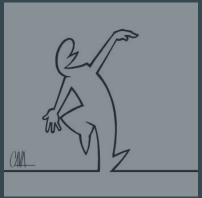
# Reinforcement learning

why and how?

...

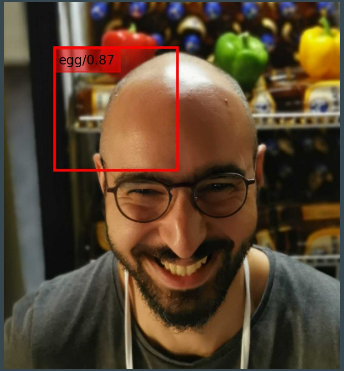


Nima H. Siboni
https://github.com/nima-siboni

# Intro to myself

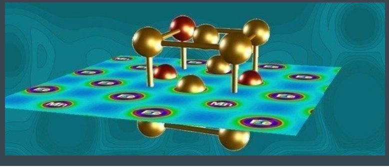
Former position: Machine learning team-lead in DeepMetis
Future position: Senior RL research engineer in InstaDeep

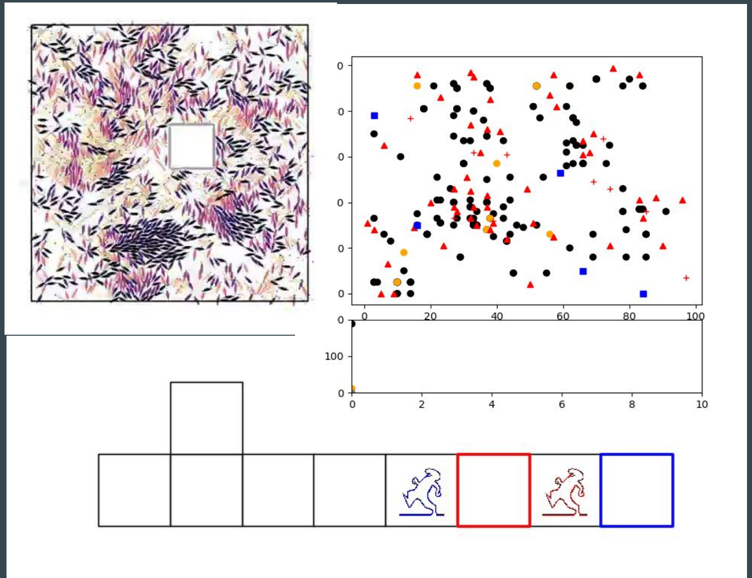
Short CV (in pictures)




- B.Sc. Mechanical Engineer
- M.Sc. Simulation Science
- PhD in Simulation of Complex Systems
- Postdoc in Simulation of Complex Fluids
- Data Science and Machine learning training

2

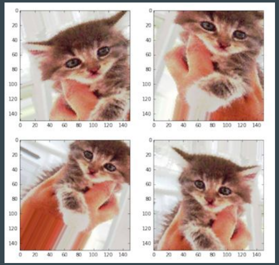


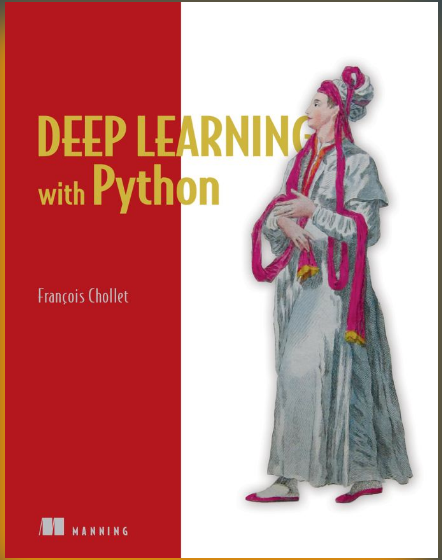


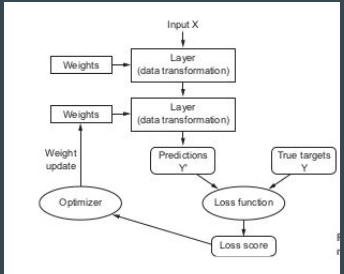

Current Affiliations:

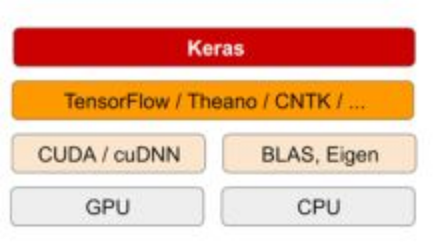
- Data Science Retreat (lecturer)
- Max-Planck-Institute (guest researcher)
- The Mentoring Club gUG (mentor/mentee)

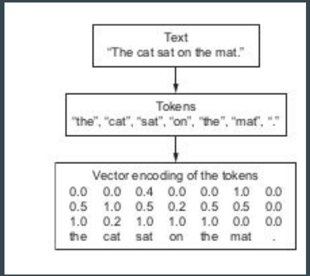
# Course disclaimer!

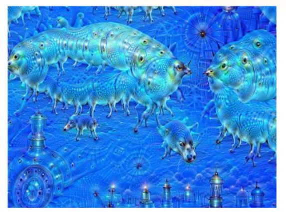


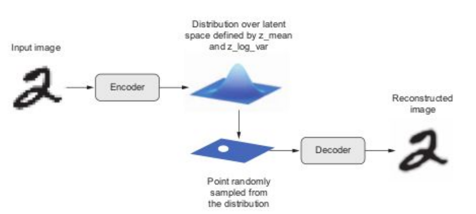




# Course outline

- Introduction

  What sort of problems you can solve with it? How is it new to you?

- RL problem formulation

  Modeling: Lots of new terms to be defined and connected to each other

- Solution of a RL problem

  Simulation environment and a zoo of methods

# Course outline

- Introduction

  What sort of problems you can solve with it? How is it new to you?

- RL problem formulation

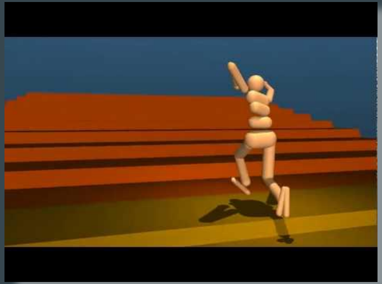
  Modeling: Lots of new terms to be defined and connected to each other

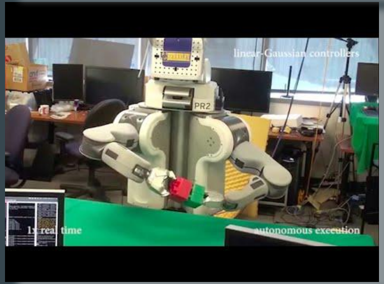
- Solution of a RL problem

  Simulation environment and a zoo of methods

# Some examples

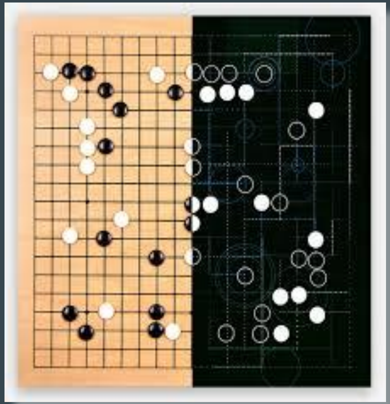
Robotics and Autonomous Systems

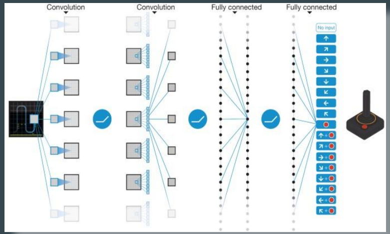


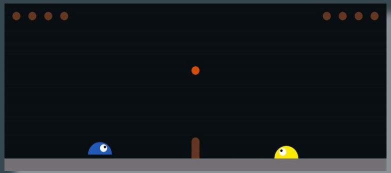




Games, Games, Games, and more Games





Some Other Industrial Applications

Chip Design with Deep Reinforcement Learning

Thursday, April 23, 2020

Posted by Anna Goldie, Senior Software Engineer and Azalia Mirhoseini, Senior Research Scientist, Google Research, Brain Team

