Обучение с подкреплением

Lecture 2 - Deep Reinforcement Learning

Власов Кирилл Вячеславович

2021


Agent


Agent


Environment
[adult swim]

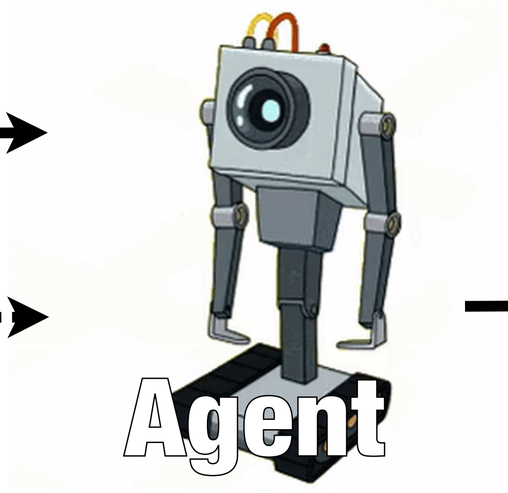
State s_t

State s_t

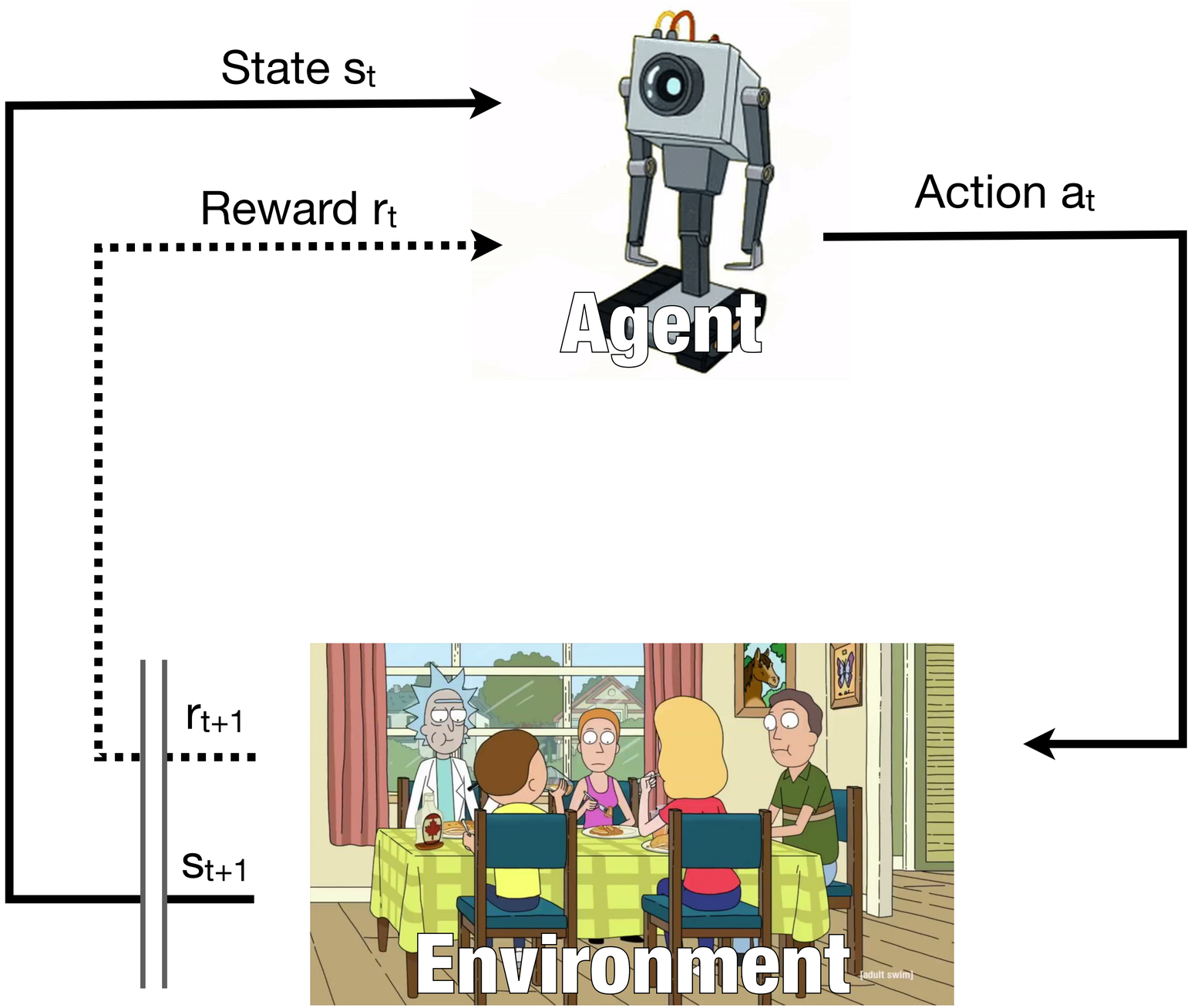
Reward r_t

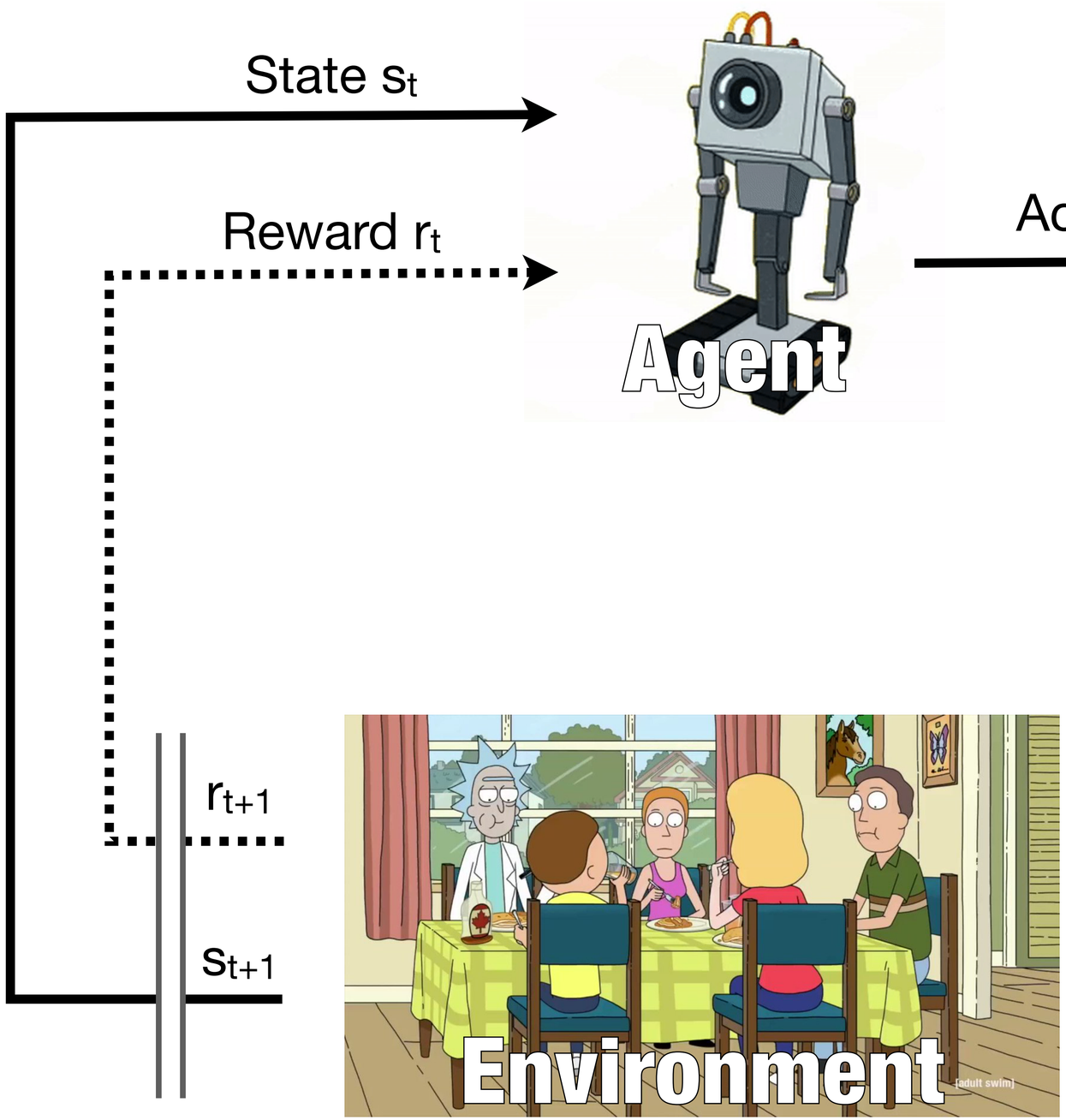
State s_t

Reward r_t



Action a_t


State s_t
Reward r_t
Action a_t
Agent
r_{t+1}
s_{t+1}
Environment
[adult swim]



Среда может:

- быть не детерминированной (выдавать каждый раз разный стейт)
- внутреннее состояние (которое явно можно не наблюдать в стейте)

$$\sum_{t=1}^{n} r_t \longrightarrow max$$

$$\sum_{t=1}^{n} r_t \longrightarrow max$$

$p(a \mid s)$ policy function

$v(s)$ value function

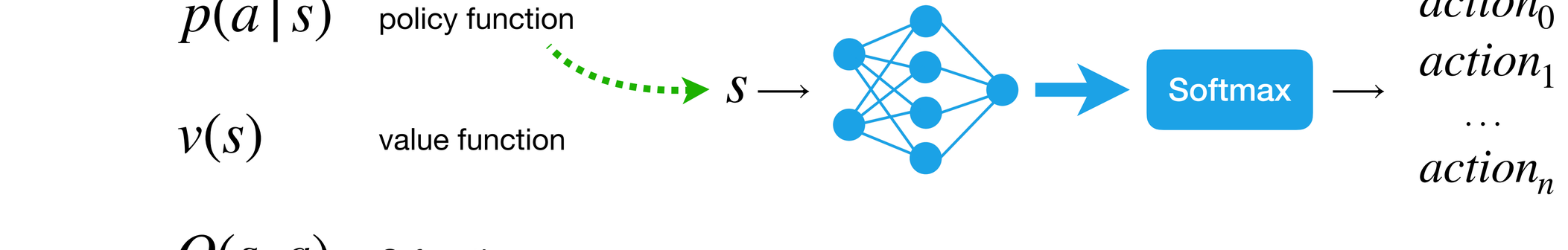
$Q(s, a)$ Q-function

$$\sum_{t=1}^{n} r_t \longrightarrow max$$

$p(a \mid s)$ policy function

$v(s)$ value function

$Q(s, a)$ Q-function

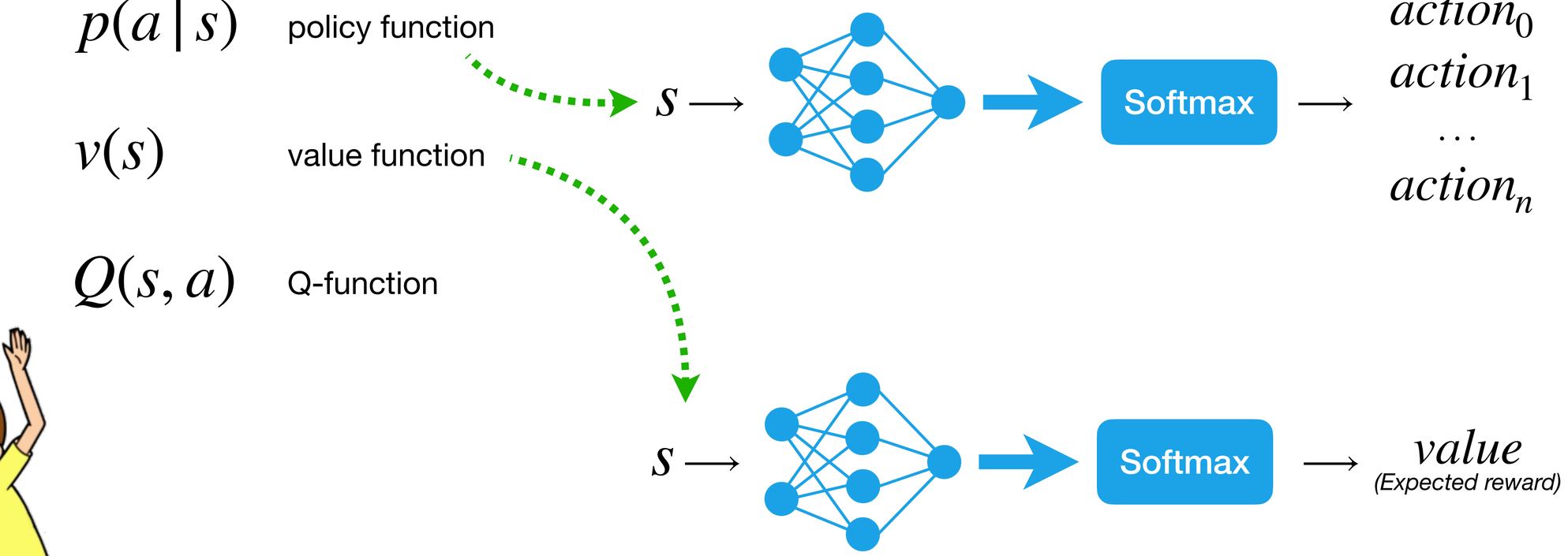


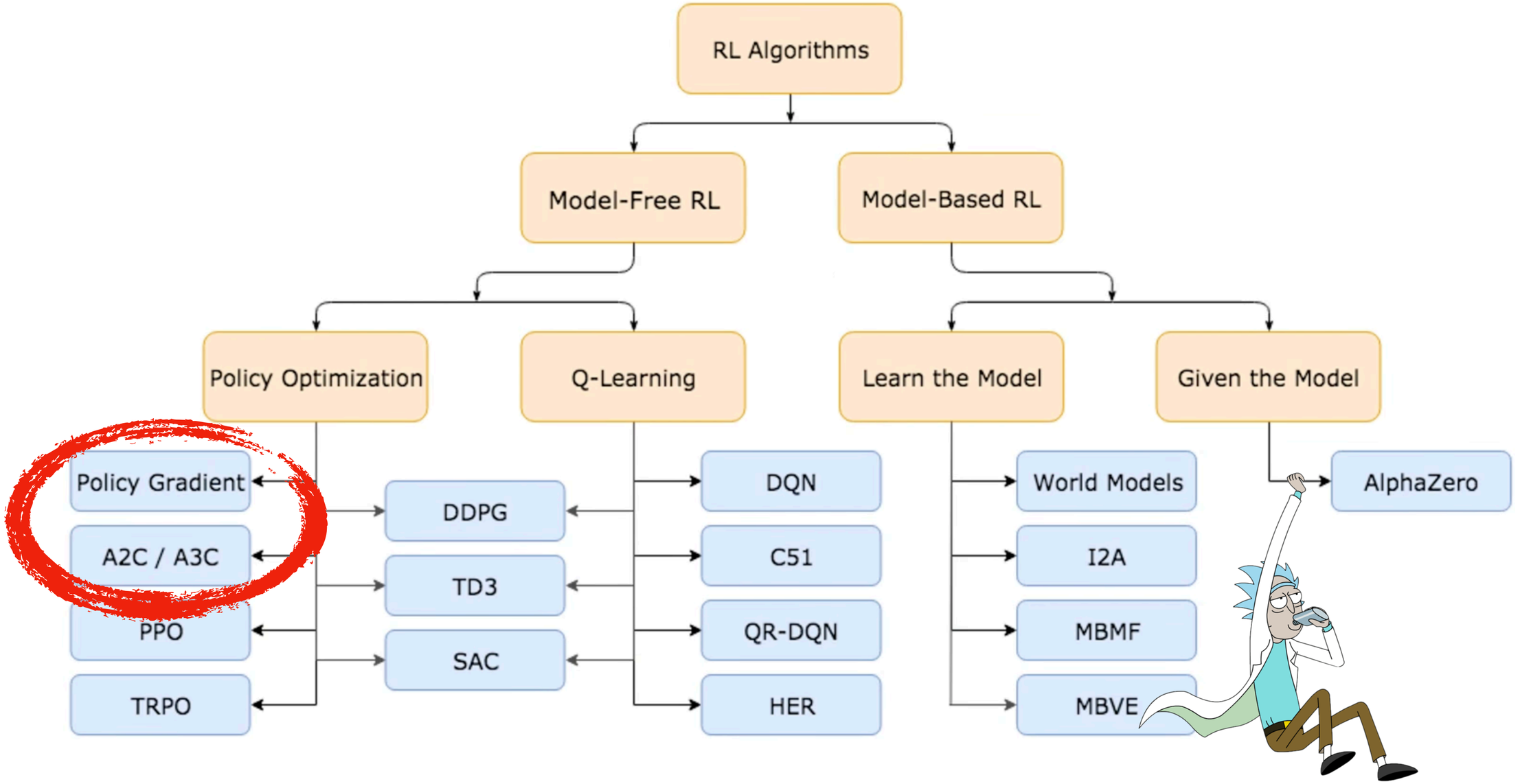
$$\sum_{t=1}^{n} r_t \longrightarrow max$$

