

# Lecture 12. Intro to Reinforcement Learning

Alex Avdyushenko

Kazakh-British Technical University

November 26, 2022



# Five-minutes block

## Five-minutes block

- Describe (or draw) the general architecture of the autoencoder
- What is the main advantage of the self-supervised learning method?
- Please, write down GAN loss functions

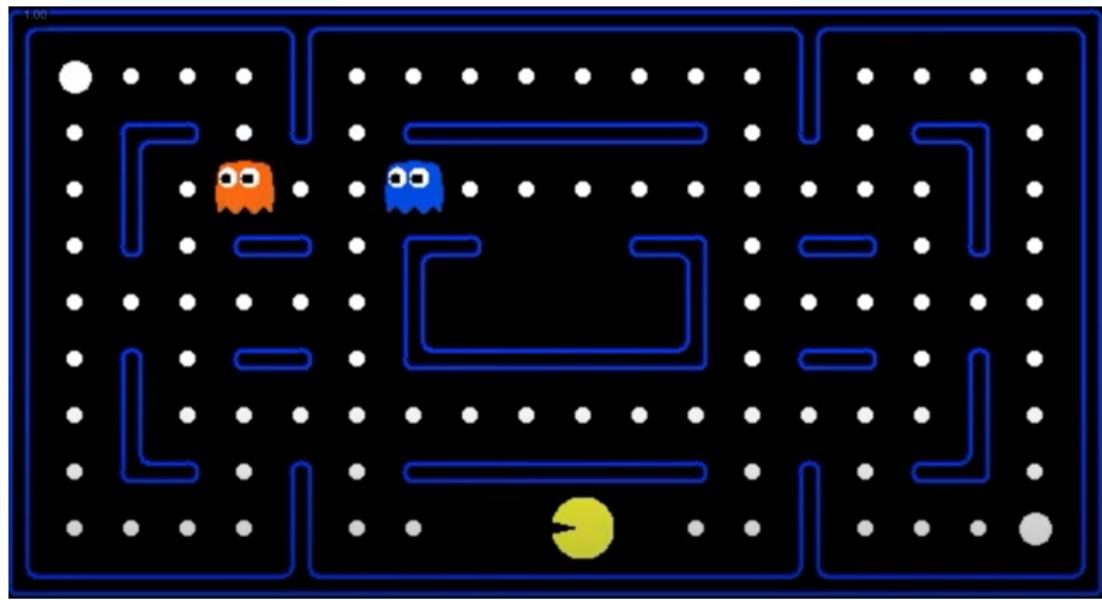
# Problem Statement

- Up to this point, we either restored the function from the training set  $(X, Y)$  (supervised learning) or searched for the structure in the set of objects  $X$  (unsupervised learning)

# Problem Statement

- Up to this point, we either restored the function from the training set  $(X, Y)$  (supervised learning) or searched for the structure in the set of objects  $X$  (unsupervised learning)
- But how does learning work in real world? Usually we do some action and get the result, gradually learning

# Real world



Deep Reinforcement Learning in Pac-man

# One more motivation

For example, we need to select the main page of a pizzeria website to attract the clients

**ДОДО ПИЦЦА** Доставка из пекарни № 1 в Казахстане  
Печь | Кухня | Закуски | Десерты | Напитки | Дорогие гости | Иные | Капуста | Брашинг | С омегой | Тропический  
Доставка из пекарни № 1 в Алматы  
23 часа - 47 руб.  
Часто заказывают:  
Итальянская пицца от 300 тг.  
Итальянская пицца от 350 тг.  
Супрематичный пицца от 350 тг.  
Капуста в сырном соусе от 350 тг.  
Му-пицца с перепелиными яйцами и сиропом от 350 тг.  
Быстро-брюзги от 350 тг.  
Без глютена  
Мы готовим для вас с любовью и заботой  
**Пицца**  
Итальянская пицца с куриным филе и соусом томатным, крем-сыром, базиликом, зеленью, соусом альбоне  
от 3 700 тг. [Выбрать](#)  
Пицца из половинки Сырно-куриная пицца из половинки пиццы с курицей и сыром  
от 3 400 тг. [Выбрать](#)  
Му-пицца с куричкой и сиропом Быстро-брюзги с курицей, перепелиными яйцами, соусом томатным, зеленью  
от 2 900 тг. [Выбрать](#)  
Му-пицца с перепелиными яйцами и сиропом Быстро-брюзги с перепелиными яйцами, соусом томатным, зеленью  
от 2 900 тг. [Выбрать](#)

