

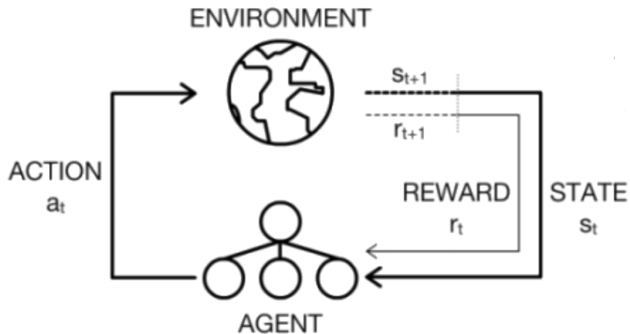
# Нейронные сетки

Ульянкин Филипп

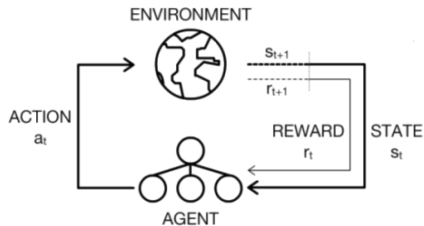
9 ноября 2019 г.

Введение в RL

# Reinforcement Learning

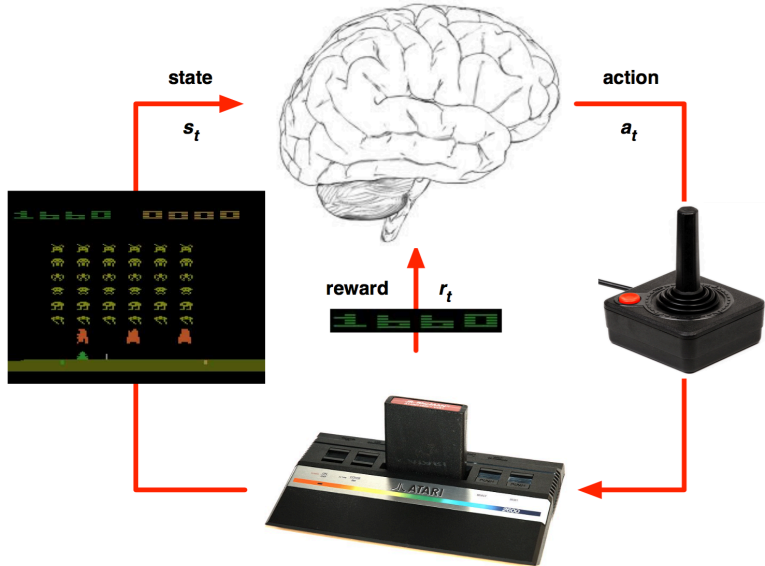


# Reinforcement Learning

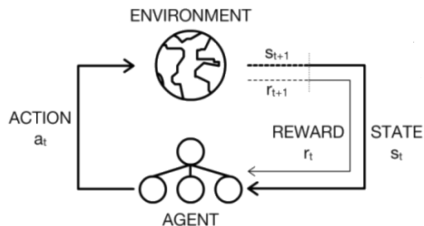


- $s_t$  — состояние среды (state);
- $a_t$  — действия агента (action);
- $r_t$  — награда (reward) за действие;

# Reinforcement Learning



# Reinforcement Learning

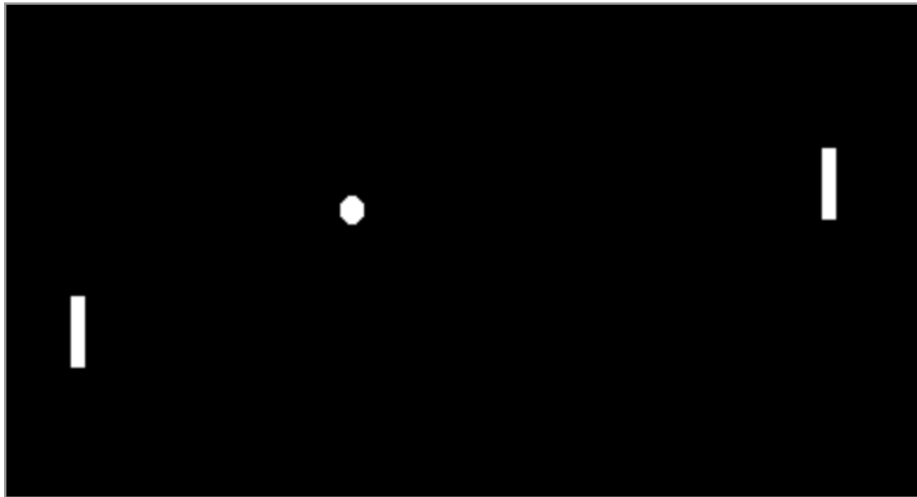


- $v(s)$  — **value function**, по state выдаёт оценку всех будущих reward;
- $p(a | s)$  — **policy function**, выводит вероятность действия в конкретном состоянии так, чтобы вероятность действия, максимизирующего reward была наибольшей
- $Q(s, a)$  — **Q-function**, сообщает reward для действия  $a$  в состоянии  $s$ .