$p(a \mid s)$ policy function

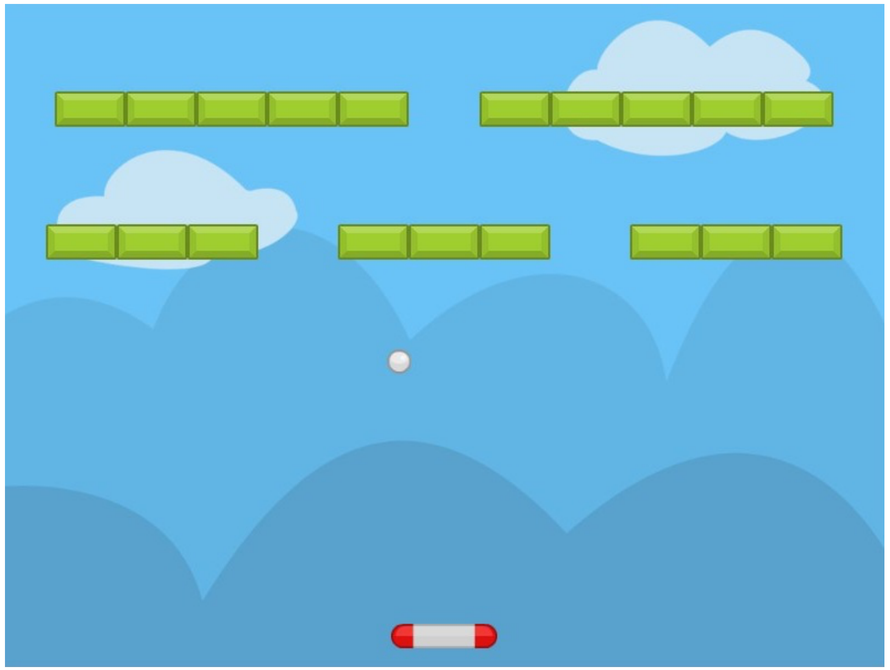
$v(s)$ value function

$Q(s, a)$ Q-function

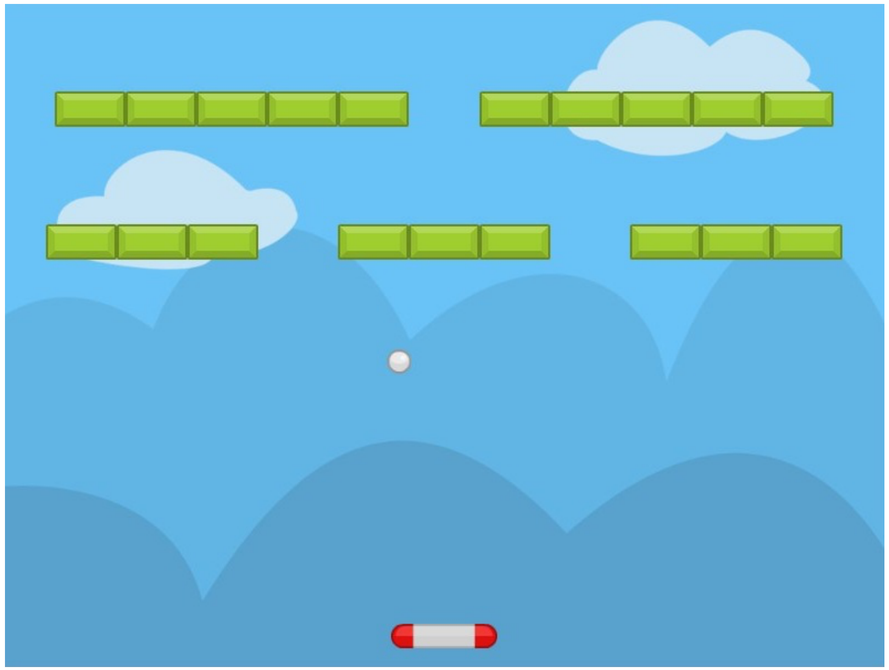


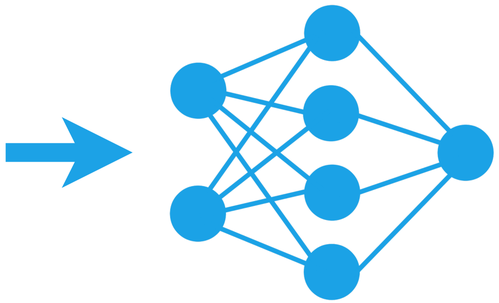

RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

Policy Gradients

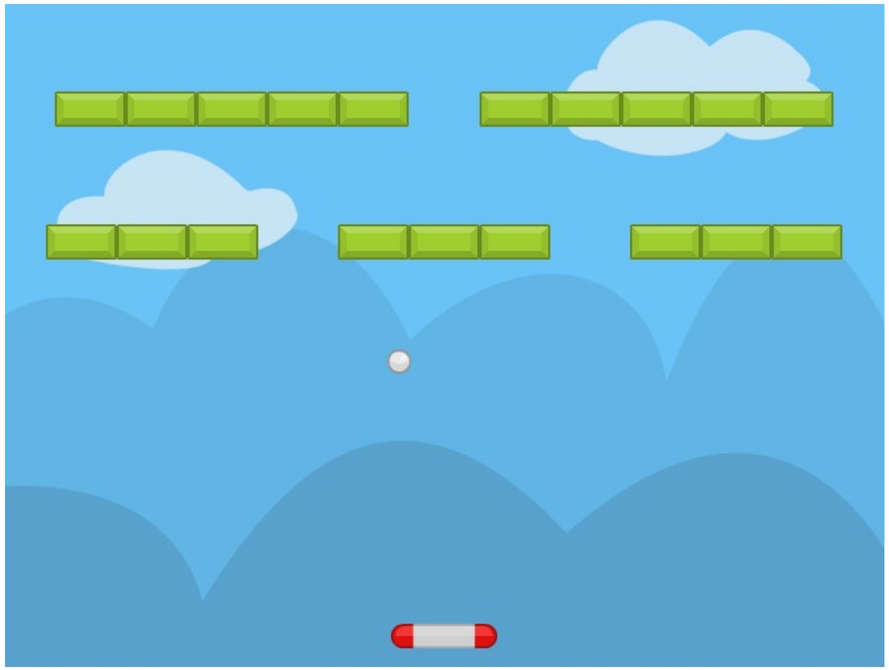


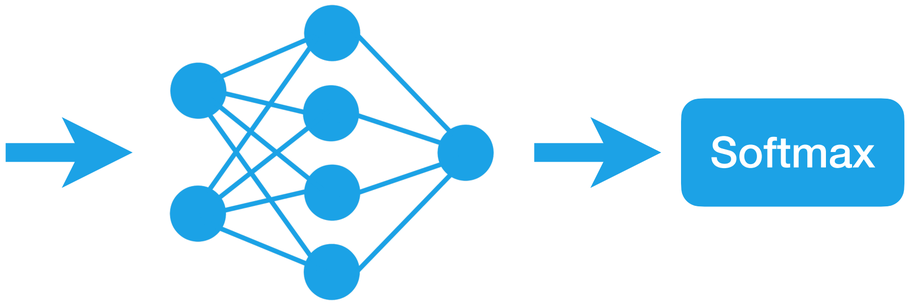
Policy Gradients



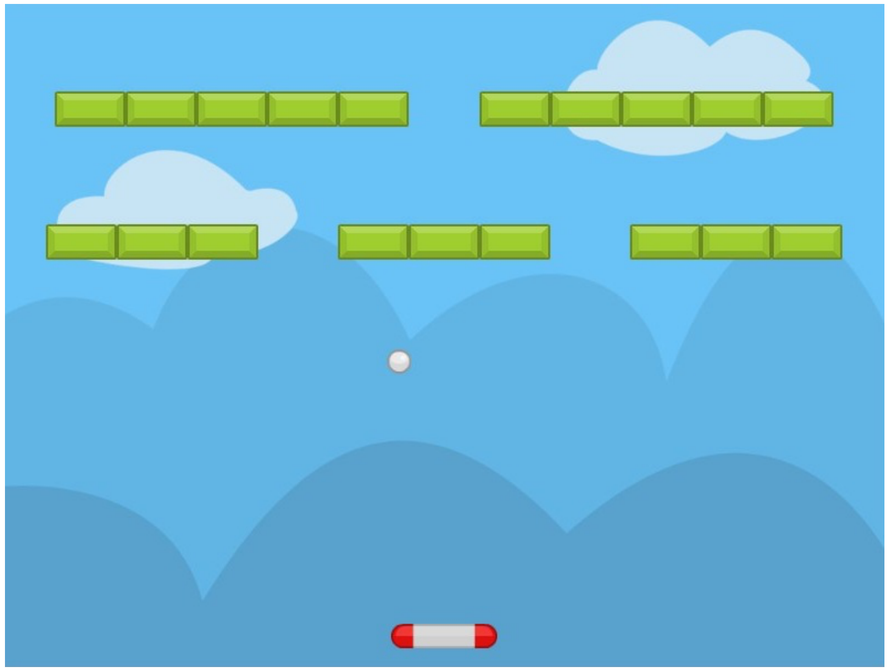


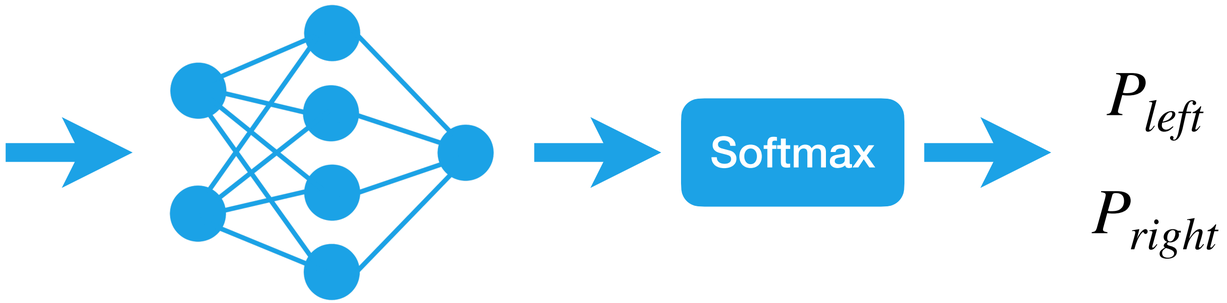
Policy Gradients





Policy Gradients





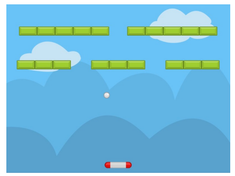
Отметим:

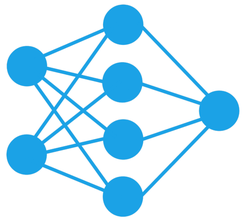
- Несколько эпизодов игры
- Каждый эпизод содержит разное количество ходов
- Мы можем оценить реворд (выиграл или проиграл) только после конца эпизода

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i \,|\, x_i)$$



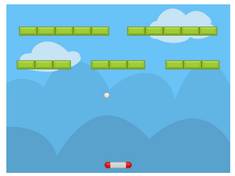


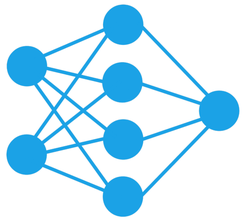
Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i \,|\, x_i)$$

Forward pass



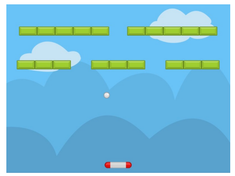


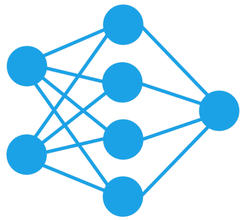
Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i \,|\, x_i)$$

Forward pass





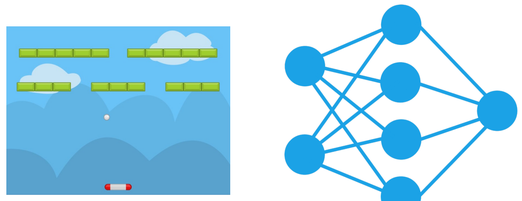
$$lnP_{left} = -1.2$$

$$lnP_{right} = -0.36$$

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i \,|\, x_i)$$



Forward pass

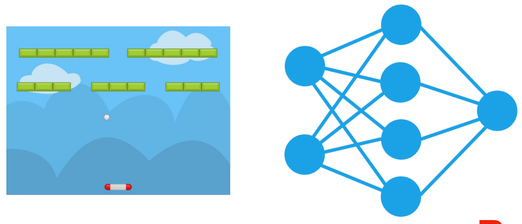
if left is correct

		Gradients
$lnP_{left} = -1.2$	⟶	1.0
$lnP_{right} = -0.36$	⟶	0

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i \,|\, x_i)$$



Forward pass

if left is correct

Gradients

$$lnP_{left} = -1.2 \longrightarrow 1.0$$

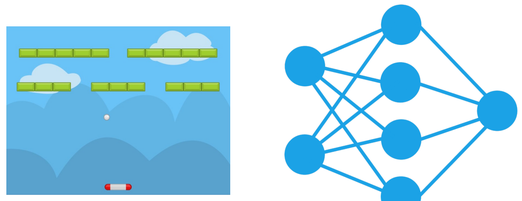
$$lnP_{right} = -0.36 \longrightarrow 0$$