**ПАПА ЙОННС** Узнайте свой адрес  
ПРИ ЗАКАЗЕ В ЗАЛЕ ПИЦЦЫ  
**35 ИЛИ 40 СМ  
2 СТАНАНА СОКА COLA 0,5  
В ПОДАРОК**  
Пицца Сеты Закуски Салаты Десерты напитки соусы  
**Пицца**  
Папа Макс Рэн  
Четверть итальянской пиццы в соусе томатном, крем-сыре, базилике, зелени, соусом альбоне  
от 2 300 тг. [Выбрать](#)  
Грандис  
Четверть итальянской пиццы в соусе томатном, крем-сыре, базилике, зелени, соусом альбоне, сырной кальмариной и зразами  
от 2 100 тг. [Выбрать](#)  
Чачак Блю Чиз  
Итальянское сочетание: итальянская пицца с соусом томатным, крем-сыре, базиликом, зеленью, соусом пармезаном, сырной кальмариной и зразами  
от 2 100 тг. [Выбрать](#)  
Корзинка пуста. Выберите пиццу из меню или напишите нам в чате  
Ввести промокод

# What approaches are there?

# What approaches are there?

- A/B testing — many users see potentially bad options

# What approaches are there?

- A/B testing — many users see potentially bad options
- multi-armed bandits — a special case of reinforcement learning

# Bernoulli's one-armed bandit



Probability of winning  $\theta = 0.05$

# Multi-Armed Bandits



$$\theta_1 = 0.02$$

$$\theta_2 = 0.01(\min)$$

$$\theta_3 = 0.05$$

$$\theta_4 = 0.1(\max)$$

We do not know the true probabilities, but we want to come up with a strategy that maximizes the payoff (reward)

# Mathematical statement of the problem

Given possible actions

$$x_1, \dots, x_n$$

# Mathematical statement of the problem

Given possible actions

$$x_1, \dots, x_n$$

At the next iteration  $t$ , for each action  $x_i^t$  performed, we get the answer

$$y_i^t \sim q(y|x_i^t, \Theta),$$

# Mathematical statement of the problem

Given possible actions

$$x_1, \dots, x_n$$

At the next iteration  $t$ , for each action  $x_i^t$  performed, we get the answer

$$y_i^t \sim q(y|x_i^t, \Theta),$$

which brings us a **reward**

$$r_i^t = r(y_i^t)$$

# Mathematical statement of the problem

Given possible actions

$$x_1, \dots, x_n$$

At the next iteration  $t$ , for each action  $x_i^t$  performed, we get the answer

$$y_i^t \sim q(y|x_i^t, \Theta),$$

which brings us a **reward**

$$r_i^t = r(y_i^t)$$

There is an optimal action  $x_{i^*}$  (sometimes  $x_{i_t^*}$ )

$$\forall i : E(r_{i_t^*}^t) \geq E(r_i^t)$$

# Mathematical statement of the problem

Given possible actions

$$x_1, \dots, x_n$$

At the next iteration  $t$ , for each action  $x_i^t$  performed, we get the answer

$$y_i^t \sim q(y|x_i^t, \Theta),$$

which brings us a **reward**

$$r_i^t = r(y_i^t)$$

There is an optimal action  $x_{i^*}$  (sometimes  $x_{i_t^*}$ )

$$\forall i : E(r_{i_t^*}^t) \geq E(r_i^t)$$

## Question

How to evaluate different strategies?