BLOG POST
RESEARCH

20 JUL 2016

**DeepMind AI Reduces Google Data Centre Cooling Bill by 40%**

Article | Open Access | Published: 16 February 2022

**Magnetic control of tokamak plasmas through deep reinforcement learning**

Not RL but fun to watch

# RL vs. SL & UL

**Unsupervised Learning**
Clusters or dimension reduction
k-means, PCA, etc.

**Supervised Learning**
Classifier or regressor
Neural networks, SVMs, etc.

**Reinforcement Learning**
Find an optimal behavior
Monte-Carlo, Q-learning, etc.

# RL's basic ingredients

How do we make good decisions?!

What is the situation?!

State

What are the possible actions?

Action space

What are the consequences of each action?

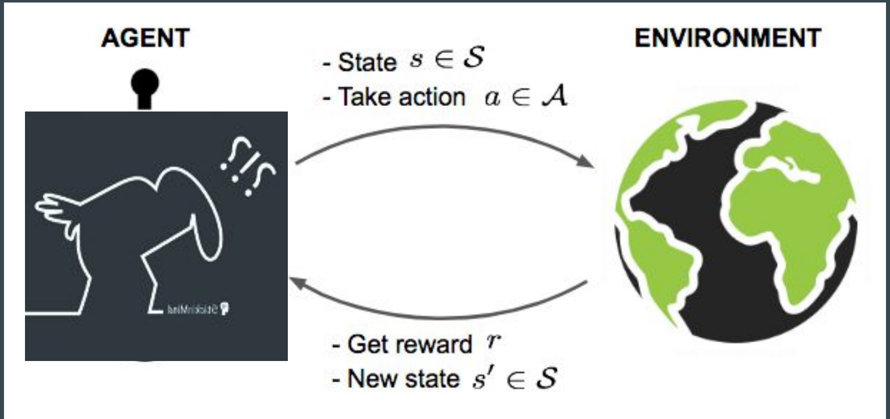
- How rewarding/costly is each action?
- Where do I end up after this action?

Environment

Policy

The mapping between the state and the actions

State $\xrightarrow{\text{Policy}}$ Action




The goal:
Find the optimal policy for a sequential decision making problem.

Optimal solution: Maximize the sum of all rewards, i.e. current reward and what comes after.

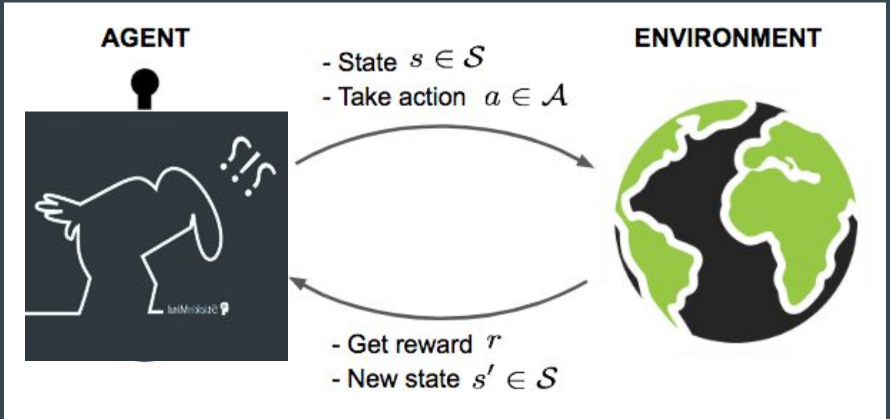
# Where is the learning?