Backward pass

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i \mid x_i)$$



Forward pass

if left is correct

Gradients

$$lnP_{left} = -1.2 \longrightarrow 1.0$$

$$lnP_{right} = -0.36 \longrightarrow 0$$

Backward pass

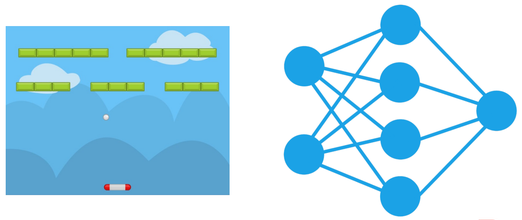
Проблема:

Мы получаем оценку, что действие правильное, только в конце эпизода, а не после каждого шага

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i | x_i)$$



Forward pass

if left is correct

Gradients

$$lnP_{left} = -1.2 \longrightarrow 1.0$$

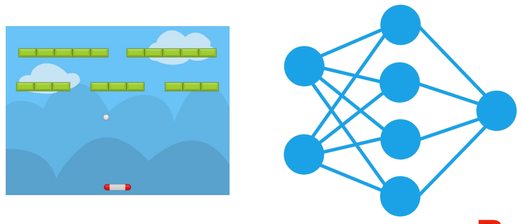
$$lnP_{right} = -0.36 \longrightarrow 0$$

Backward pass

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i \,|\, x_i)$$



Forward pass

$lnP_{left} = -1.2$

$lnP_{right} = -0.36$

if left is correct

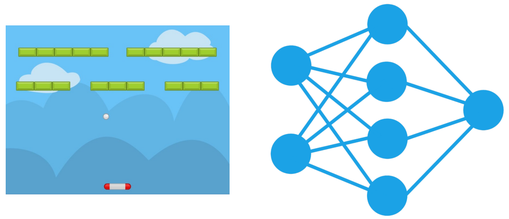
Gradients

1.0

0

Backward pass

Reinforcement learning



Forward pass

$lnP_{left} = -1.2$

$lnP_{right} = -0.36$

Policy Gradients

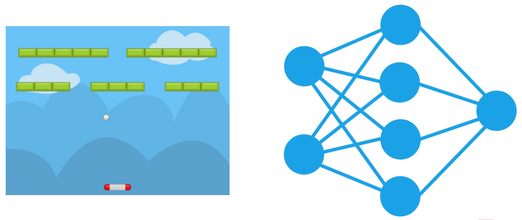
Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i | x_i)$$

if left is correct

Gradients

Forward pass



$$lnP_{left} = -1.2 \longrightarrow 1.0$$

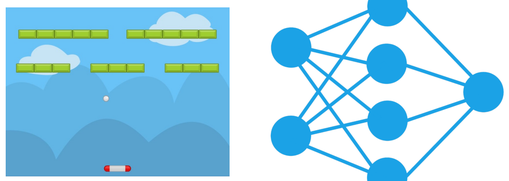
$$lnP_{right} = -0.36 \longrightarrow 0$$

Backward pass

Reinforcement learning

Eventual Reward: -1

Forward pass



$$lnP_{left} = -1.2$$

$$lnP_{right} = -0.36$$

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_{i} y_i \ln p(c = t_i \mid x_i)$$

Forward pass

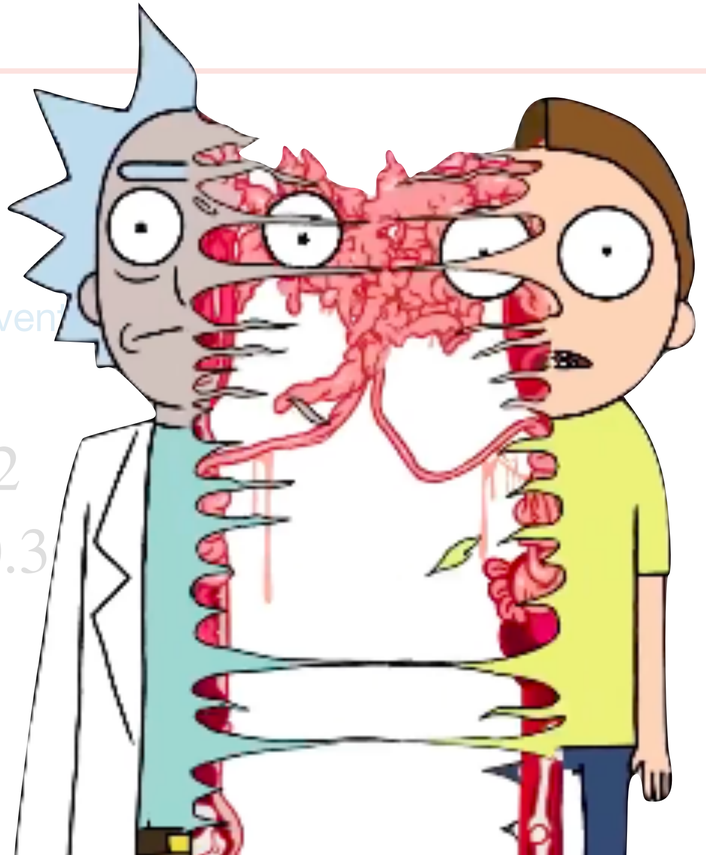
if left is correct

Gradients

$lnP_{left} = -1.2 \longrightarrow 1.0$

$lnP_{right} = -0.36 \longrightarrow 0$

Backward pass



Reinforcement learning

Forward pass

$lnP_{left} = -1.2$

$lnP_{right} = -0.3$

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_i y_i \ln p(c = t_i \mid x_i)$$

Reinforcement learning

$$\mathscr{L}_{PG} = -\sum_i r_i \ln p(c = a_i \mid x_i)$$

Forward pass

if left is correct

Gradients

$lnP_{left} = -1.2 \longrightarrow 1.0$

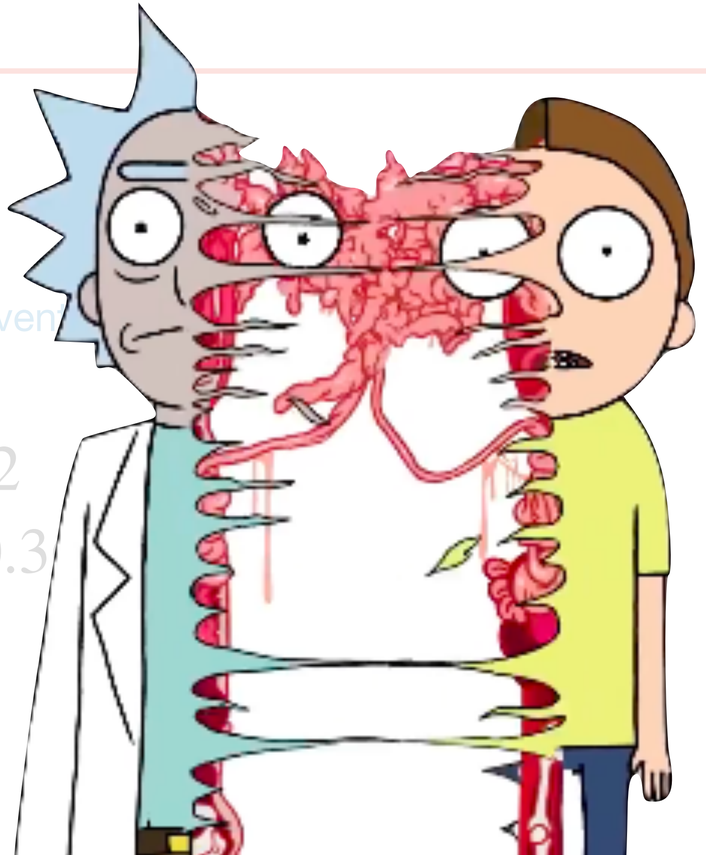
$lnP_{right} = -0.36 \longrightarrow 0$

Backward pass

Forward pass

$lnP_{left} = -1.2$

$lnP_{right} = -0.3$



Policy Gradients

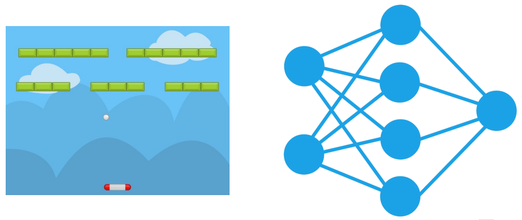
Supervised learning

$$\mathscr{L}_{CE} = -\sum_i y_i \ln p(c = t_i | x_i)$$

if left is correct

Gradients

Forward pass



$$lnP_{left} = -1.2 \longrightarrow 1.0$$

$$lnP_{right} = -0.36 \longrightarrow 0$$

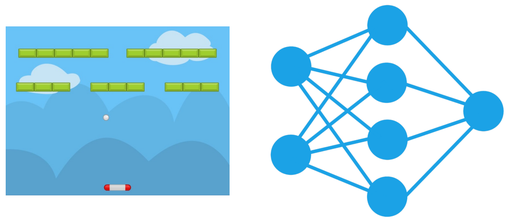
Backward pass

Eventual Reward: -1

Reinforcement learning

$$\mathscr{L}_{PG} = -\sum_i r_i \ln p(c = a_i | x_i)$$

Forward pass



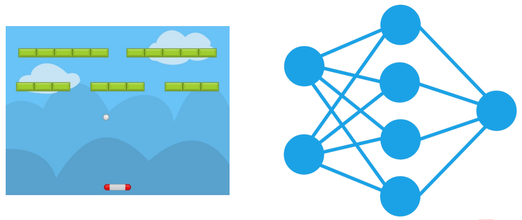
$$lnP_{left} = -1.2$$

$$lnP_{right} = -0.36$$

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_i y_i \ln p(c = t_i | x_i)$$



