алгоритм обучения. Инициализировать $Q(S,A)$ случайными значениями или нулями. Задать гиперпараметры: вероятность ϵ, скорость обучения α.

Для каждого эпизода обучения:

1. Получить данные о начальном состоянии S_t $(t=1)$
2. Повторять для каждого шага t до достижения terminal state:

2.1 Для текущего состояния S_t выбрать случайное действие A_t с вероятностью ϵ (может уменьшаться в процессе обучения), иначе действие, для которого значение наибольшее: $A_t = \max_A [Q_t(S_t, A_t)]$

2.2 Выполнить действие A_t, получить награду r_t и данные о новом состоянии S_{t+1}.

2.3 Если новое состояние S_{t+1} терминальное, то установить значение целевой функции $y_t = r_t$. Иначе $y_t = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]$

2.4 Обновить компоненту матрицы $Q(S,A)$:

$q(S_t, A_t) = q(S_t, A_t) + \propto [y_t - q(S_t, A_t)]$.

	$a^{(1)}$ ↑	$a^{(2)}$ ↓	$a^{(3)}$ →	$a^{(4)}$ ←
$s^{(0,0)}$	0	0	0	0
$s^{(0,1)}$	0	0	0	0
…	0	0	0	0
$s^{(2,0)}$	0	0	0	0
$s^{(2,1)}$	0	0	0	0
$s^{(2,2)}$	0	0	0	0
…	0	0	0	0
$s^{(3,0)}$	0	0	0	0
$s^{(3,1)}$	0	0	0	0
$s^{(3,2)}$	0	0	0	0
…	0	0	0	0
$s^{(3,11)}$	0	0	0	0

Реализация алгоритма.

Инициализация $Q(S,A)$ нулями. Задание: $\epsilon = 1$, $\alpha = 0.1$, $\gamma = 0.9$.

Эпизод 1. $\epsilon = 1$, тогда все действия 1 эпизода случайные.

1. $t = 1$. $\epsilon = 1$ (все действия случайные), $s_t = s^{(3,0)}$.

2.1 Выбор сл. действия $a_t =$

2.2 Вып. действия $a_t =$, награда $r_t =$, $s_{t+1} = s^{(2,0)}$.

2.3 Расчет цел. функц. $y_t =$

2.4 Обновление $q^{(3,0,1)} =$

$t = 2$. $\epsilon = 1$ (все действия случайные), $s_t = s^{(2,0)}$.

2.1 Выбор сл. действия $a_t =$

2.2 Вып. действия $a_t =$, награда $r_t =$, $s_{t+1} = s^{(2,1)}$.

2.3 Расчет цел. функц. $y_t =$

2.4 Обновление $q^{(2,0,3)} =$

$t = 3$. $\epsilon = 1$ (все действия случайные), $s_t = s^{(2,1)}$.

2.1 Выбор сл. действия $a_t =$.

2.2 Вып. действия $a_t =$, награда $r_t =$.

2.3 Расчет цел. функц. $y_t =$

2.4 Обновление $q^{(2,1,2)} =$

….

Thank you for your attention!

a.kornaev@innopolis.ru, @avkornaev

Atari Deep. Q-learning

$J \frac{d\omega}{dt} = M \quad ; \quad t = 0 \quad \varphi = 0$

$M = ?$

$\varphi = \frac{\pi}{6}$