# Reinforcement Learning

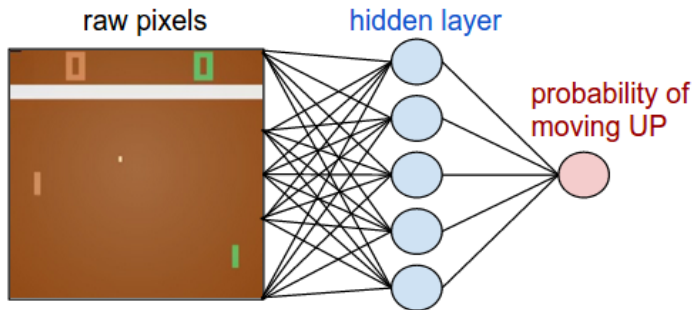
- **Идея:** мы пытаемся выразить действия агента с помощью различных функций. Если мы пытаемся каждую функцию представить в виде нейросетки, мы входим в зону deep reinforcement learning.
- В зависимости от того, какую функцию мы оптимизируем, получаем разные алгоритмы
- В примере ниже мы будем максимизировать policy function (policy gradient algorithm), после будем заниматься оптимизацией Q-функции (Q-learning)

## Пример: игра в Pong



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

# Пример: игра в Pong



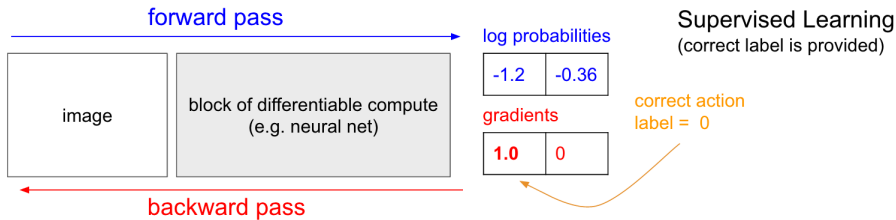
<http://karpathy.github.io/2016/05/31/rl/>



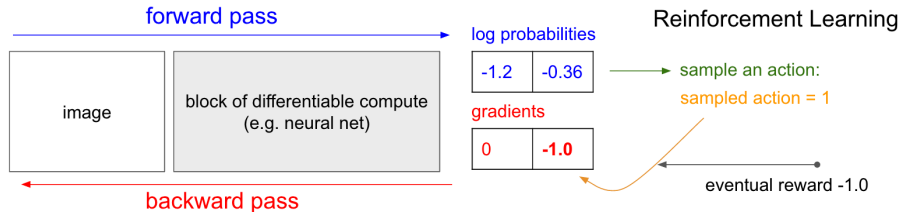
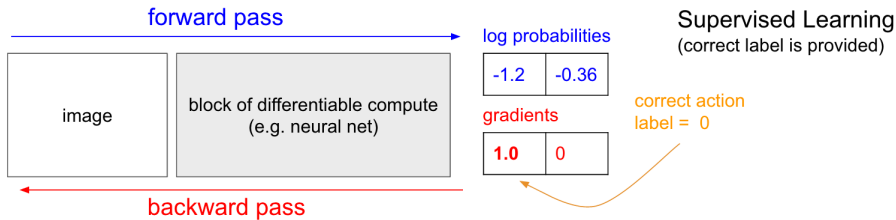
## Пример: игра в Pong

- Хотим натренировать зелёную палку выигрывать!
- Состояние  $s_t$  — пиксели экрана
- Мы хотим обучить нейросеть с одним скрытым слоем, которая по текущему дифу из пикселей предсказывать вероятность движения вверх или вниз
- Сеть сама пытается разобраться как правильно играть в эту игру на основе разницы между текущим и предыдущим кадром

# Пример: игра в Pong



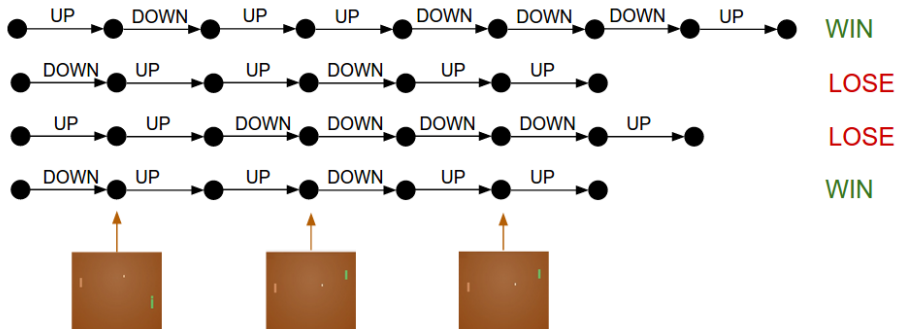
# Пример: игра в Pong



## Пример: игра в Pong

- Если бы мы учили обычную сетку, у нас был бы вектор правильных ответов для каждого действия, и мы бы делали backpropagation
- На деле мы знаем результат после многих шагов
- Будем передавать вместо таргета то, что произошло на самом деле для последовательности действий

# Пример: игра в Pong



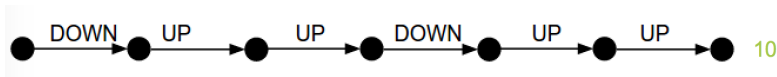
# Почему это работает?

- Пусть  $p(a \mid s, \theta)$  — вероятность действия  $a$  в состоянии  $s$ , описываемая параметрами  $\theta$ , а  $f(a)$  — награда от действия  $a$
- Мы хотим двигать веса  $\theta$  так, чтобы средняя награда  $E_a[f(a)]$  увеличивалась
- Добиться этого можно с помощью градиентного спуска

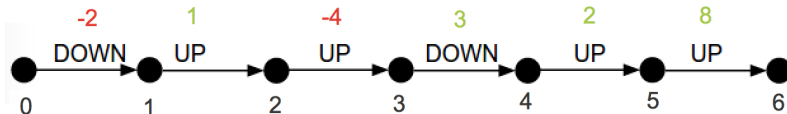
# Почему это работает?

$$\begin{aligned}\nabla_{\theta} E_a[f(a)] &= \nabla \sum_a p(a) \cdot f(a) = \\&= \sum_a \nabla_{\theta} p(a) \cdot f(a) = \\&= \sum_a p(a) \cdot \frac{\nabla_{\theta} p(a)}{p(a)} \cdot f(a) = \\&= \sum_a p(a) \cdot \nabla_{\theta} \ln p(a) \cdot f(a) = \\&= E_a[f(a) \cdot \nabla_{\theta} \ln p(a)] = -\logloss\end{aligned}$$

# Какой бывает награда

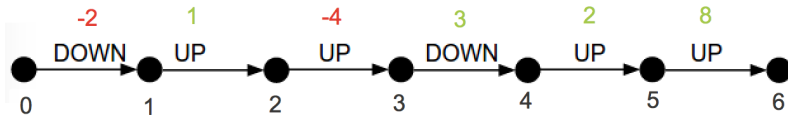


- В примере выше мы получали награду  $r$  за какую-то последовательность действий
- Можно определить награду так, что мы её получаем на каждом шаге  $r_t$





# Какой бывает награда



$$R_0 = r_0 + r_1 + r_2 + \dots + r_h$$

$$R_0 = r_0 + \gamma \cdot r_1 + \gamma \cdot r_2 + \dots + \gamma^h \cdot r_h$$

$$R_1 = r_1 + r_2 + \dots + r_h \dots$$

$$R_1 = r_1 + \gamma \cdot r_2 + \dots + \gamma^{h-1} \cdot r_h \dots$$

# Какой бывает награда

**Фактор дисконтирования  $\gamma$**  нужен, чтобы очень далёкое будущее влияло на текущее состояние сети не так сильно, как в недалёком. Когда мы просим сеть разобраться с ближайшим будущим, ей легче разобраться в ситуации и обучение стабильнее.

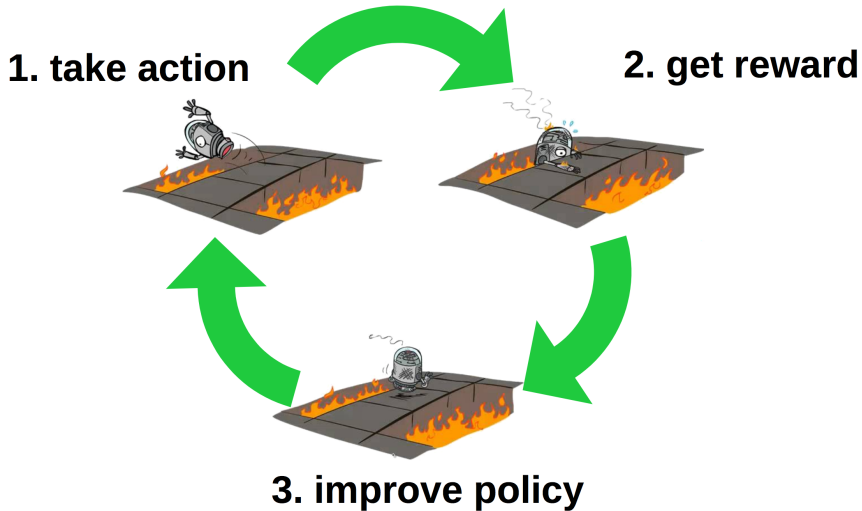
Иногда агент попадает в очень хорошее состояние и начинает постоянно получать очень большую награду. Из-за этого нейросетке сложно понимать, что 1000 лучше 999. Хочется уменьшить масштаб  $\Rightarrow$  центрирование и нормирование

Поздравляю!

Вы знаете reinforcement learning

Остальное — это детали :D

# Reinforcement Learning - это просто!



# Reinforcement Learning - это просто!