Forward pass

if left is correct

Gradients

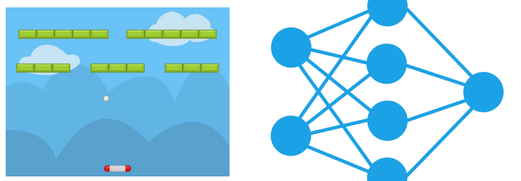
$$lnP_{left} = -1.2 \longrightarrow 1.0$$

$$lnP_{right} = -0.36 \longrightarrow 0$$

Backward pass

Reinforcement learning

$$\mathscr{L}_{PG} = -\sum_i r_i \ln p(c = a_i | x_i)$$



Forward pass

Eventual Reward: -1

Gradients

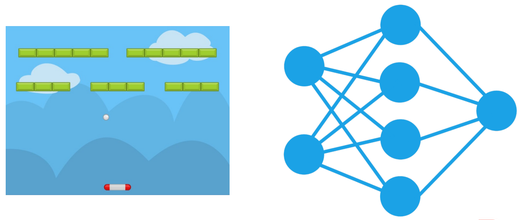
$$lnP_{left} = -1.2 \longrightarrow 0$$

$$lnP_{right} = -0.36 \longrightarrow -1.0$$

Policy Gradients

Supervised learning

$$\mathscr{L}_{CE} = -\sum_i y_i \ln p(c = t_i | x_i)$$



Forward pass

if left is correct

Gradients

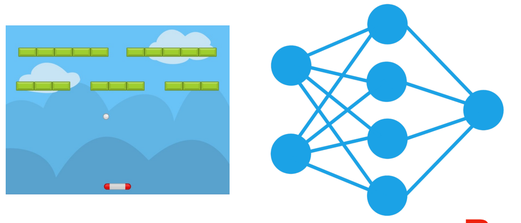
$$lnP_{left} = -1.2 \longrightarrow 1.0$$

$$lnP_{right} = -0.36 \longrightarrow 0$$

Backward pass

Reinforcement learning

$$\mathscr{L}_{PG} = -\sum_i r_i \ln p(c = a_i | x_i)$$



Forward pass

Eventual Reward: -1

Gradients

$$lnP_{left} = -1.2 \longrightarrow 0$$

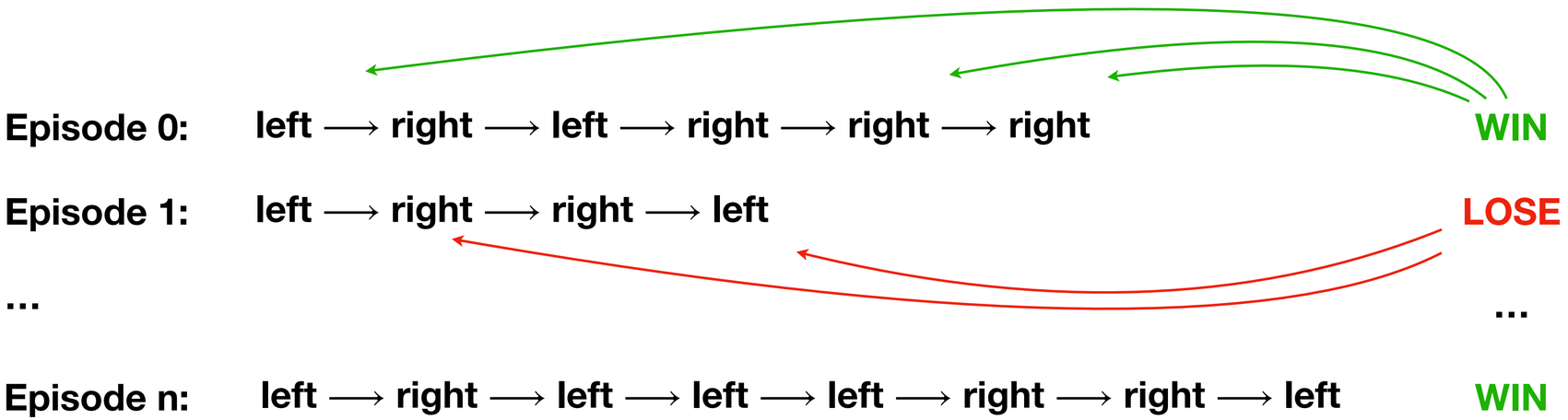
$$lnP_{right} = -0.36 \longrightarrow -1.0$$

Backward pass

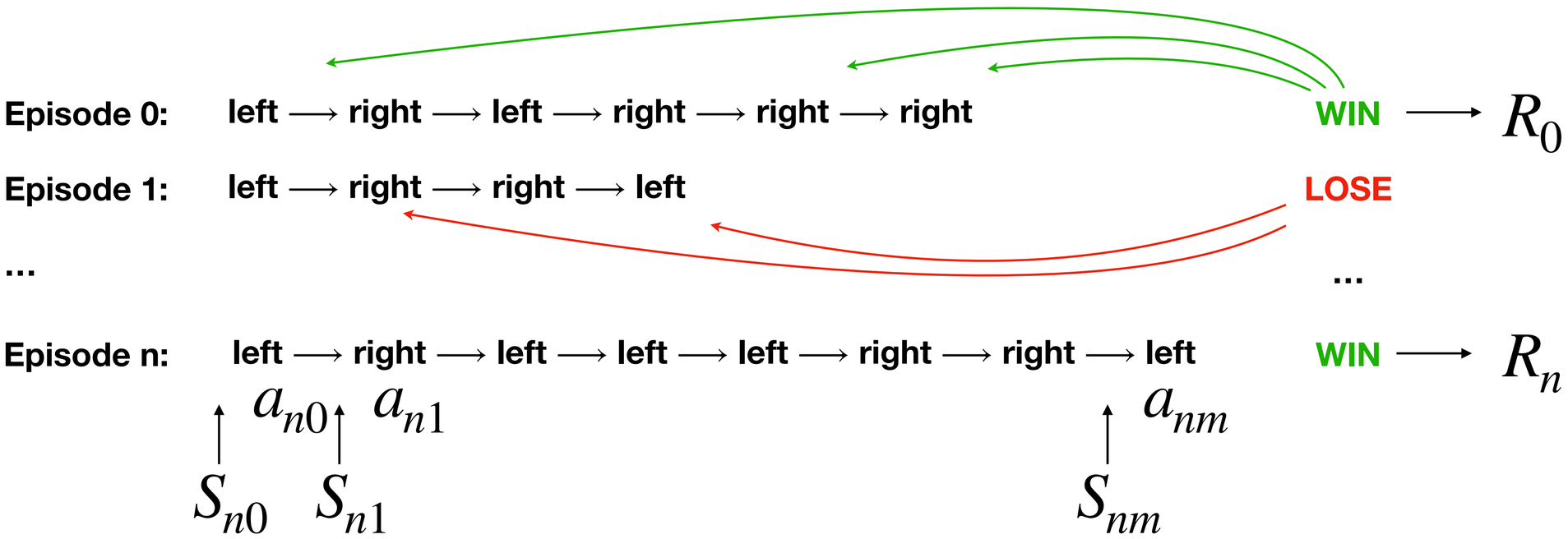
Policy Gradients

Episode 0:	**left ⟶ right ⟶ left ⟶ right ⟶ right ⟶ right**	**WIN**
Episode 1:	**left ⟶ right ⟶ right ⟶ left**	**LOSE**
...		**...**
Episode n:	**left ⟶ right ⟶ left ⟶ left ⟶ left ⟶ right ⟶ right ⟶ left**	**WIN**

Policy Gradients



Policy Gradients



Policy Gradients

Episode 0: left ⟶ right ⟶ left ⟶ right ⟶ right ⟶ right ⟶ **WIN** ⟶ R_0

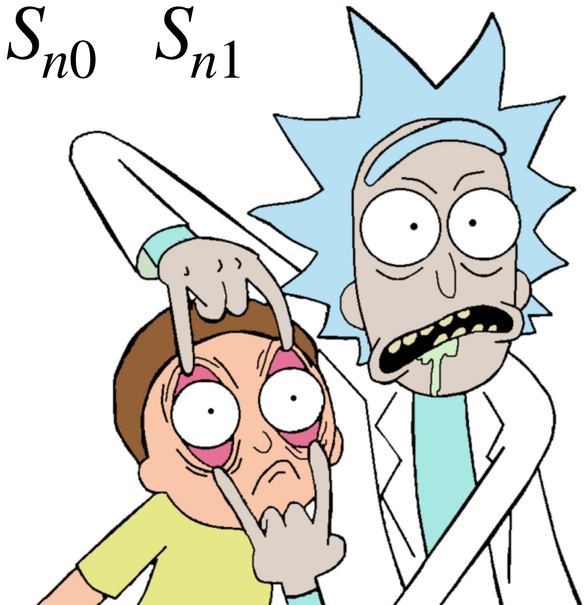
Episode 1: left ⟶ right ⟶ right ⟶ left ⟶ **LOSE**

... ...