Optimization formulation

Given the state,

what is the action that maximizes

my <u>expected reward</u>




How is it a machine learning problem?

Reinforcement Learning

Learning the ***OPTIMAL*** policy from ***EXPERIENCE***

Anatomy of a RL solution:

- Start with a policy
- Make it better (?!)

*until you reach the optimal policy*

# Examples

Give me examples!

- What are the states?
- What are the actions?
- How are the dynamics?
- What are the rewards at each step?
- What is your policy?

What is RL "Hello World!"?

Food for thought: Can you always cast your problem into a RL problem?

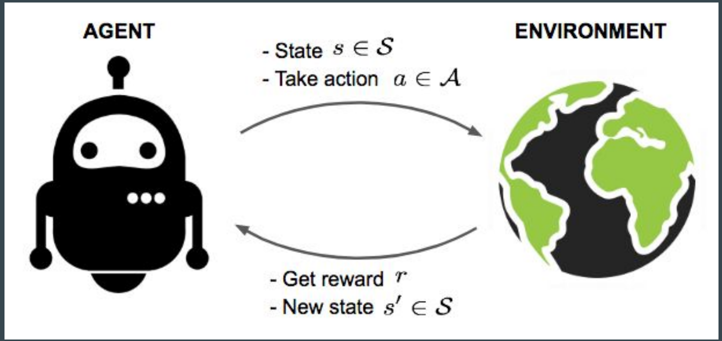
- Reward hypothesis

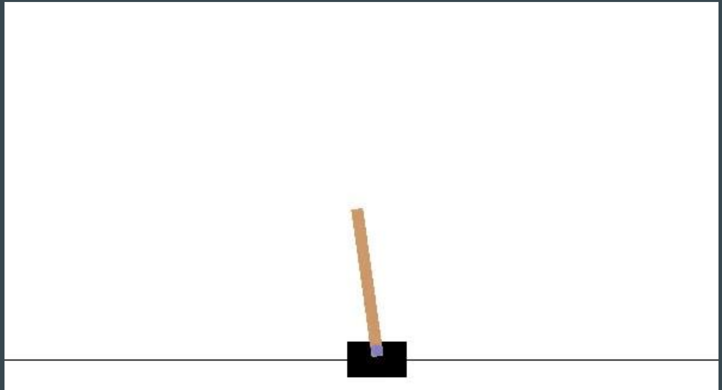
  States that "all of what we **mean by goals and purposes** can be well thought of as **maximization** of the expected value of the **cumulative sum** of a received **scalar signal** (**reward**)." Rich Sutton

Food for thought: Can you think of a badly designed reward?