# Measure of quality

# Measure of quality

The quality measure of the multi-armed bandit algorithm  $a$  is usually **regret**

$$R(a) = \sum_{t=1}^T \left( E(r_{i_t^*}^t) - E(r_{i_t^a}^t) \right)$$

Under synthetic conditions (when we know the probabilities), we can consider

$$E(R) = \int_{\Theta} R(a) d\Theta$$

# Multi-Armed Bandits possible applications

## Areas:

- advertising banners
- recommendations (goods, music, movies etc.)
- slot machines in the casino

## Approaches:

- Thompson sampling
- Upper Confidence Bound (UCB)
- $\varepsilon$ -greedy strategy

### Note

The main difference from more general reinforcement learning approach is that there is no environment state in multi-armed bandits

# Multi-Armed Bandits

## Disadvantage

Do not take into account delayed effects

For example, the effect of clickbait in advertising

# Multi-Armed Bandits

## Disadvantage

Do not take into account delayed effects

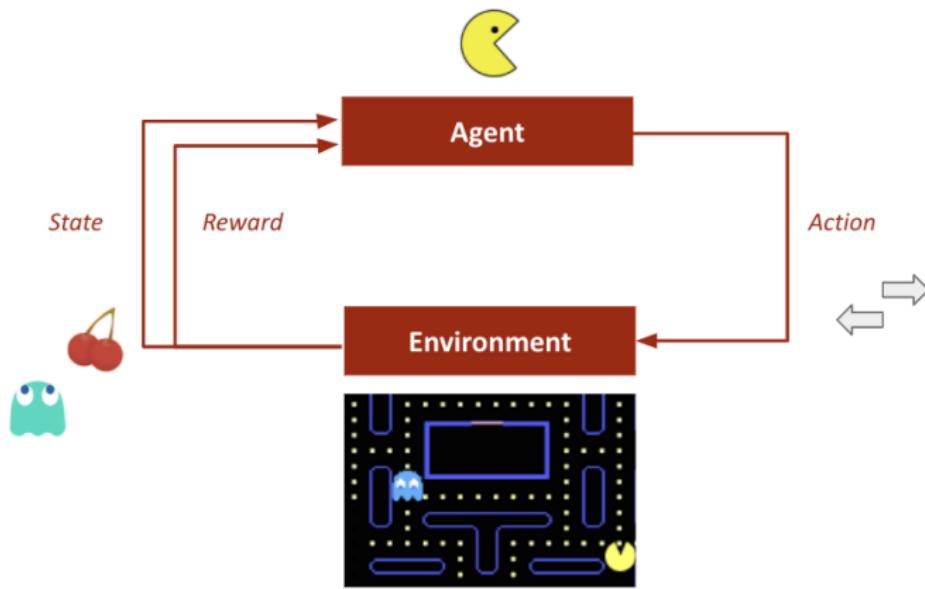
For example, the effect of clickbait in advertising



SHOCK! Cats want to enslave us

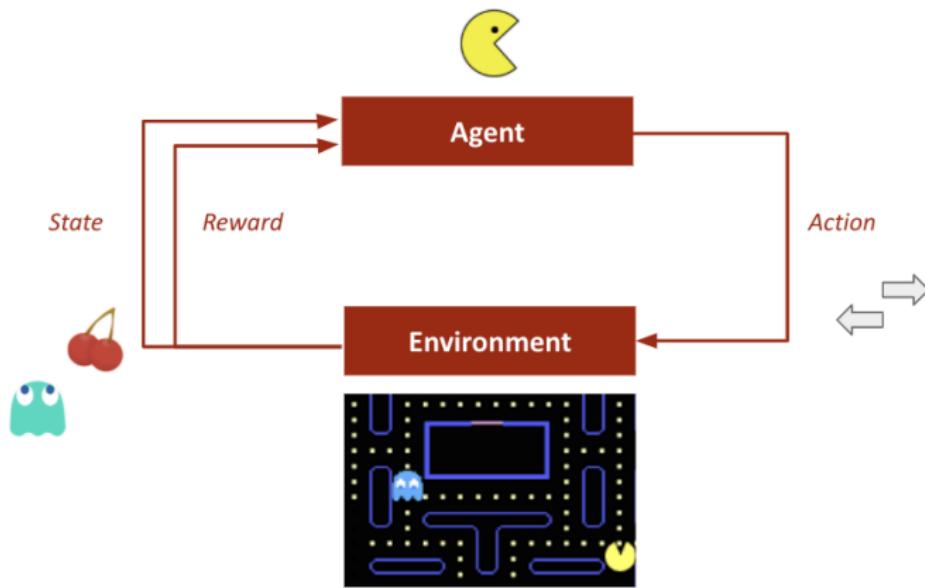
# Reinforcement Learning

## Examples



# Reinforcement Learning

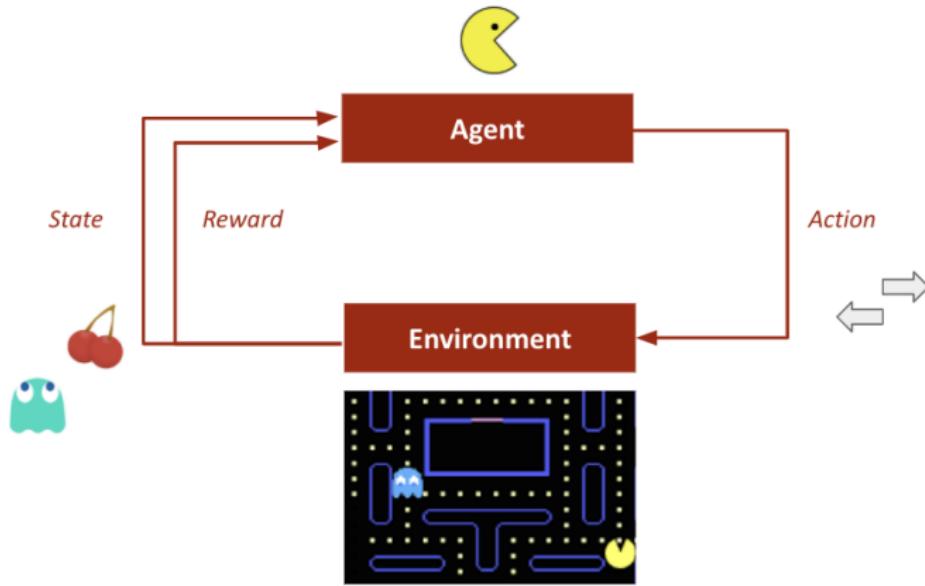
## Examples



- robot control

# Reinforcement Learning

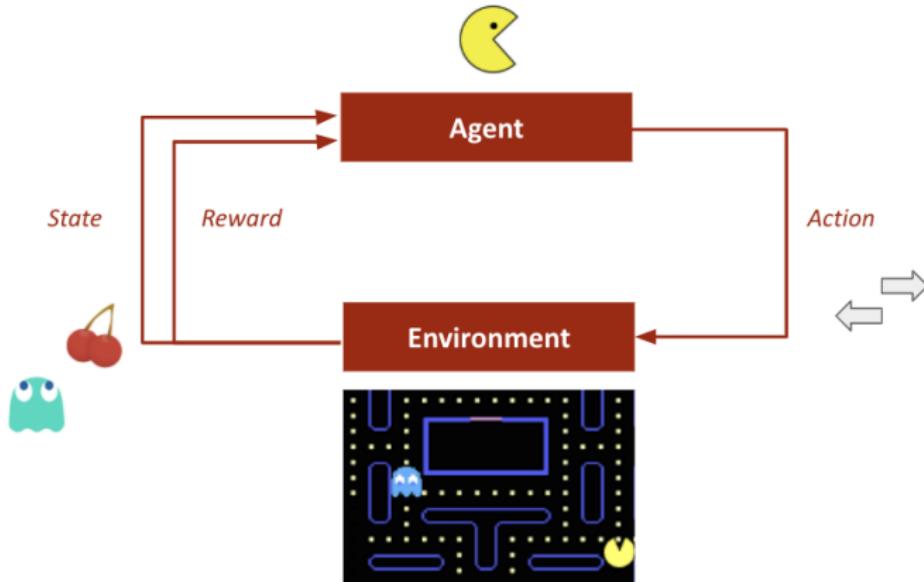
## Examples



- robot control
- (video) games

# Reinforcement Learning

## Examples



- robot control
- (video) games
- security management

# Agent Model in a Changing Environment

## Definitions:

- state  $s \in S$
- agent action  $a \in A$
- reward  $r \in R$
- state transition dynamics

$$P(s_{t+1}|s_t, a_t, \dots, s_{t-i}, a_{t-i}, \dots, s_0, a_0)$$

- win function

## Task:

$$\pi(a|s) : \mathbb{E}_\pi[R] \rightarrow \max$$

$$r_t = r(s_t, a_t, \dots, s_0, a_0)$$

- total reward  $R = \sum_t r_t$
- agent policy  $\pi(a|s)$

# Cross-entropy method. Algorithm

Trajectory —  $[s_0, a_0, s_1, a_1, s_2, \dots, a_{T-1}, s_T]$

**Initialize** strategy model  $\pi(a|s)$

**Repeat:**

- play  $N$  sessions
- choose the best  $K$  of them and take their trajectories
- adjust  $\pi(a|s)$  so that  $s$  is able to maximize the probabilities of actions from the best trajectories

# Training Example

## Cross-entropy method. Implementation with a table

As a strategy model, we simply take a matrix  $\pi$  of dimension  $|S| \times |A|$   
 $\pi(a|s) = \pi_{s,a}$   
after selecting the best trajectories, we obtain a set of pairs

$$\text{Elite} = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_H, a_H)]$$

and maximize the likelihood

$$\pi_{s,a} = \frac{\sum_{s_t, a_t \in \text{Elite}} [s_t = s][a_t = a]}{\sum_{s_t, a_t \text{ in Elite}} [s_t = s]}$$

There is a problem..

# There is a problem..

that there are a lot of states:



# Approximate cross-entropy methods

Possible solutions:

- split the state space into sections and treat them as states

# Approximate cross-entropy methods

Possible solutions:

- split the state space into sections and treat them as states
- get probabilities from  $\pi_\theta(a|s)$  machine learning model: linear model, neural network, random forest

# Approximate cross-entropy methods

Possible solutions:

- split the state space into sections and treat them as states
- get probabilities from  $\pi_\theta(a|s)$  machine learning model: linear model, neural network, random forest
- often these probabilities need to be further specified later

# Approximate cross-entropy method

## Example

# Approximate cross-entropy method

## Example

- As a strategy model, we take just a neural network  $\pi_\theta$

# Approximate cross-entropy method

## Example

- As a strategy model, we take just a neural network  $\pi_\theta$
- Initialize with random weights

# Approximate cross-entropy method

## Example

- As a strategy model, we take just a neural network  $\pi_\theta$
- Initialize with random weights
- At each iteration, after selecting the best trajectories, we obtain a set of pairs

$$\text{Elite} = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_H, a_H)]$$

# Approximate cross-entropy method

## Example

- As a strategy model, we take just a neural network  $\pi_\theta$
- Initialize with random weights
- At each iteration, after selecting the best trajectories, we obtain a set of pairs

$$\text{Elite} = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_H, a_H)]$$

- and perform optimization

$$\pi = \arg \max_{\theta} \sum_{s_i, a_i \in \text{Elite}} \log \pi(a_i | s_i) = \arg \max_{\theta} \mathcal{L}(\theta)$$

# Approximate cross-entropy method

## Example

- As a strategy model, we take just a neural network  $\pi_\theta$
- Initialize with random weights
- At each iteration, after selecting the best trajectories, we obtain a set of pairs

$$\text{Elite} = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_H, a_H)]$$

- and perform optimization

$$\pi = \arg \max_{\theta} \sum_{s_i, a_i \in \text{Elite}} \log \pi(a_i | s_i) = \arg \max_{\theta} \mathcal{L}(\theta)$$

- that is

$$\theta_{t+1} = \theta_t + \alpha \nabla \mathcal{L}(\theta)$$

# Why is it called cross-entropy method?

$$\begin{aligned} KL(p_1(x) \| p_2(x)) &= E_{x \sim p_1(x)} \log \frac{p_1(x)}{p_2(x)} = \\ &= \underbrace{E_{x \sim p_1(x)} \log p_1(x)}_{\text{entropy}} - \underbrace{E_{x \sim p_1(x)} \log p_2(x)}_{\text{cross entropy}} \end{aligned}$$

# Disadvantages of the cross-entropy method

## Disadvantages of the cross-entropy method

- it is unstable for small samples

# Disadvantages of the cross-entropy method

- it is unstable for small samples
- in the case of a non-deterministic environment, it chooses lucky cases (random played in favor of the agent)

# Disadvantages of the cross-entropy method

- it is unstable for small samples
- in the case of a non-deterministic environment, it chooses lucky cases (random played in favor of the agent)
- it is focusing on behavior in simple states

# Disadvantages of the cross-entropy method

- it is unstable for small samples
- in the case of a non-deterministic environment, it chooses lucky cases (random played in favor of the agent)
- it is focusing on behavior in simple states
- it ignores a lot of information

## Disadvantages of the cross-entropy method

- it is unstable for small samples
- in the case of a non-deterministic environment, it chooses lucky cases (random played in favor of the agent)
- it is focusing on behavior in simple states
- it ignores a lot of information
- there are tasks in which the end never comes (stock market game)

# Q-Learning

- sometimes it is not necessary to finish the game to evaluate the effectiveness of the strategy
- the effect of the action may appear later

# Markov Decision Process

## Definitions:

- state  $s \in S$
- agent action  $a \in A$
- reward  $r \in R$
- transition dynamics
- agent policy

$$P(s_{t+1}|s_t, a_t, \dots, s_{t-i}, a_{t-i}, \dots, s_0, a_0) = P(s_{t+1}|s_t, a_t) \text{ (Markov's assumption)}$$

- win function

Task:

$$r_t = r(s_t, a_t)$$

$$\pi(a|s) : \mathbb{E}_\pi[R] \rightarrow \max$$

(Markov's assumption)

- total reward  $R = \sum_t r_t$

# Important definitions

Average absolute win:

$$\mathbb{E}_{s_0 \sim p(s_0)}, \mathbb{E}_{a_0 \sim \pi(a|s_0)}, \mathbb{E}_{s_1, r_0 \sim P(s', r|s, a)} \dots [r_0 + r_1 + \dots + r_T]$$

State value function:

$$V^\pi(s) = \mathbb{E}_\pi[R_t | s_t = s] = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right]$$

Action value function in state:

$$Q^\pi(s, a) = \mathbb{E}_\pi[R_t | s_t = s, a_t = a] = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right]$$

where  $\pi$  is the strategy followed by the agent,  $\gamma \in [0, 1]$

# TD-training

## temporal difference

We arbitrarily initialize the function  $V(s)$  and the strategy  $\pi$   
Then we repeat

- initialize  $s$
- for each agent step
  - ▶ select  $a$  by strategy  $\pi$
  - ▶ do action  $a$ , get result  $r$  and next state  $s'$
  - ▶ update function  $V(s)$  by formula

$$V(s) = V(s) + \alpha (r + \gamma V(s') - V(s))$$

- ▶ go to the next step by assigning  $s := s'$

# Bellman Equation

for the optimal value function  $Q^*$

$$Q^*(s, a) = \mathbb{E}_\pi \left[ r_t + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a \right]$$

## Note

The greedy strategy  $\pi$  with respect to  $Q^*(s, a)$  “choose the action that maximizes the Bellman equations” is optimal.

# State utility recalculation

$$V(s) = \max_a [r(s, a) + \gamma V(s'(s, a))]$$

that is, with probabilistic transitions

$$V(s) = \max_a [r(s, a) + \gamma \mathbb{E}_{s' \sim P(s'|s, a)} V(s')]$$

## Iterative State Utility Recalculation Formula

$$\forall s \quad V_0(s) = 0$$

$$V_{i+1}(s) = \max_a [r(s, a) + \gamma \mathbb{E}_{s' \sim P(s'|s, a)} V_i(s')]$$

# State utility recalculation

$$V(s) = \max_a [r(s, a) + \gamma V(s'(s, a))]$$

that is, with probabilistic transitions

$$V(s) = \max_a [r(s, a) + \gamma \mathbb{E}_{s' \sim P(s'|s, a)} V(s')]$$

## Iterative State Utility Recalculation Formula

$$\forall s \quad V_0(s) = 0$$

$$V_{i+1}(s) = \max_a [r(s, a) + \gamma \mathbb{E}_{s' \sim P(s'|s, a)} V_i(s')]$$

### Note

To use this formula in practice, we need to know the transition probabilities  $P(s'|s, a)$

## Utility of an action

$$Q(s, a) = r(s, a) + \gamma V(s')$$

The strategy of the game is defined as follows

$$\pi(s) : \arg \max_a Q(s, a)$$

# Utility of an action

$$Q(s, a) = r(s, a) + \gamma V(s')$$

The strategy of the game is defined as follows

$$\pi(s) : \arg \max_a Q(s, a)$$

Again due to stochasticity

$$Q(s, a) = \mathbb{E}_{s'}[r(s, a) + \gamma V(s')]$$

It is possible to estimate the expected value without an explicit distribution, using the Monte Carlo method and averaging over the outcomes:

$$Q(s_t, a_t) \leftarrow \alpha \left( r_t + \gamma \max_a Q(s_{t+1}, a) \right) + (1 - \alpha) Q(s_t, a_t)$$

# DQN (Deep Q-Learning Network)

Environment is Atari game emulator, each frame  $210 \times 160$  pix, 128 col



Pong



Breakout



Space  
Invaders



Seaquest



Beam Rider

# DQN (Deep Q-Learning Network)

Environment is Atari game emulator, each frame  $210 \times 160\text{pix}$ ,  $128\text{col}$



Pong

Breakout

Space  
Invaders

Seaquest

Beam Rider

$s$  states: 4 consecutive frames compressed to  $84 \times 84$

Actions  $a$ : from 4 to 18, depending on the game

Rewards: current in-game SCORE

Value function  $Q(s, a; w)$ :

ConvNN with input  $s$  and  $|A|$  outputs

# DQN Method

# DQN Method

Saves trajectories  $(s_t, a_t, r_t)_{t=1}^T$  in replay memory for repeated experience replay

## DQN Method

Saves trajectories  $(s_t, a_t, r_t)_{t=1}^T$  in replay memory for repeated experience replay

Gets approximation of the optimal value function  $Q(s_t, a_t)$  for fixed current network parameters  $w_t$

$$y_t = \begin{cases} r_t, & \text{if state } s_{t+1} \text{ terminal} \\ r_t + \gamma \max_a Q(s_{t+1}, a; w_t), & \text{otherwise} \end{cases}$$

## DQN Method

Saves trajectories  $(s_t, a_t, r_t)_{t=1}^T$  in replay memory for repeated experience replay

Gets approximation of the optimal value function  $Q(s_t, a_t)$  for fixed current network parameters  $w_t$

$$y_t = \begin{cases} r_t, & \text{if state } s_{t+1} \text{ terminal} \\ r_t + \gamma \max_a Q(s_{t+1}, a; w_t), & \text{otherwise} \end{cases}$$

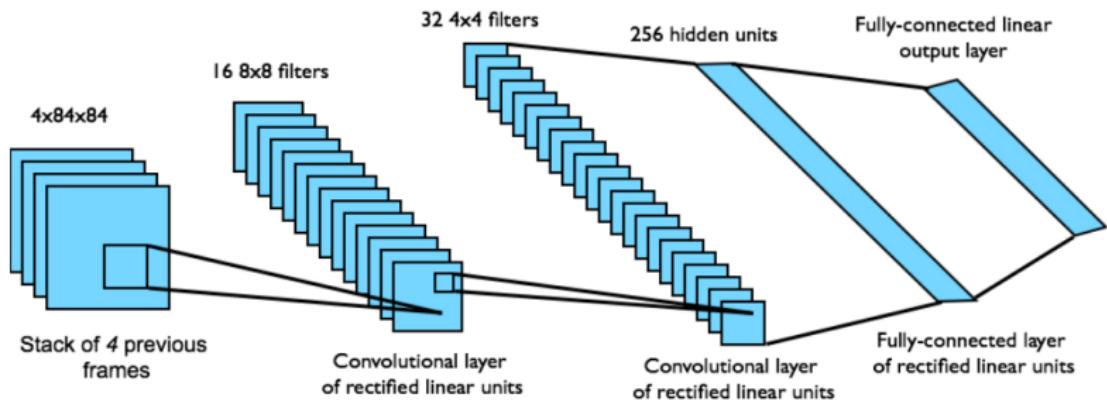
Evaluates loss function for training the neural network model  $Q(s, a; w)$ :

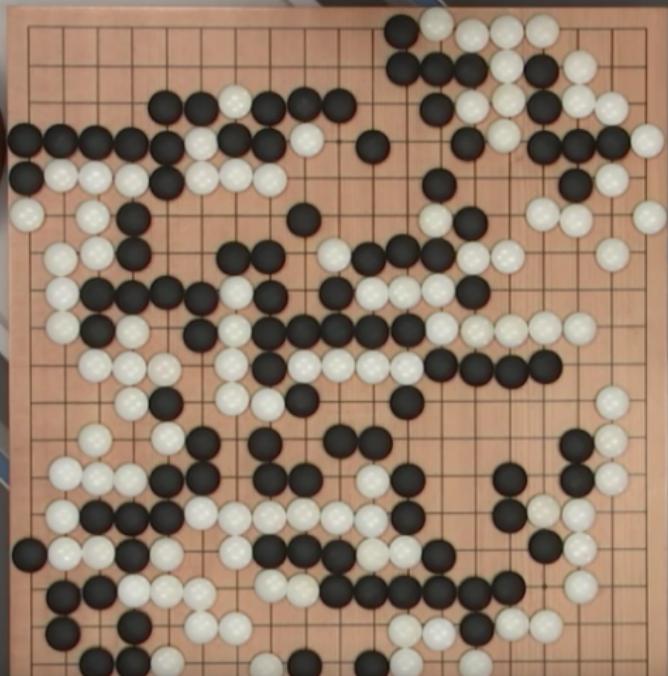
$$\mathcal{L}(w) = (Q(s_t, a_t; w) - y_t)^2$$

with stochastic gradient SGD (by mini-batches of length 32):

$$w_{t+1} = w_t - \eta (Q(s_t, a_t; w_t) - y_t) \nabla_w Q(s_t, a_t; w_t)$$

# Network architecture for the value function







BLIZZARD  
ENTERTAINMENT

# Summary

- we got acquainted with the reinforcement learning problem statement
- touched a little multi-armed bandits
- got acquainted with the methods of cross-entropy, TD-learning and Q-learning
- discussed Deep Q-learning Network
- did NOT discuss the central method in robotics — policy gradient descent

# Summary

- we got acquainted with the reinforcement learning problem statement
- touched a little multi-armed bandits
- got acquainted with the methods of cross-entropy, TD-learning and Q-learning
- discussed Deep Q-learning Network
- did NOT discuss the central method in robotics — policy gradient descent

What else can you see?

- main book on the topic: Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto
- Educational resource on RL produced by OpenAI
- Yuxi Li. Resources for Deep Reinforcement Learning. 2018
- Intro to RL with David Silver