Episode n: left ⟶ right ⟶ left ⟶ left ⟶ left ⟶ right ⟶ right ⟶ left ⟶ **WIN** ⟶ R_n

a_{n0} a_{n1} a_{nm}

S_{n0} S_{n1} S_{nm}



Dataset

$\mathcal{X}$	$\mathcal{Y}$	
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

Policy Gradients

Policy Gradients

1. Инициализировали сеть со случайными весами

Policy Gradients

1. Инициализировали сеть со случайными весами
2. Прогнали n эпизодов

Policy Gradients

1. Инициализировали сеть со случайными весами
2. Прогнали n эпизодов
3. Обучили на 1 эпохе

Policy Gradients

1. Инициализировали сеть со случайными весами
2. Прогнали n эпизодов
3. Обучили на 1 эпохе
4. Прогнали n эпизодов

Policy Gradients

1. Инициализировали сеть со случайными весами
2. Прогнали n эпизодов
3. Обучили на 1 эпохе
4. Прогнали n эпизодов
5. Перешли к п. 3

Policy Gradients

1. Инициализировали сеть со случайными весами
2. Прогнали n эпизодов
3. Обучили на 1 эпохе
4. Прогнали n эпизодов*
5. Перешли к п. 3

**и выкинули данные прошлого прогона*

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

$$E[f(a)] = \sum_a p(a) f(a)$$

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

$$E[f(a)] = \sum_a p(a) f(a)$$

$$\nabla_\theta E[f(a)] = \nabla_\theta \sum_a p(a) f(a)$$

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

$$E[f(a)] = \sum_a p(a) f(a)$$

$$\nabla_\theta E[f(a)] = \nabla_\theta \sum_a p(a) f(a)$$

$$= \sum_a \nabla_\theta p(a) f(a)$$

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

$$E[f(a)] = \sum_a p(a) f(a)$$

$$\nabla_\theta E[f(a)] = \nabla_\theta \sum_a p(a) f(a)$$

$$= \sum_a \nabla_\theta p(a) f(a)$$

$$= \sum_a p(a) \left[\frac{\nabla_\theta p(a)}{p(a)} \right] f(a)$$

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

$$E[f(a)] = \sum_a p(a) f(a)$$

$$\nabla_\theta E[f(a)] = \nabla_\theta \sum_a p(a) f(a)$$

$$= \sum_a \nabla_\theta p(a) f(a)$$

$$= \sum_a p(a) \left[\frac{\nabla_\theta p(a)}{p(a)} \right] f(a)$$

$$\frac{1}{z} \nabla_\theta z = \nabla_\theta log(z)$$

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

$$E[f(a)] = \sum_a p(a) f(a)$$

$$\nabla_\theta E[f(a)] = \nabla_\theta \sum_a p(a) f(a)$$

$$= \sum_a \nabla_\theta p(a) f(a)$$

$$= \sum_a p(a) \left[\frac{\nabla_\theta p(a)}{p(a)} \right] f(a)$$

$$= \sum_a p(a) \left[\nabla_\theta \log(p(a)) \right] f(a)$$

$$\frac{1}{z} \nabla_\theta z = \nabla_\theta log(z)$$

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

$$E[f(a)] = \sum_a p(a) f(a)$$

$$\nabla_\theta E[f(a)] = \nabla_\theta \sum_a p(a) f(a)$$

$$= \sum_a \nabla_\theta p(a) f(a)$$

$$= \sum_a p(a) \left[\frac{\nabla_\theta p(a)}{p(a)} \right] f(a)$$

$$= \sum_a p(a) \left[\nabla_\theta \log(p(a)) \right] f(a)$$

$$= E\left[f(a) \nabla_\theta \log(p(a)) \right]$$

$$\frac{1}{z} \nabla_\theta z = \nabla_\theta log(z)$$

Policy Gradients

$p(a \mid s, \theta)$ - policy function

$f(a)$ - reward from environment

$E[f(a)] \longrightarrow \max$

$$E[f(a)] = \sum_a p(a) f(a)$$

$$\nabla_\theta E[f(a)] = \nabla_\theta \sum_a p(a) f(a)$$

$$= \sum_a \nabla_\theta p(a) f(a)$$

$$= \sum_a p(a) \left[\frac{\nabla_\theta p(a)}{p(a)} \right] f(a)$$

$$= \sum_a p(a) \left[\nabla_\theta \log(p(a)) \right] f(a)$$

$$= E\left[f(a) \nabla_\theta \log(p(a)) \right]$$

$$\frac{1}{z} \nabla_\theta z = \nabla_\theta log(z)$$

$$\mathscr{L}_{PG} = -\sum_i r_i \ln p(c = a_i \mid x_i)$$

Tips and Tricks

Tips and Tricks

Dataset

$\mathcal{X}$		$\mathcal{Y}$
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

Tips and Tricks

Dataset

$\mathcal{X}$		$\mathcal{Y}$
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

1 *Discounted rewards*

Tips and Tricks

Dataset

$\mathcal{X}$		$\mathcal{Y}$
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

1

Discounted rewards

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} \qquad 0 < \gamma < 1$$

Tips and Tricks

Dataset

$\mathcal{X}$	$\mathcal{Y}$	
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

1

Discounted rewards

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} \qquad 0 < \gamma < 1$$

2

$r_0 = 9999$

$r_1 = 10002$

$r_2 = 9996$

$r_3 = 10001$

Tips and Tricks

Dataset

$\mathcal{X}$	$\mathcal{Y}$	
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

1 Discounted rewards

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} \qquad 0 < \gamma < 1$$

2 Baseline

$r_0 = 9999$

$r_1 = 10002$

$r_2 = 9996$

$r_3 = 10001$

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} - b$$

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right]$$

Tips and Tricks

Dataset

$\mathcal{X}$		$\mathcal{Y}$
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

1 Discounted rewards

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} \qquad 0 < \gamma < 1$$

2 Baseline

$r_0 = 9999$

$r_1 = 10002$

$r_2 = 9996$

$r_3 = 10001$

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} - b$$

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right]$$

$r_0 = -1$

$r_1 = 2$

$r_2 = -4$

$r_3 = 1$

Tips and Tricks

Dataset

$\mathcal{X}$		$\mathcal{Y}$
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

1 Discounted rewards

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} \qquad 0 < \gamma < 1$$

2 Baseline

$r_0 = 9999$

$r_1 = 10002$

$r_2 = 9996$

$r_3 = 10001$

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} - b$$

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right]$$

$r_0 = -1$

$r_1 = 2$

$r_2 = -4$

$r_3 = 1$

3 Actor-Critic

$r_0 = 1$

$r_1 = 10002$

$r_2 = 9996$

$r_3 = 10001$

Tips and Tricks

Dataset

$\mathcal{X}$	$\mathcal{Y}$	
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

1

Discounted rewards

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} \qquad 0 < \gamma < 1$$

2

Baseline

$r_0 = 9999$

$r_1 = 10002$

$r_2 = 9996$

$r_3 = 10001$

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} - b$$

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right]$$

$r_0 = -1$

$r_1 = 2$

$r_2 = -4$

$r_3 = 1$

3

Actor-Critic

$r_0 = 1$

$r_1 = 10002$

$r_2 = 9996$

$r_3 = 10001$

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} - b$$

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right]$$

Tips and Tricks

Dataset

$\mathcal{X}$	$\mathcal{Y}$	
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

1 Discounted rewards

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} \qquad 0 < \gamma < 1$$

2 Baseline

$$r_0 = 9999$$
$$r_1 = 10002$$
$$r_2 = 9996$$
$$r_3 = 10001$$

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} - b$$

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right]$$

$$r_0 = -1$$
$$r_1 = 2$$
$$r_2 = -4$$
$$r_3 = 1$$

3 Actor-Critic

$$r_0 = 1$$
$$r_1 = 10002$$
$$r_2 = 9996$$
$$r_3 = 10001$$

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} - b$$

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right]$$

Хотим, чтобы он динамически изменялся

Tips and Tricks

Dataset

$\mathcal{X}$	$\mathcal{Y}$	
S_{00}	a_{00}	R_0
S_{01}	a_{01}	R_0
	...	
S_{nm}	a_{nm}	R_n

After action 0: $R_0 = r_0 + r_1 + \ldots + r_n$

After action 1: $R_0 = r_1 + r_2 + \ldots + r_n$

After action 2: $R_0 = r_2 + r_3 + \ldots + r_n$

1 Discounted rewards

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} \qquad 0 < \gamma < 1$$

2 Baseline

$$r_0 = 9999$$
$$r_1 = 10002$$
$$r_2 = 9996$$
$$r_3 = 10001$$

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} - b$$

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right]$$

$$r_0 = -1$$
$$r_1 = 2$$
$$r_2 = -4$$
$$r_3 = 1$$

3 Actor-Critic

$$r_0 = 1$$
$$r_1 = 10002$$
$$r_2 = 9996$$
$$r_3 = 10001$$

$$R_s = \sum_{i=0..n-s} \gamma^i r_{s+i} - b$$

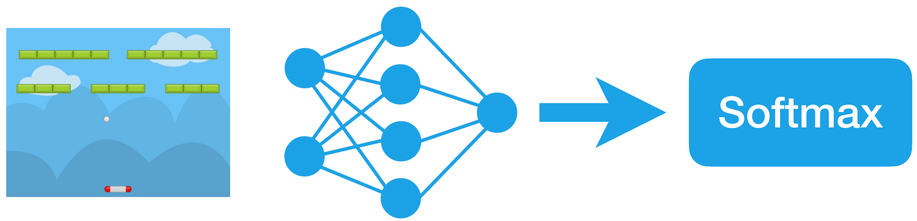
$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right] \rightarrow b = v(s)$$

Actor-Critic

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right] \rightarrow b = v(s)$$

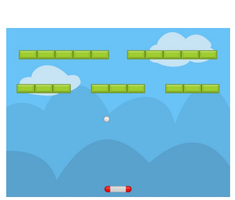
Actor-Critic

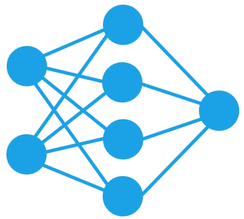
$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right] \rightarrow b = v(s)$$



Actor-Critic

$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right] \rightarrow b = v(s)$$

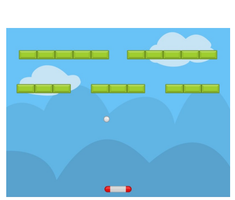


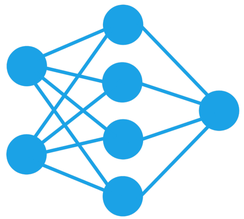


Softmax

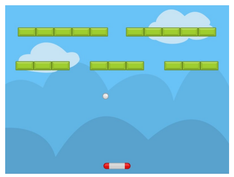
Actor-Critic

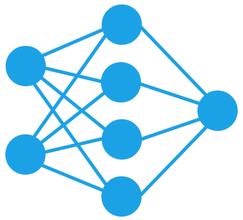
$$b = E\left[\sum_{i=0..n-s} \gamma^i r_{s+i}\right] \rightarrow b = v(s)$$