# Investigate an environment

What do you expect from an environment?



```
env = gym.make('CartPole-v0')
```

# How to improve a policy?

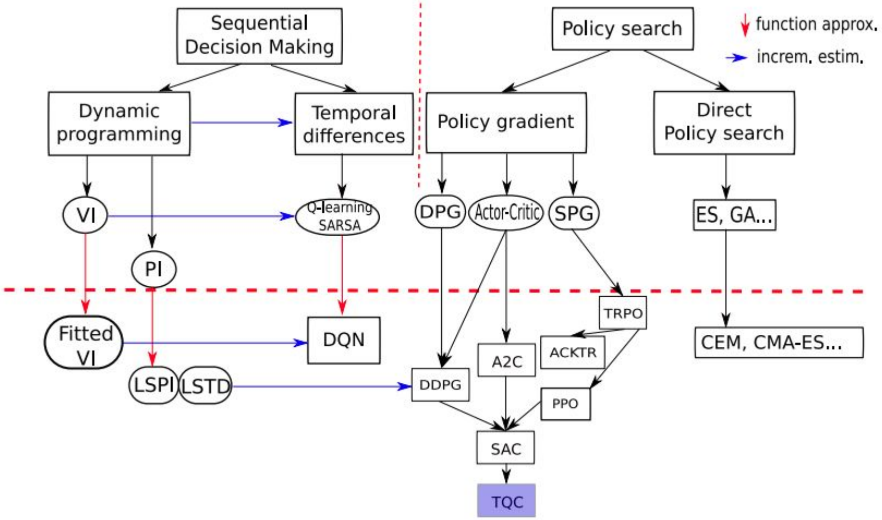

- a table,
- a function, or
- a deep neural network (*Deep RL*)

The inner mechanism of finding the optimal solution

- Find the *value/worth* of each (state, action) pair from (your experience or other’s) and take the best action
- Change the policy a bit and evaluate the consequences

# An overview of RL-algorithms taxonomy

## The Big Picture

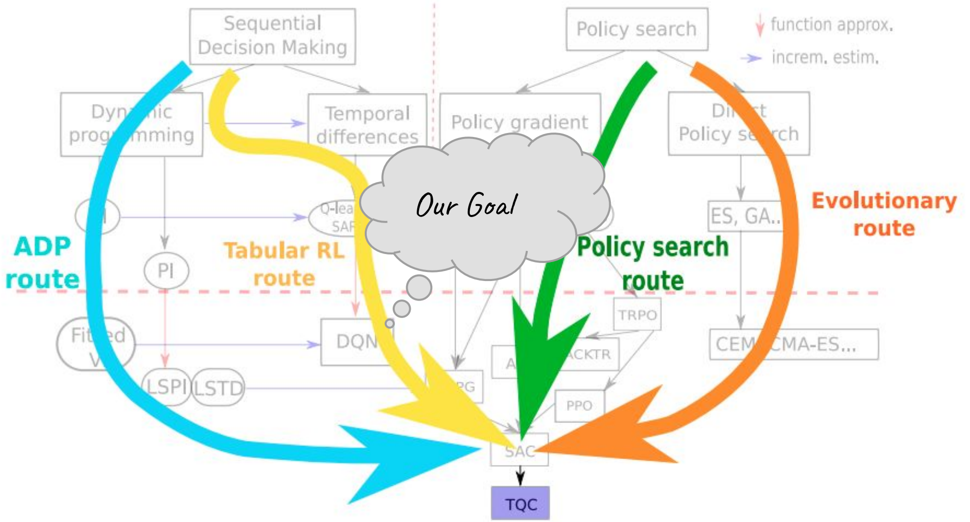



- A very partial view of the whole RL literature

Olivier Sigaud

# An overview of RL-algorithms taxonomy

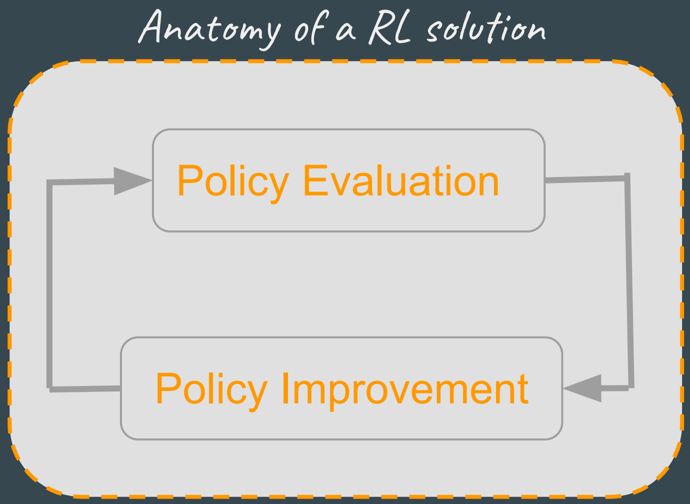
The four routes to deep RL




► Four different ways to come to Deep RL

# The core questions

- How good is a policy?
- How can you make a policy better?




My claim: The optimization problem is solved if you can find how good your policy is

# Policy Evaluation

## Q: What are the measures of a good policy?

Let’s say you are in a particular state and you are offered two policies, how do you choose?

- Immediate reward? Probably not a good idea.
- Some of all rewards you get? That seems more appropriate!
- Is future that important? Maybe, yes, Maybe no!

*Let’s formally define values function.*

## Q: How to find out V(s) for all the states?

- Directly solving the Bellman Equation
- Guessing based on the experimenting [Monte Carlo method, Q-Learning, DQN]

# (Finally) Bellman Equation

- What happens until eternity is what happens now plus what happens after that

*Let's derive the Bellman equation together!*

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a) \Big[ r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r \mid s, a) \Big[ r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

- Assumptions:
  - A Markovian process (is it an important assumption?)
  - The dynamics is known!
  - States and actions are discrete!

- Bellman equation has a good mathematical property! A lovely one!

# Let’s do an exercise!

# What did we do?

Goal:  finding the optimal behavior, without prior knowledge of the environment.

These are methods which require only experience!

Experience: sample sequences of states, actions, and rewards (from interaction with the environment).

# How to solve the Bellman equation?

How to solve an equation?

- Direct analytical methods
- Numerical (iterative/approximate) methods: Not *"the"* solution but a *good enough* solution

When to use which?

- exists an analytical approach and if it is computationally feasible → analytical
- otherwise → numerical methods

# How to solve the Bellman equation? (Cntd.)

$$x = f(x)$$

- Drawing (yes, why not?!)
- Midpoint method
- Iterative fixed-point methods
- FfT: Perturbative methods (start from where you know the best!)

# How to solve the Bellman equation? (Cntd.)

## *Exercise*

- *Choose a policy*
- *Evaluate the policy*
- *(Use the evaluation to) Improve the policy*
    - *Unless you are happy(!) go back to Evaluation Step*

**Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$**

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
       *old-action* $\leftarrow \pi(s)$
       $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
       If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

# What if?

- What if the states/actions do not fit into a table?
- What if the environment is not fully observable?
- What if the environment is stochastic?
- What if the optimal policy is stochastic?
- What if the process is not Markovian?
- What if we do not know the dynamics of the environment?

# Summary so far

- Basics of an RL problem
- The cornerstone of RL → Bellmann Eq.
- An iterative evaluation of a policy & Improving the policy to the optimal one

# Monte Carlo


Algorithms relying on repeated random sampling!

Applications: any problem having a probabilistic *interpretation*.

Give me examples!

Examples:

- Finding a probability distribution of dice
- Finding the area of lake
- Finding the value function!

# Monte Carlo Control

## ***Exercise***

- *Choose a policy*
- *Evaluate the policy using MC*
- *(Use the evaluation to) Improve the policy*
    - *Unless you are happy(!) go back to Evaluation Step*

# Q-learning

How could we make the previous algorithm better?

- What was in-efficient?
- How is it different from the way we learn?
- When did it took so long?

Let’s Bootstrap!

Let’s make the best out of what we have learned!

SARSA method

# Q-learning (Cntd.)

“SARSA” method for policy evaluation

*lets derive SARSA together!*

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[ R_{t\square} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \Big].$$

We can use replace the MC policy evaluation with “SARSA” in the general scheme.

But we can also do better: Q-learning!

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[ R_{t\square} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big].$$

# Q-learning (Cntd.)

<u>*Exercise*</u>

**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Take action $A$, observe $R$, $S'$
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until $S$ is terminal

# (D)DQN

## Q-Learning with target networks

Q-learning with replay buffer and target network:

$N\times$ $K\times$

1. save target network parameters: $\phi' \leftarrow \phi$
2. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{B}$
3. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
4. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \underbrace{[r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}'_i, \mathbf{a}'_i)]}_{\text{targets don't change in inner loop!}})$

**supervised regression**