Softmax



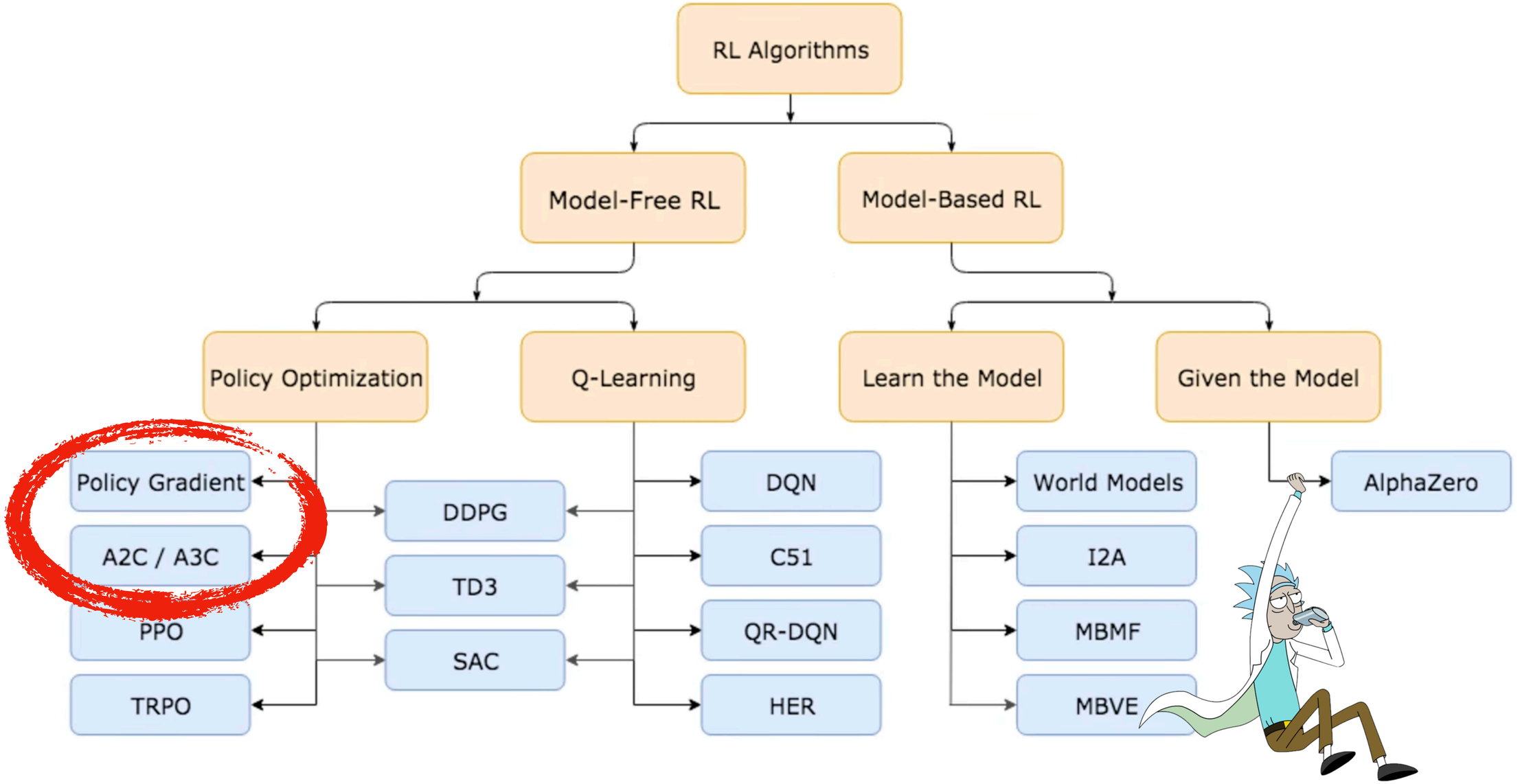


Softmax

$$\ln p(a) \sum_i \left(\gamma^i r_{s+i} - V\right)$$

Linear

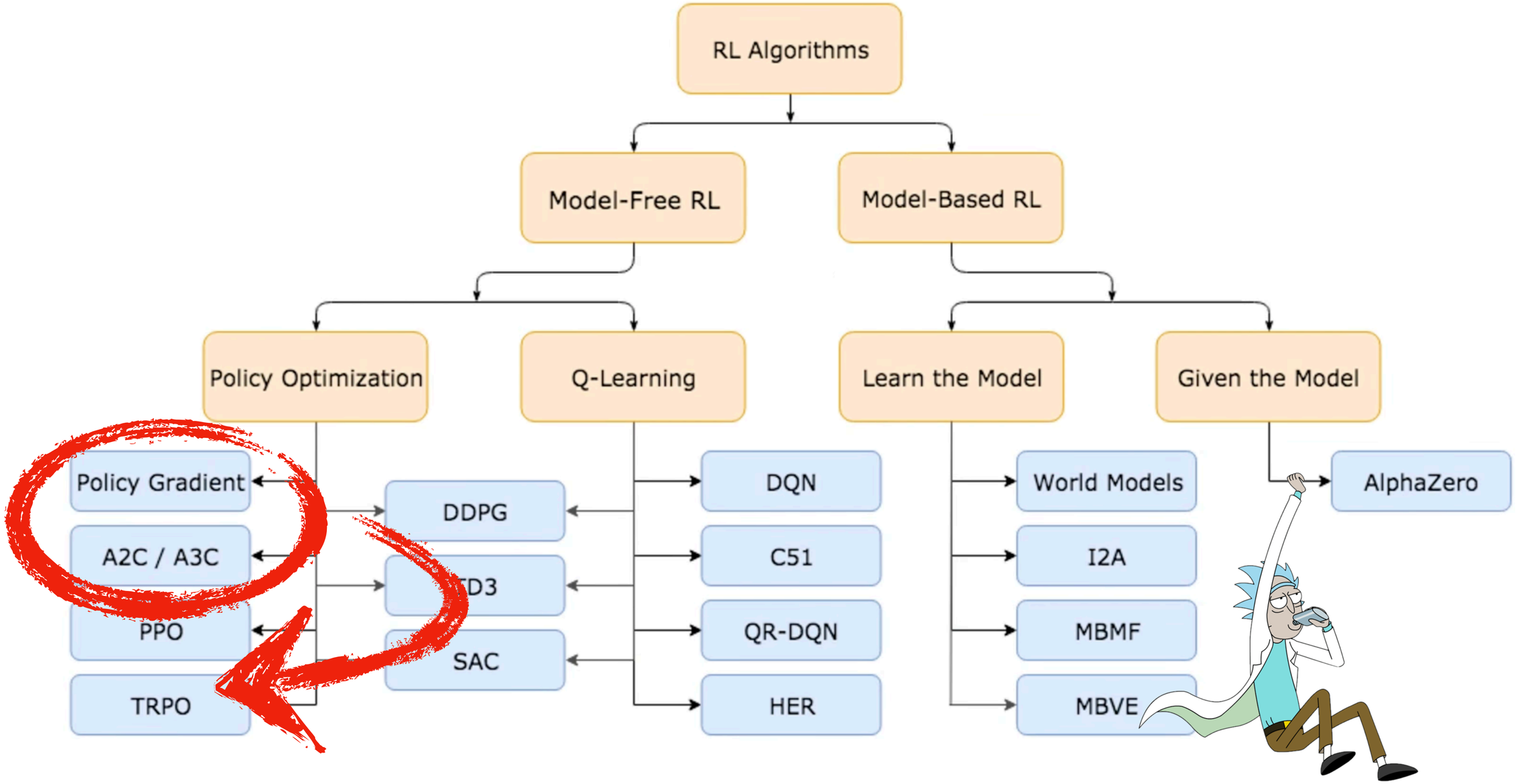
$$L_2\left(V - \gamma^i r_{s+i}\right)$$


RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

Проблема:

Если вдруг позволить агенту менять политику слишком сильно, то он может сделать катастрофически плохой шаг.

$\Rightarrow$ не сможем получать позитивных примеров


RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

Trust Region Policy Optimization (TRPO)

https://spinningup.openai.com/en/latest/algorithms/trpo.html

https://arxiv.org/pdf/1502.05477.pdf

Чтобы гарантировать, что политика не изменяется слишком сильно и не попадала в катастрофические точки, мы добавим ограничение к задаче оптимизации. Ограничение представим в виде KL-дивергенции между новой и старой политикой (которую можно представить как расстояние между двумя распределениями вероятностей) должно быть меньше дельты (δ)

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\left[\log \pi_\theta(a_t \mid s_t)\hat{A}_t\right].$$

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

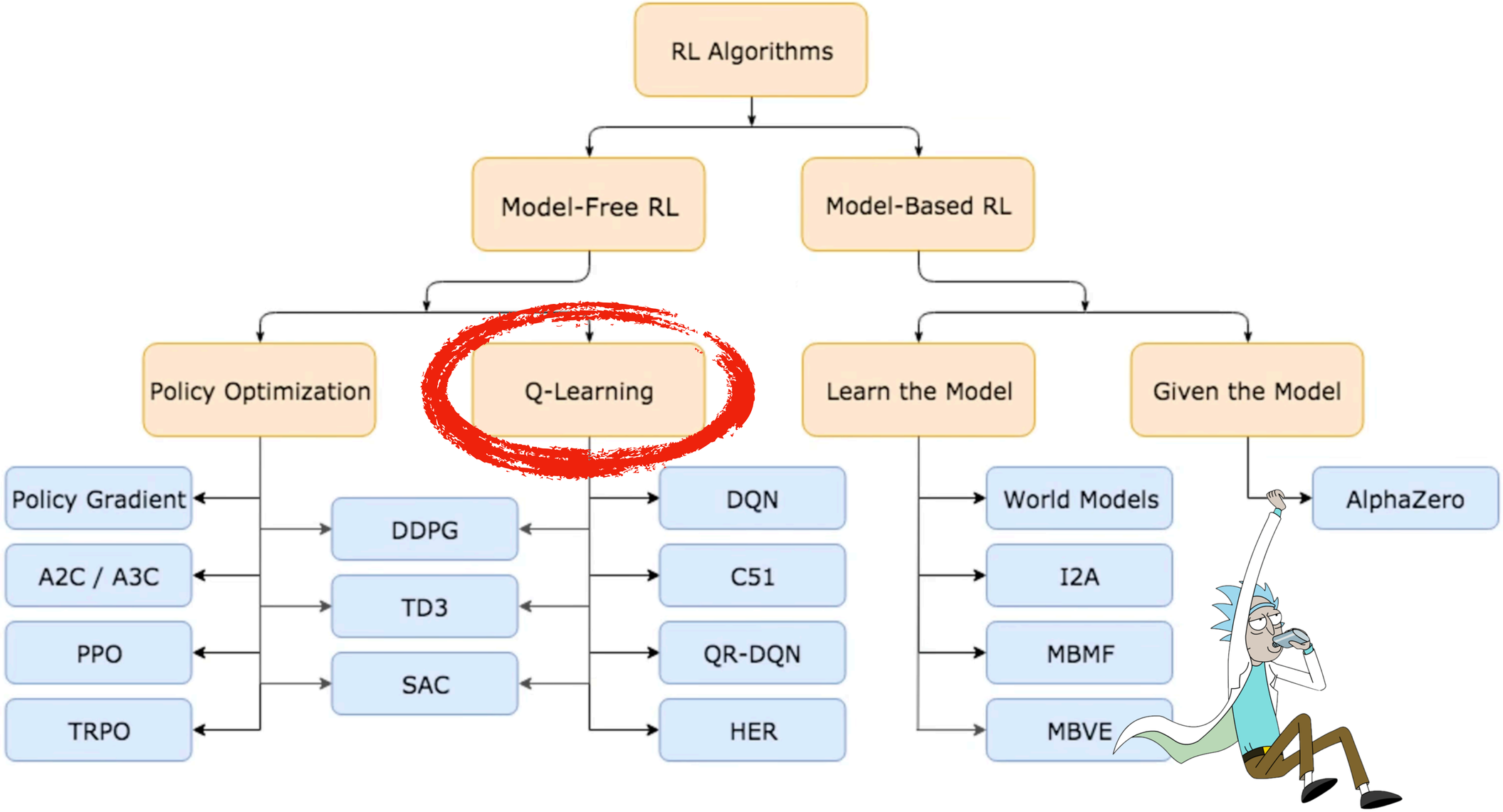
Proximal Policy Optimization (PPO)

https://openai.com/blog/openai-baselines-ppo/

https://arxiv.org/pdf/1707.06347.pdf

вместо того, чтобы добавлять ограничение отдельно, мы включаем его в целевую функцию в качестве штрафа (мы вычитаем из функции расхождение KL, умноженное на константу C).

$$\mathscr{L}_{PG} = -\sum_i r_i \ln \pi_\theta + C \times KL_{\pi_{\theta_{old}}}(\pi_\theta)$$

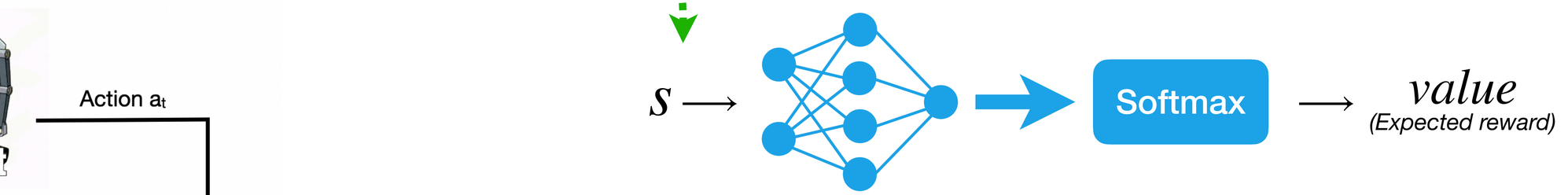

RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

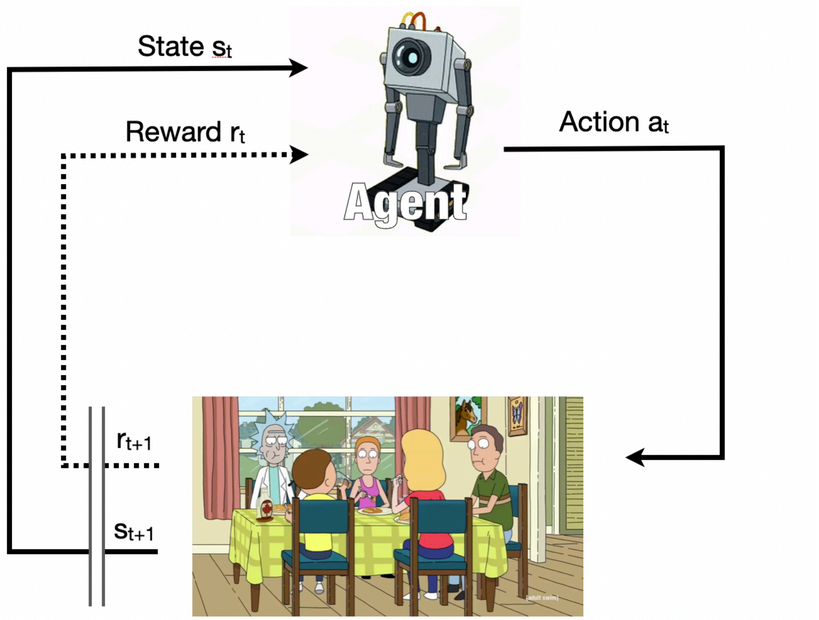
$$\sum_{t=1}^{n} r_t \longrightarrow max$$

$p(a \mid s)$ policy function

$v(s)$ value function

$Q(s, a)$ Q-function



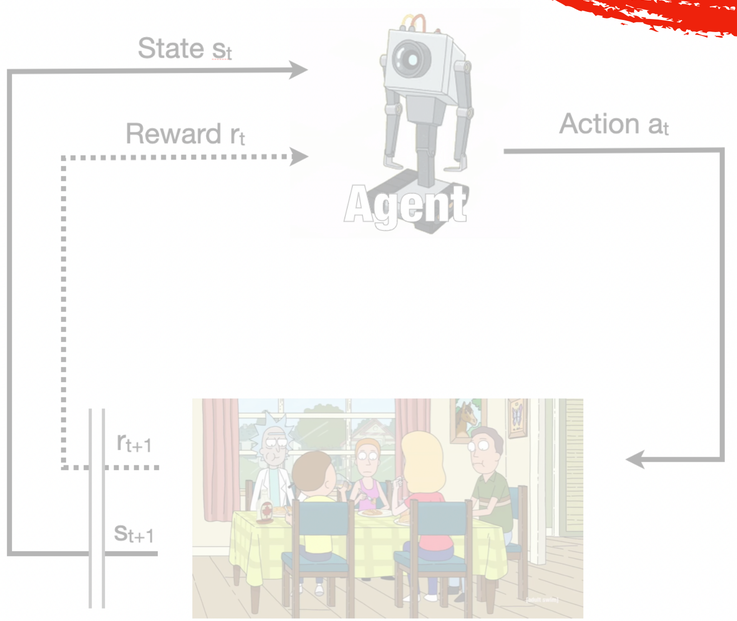


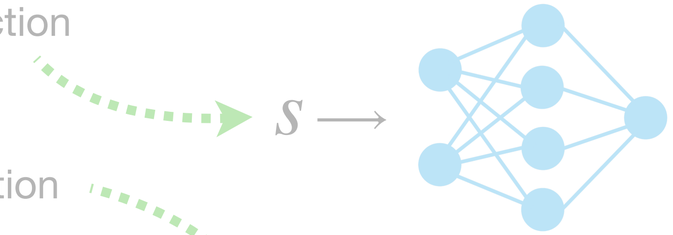
$$\sum_{t=1}^{n} r_t \longrightarrow max$$

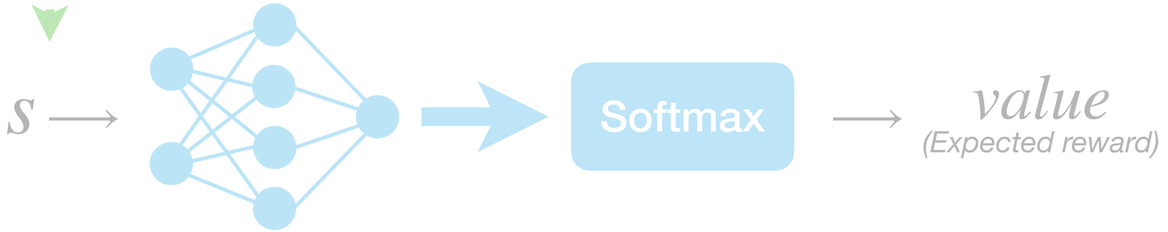
$p(a \mid s)$ policy function

$v(s)$ value function

$Q(s, a)$ Q-function







Bellman equation

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') \middle| s,a\right]$$

$Q^*(s,a)$ – Q-функция оптимальной стратегии

$\mathcal{E}$ – пространство возможных следующих состояний

Приближаем Q^* нейросетью с параметрами θ:

$$Q(s,a;\theta) \approx Q^*(s,a)$$

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a') | s,a\right]$$

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_i - Q(s,a;\theta_i)\right)^2\right],$$

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right].$$

1. Initialize Table Q(S,A) with random values.

2. Take a action (A) with epsilon — greedy policy and move to next state S'

3. Update the Q value of a previous state by following the update equation:

$$Q(s,a) = \overbrace{Q(s,a)}^{\text{Old value}} + \alpha\Big(\overbrace{r}^{\text{reward}} + \gamma \overbrace{max Q(s'a')}^{\text{optimal future value}} - \overbrace{Q(s,a)}^{\text{Old value}}\Big)$$

learning rate: α; discount: γ

Step-by-step

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Playing Atari with Deep Reinforcement Learning

https://arxiv.org/pdf/1312.5602.pdf

Что почитать?

Deep Reinforcement Learning: Pong from Pixels

http://karpathy.github.io/2016/05/31/rl/

Deep Reinforcement Learning (или за что купили DeepMind)

https://habr.com/ru/post/279729/

Курс Deep Learning на пальцах от Семена Козлова

13 - Reinforcement Learning — https://youtu.be/_x0ASf9jV9U

14 - Еще RL — https://youtu.be/aOIK1i1xt_M

Introduction to Reinforcement Learning with David Silver

https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver

https://www.davidsilver.uk/teaching/

Reinforcement Learning: An Introduction. Richard S. Sutton and Andrew G. Barto

https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf