Reinforcement learning

why and how?

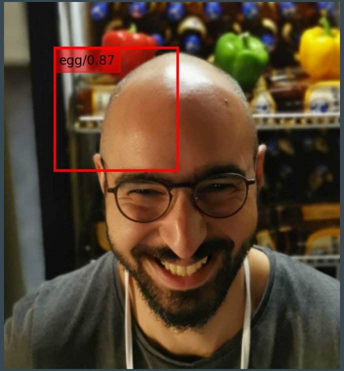
...

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

Intro to myself

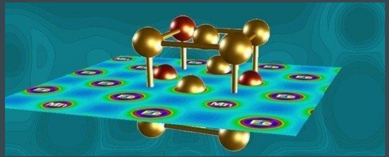
Former position: Machine learning team-lead in DeepMetis
Future position: Senior RL research engineer in InstaDeep

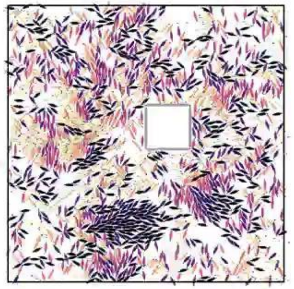
Short CV (in pictures)

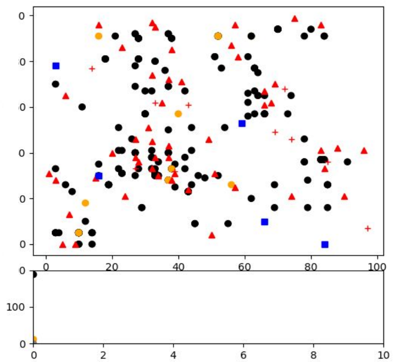


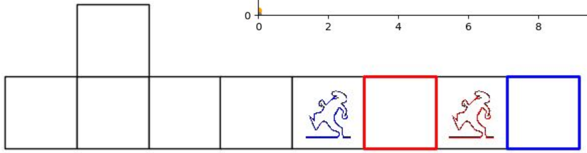
- B.Sc. Mechanical Engineer
- M.Sc. Simulation Science
- PhD in Simulation of Complex Systems
- Postdoc in Simulation of Complex Fluids
- Data Science and Machine learning training

2





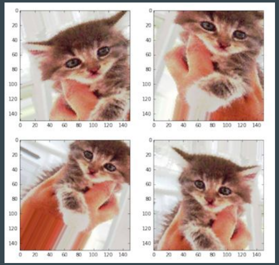


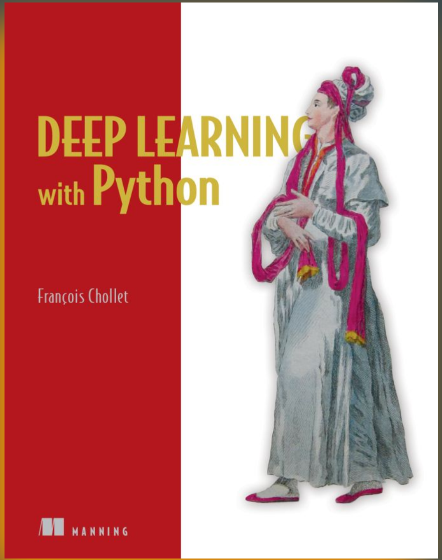


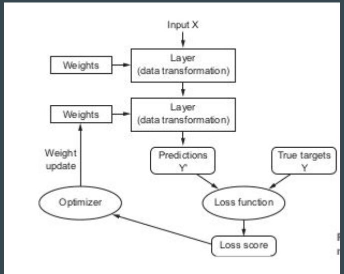
Current Affiliations:

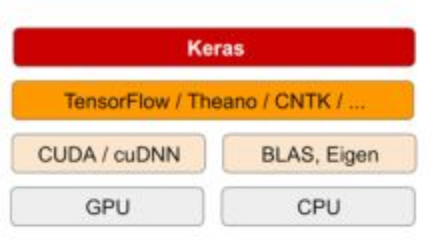
- Data Science Retreat (lecturer)
- Max-Planck-Institute (guest researcher)
- AI-grid

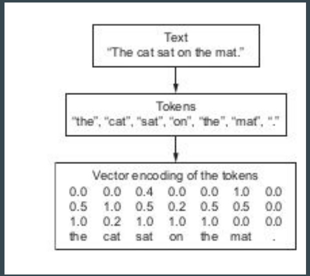
Course disclaimer!

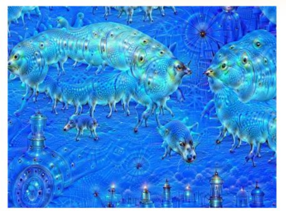


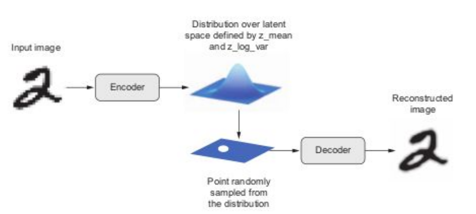












Course outline

- ## Introduction

  What sort of problems you can solve with it? How is it new to you?
  - What is Machine Learning?
  - Different tasks in ML
  - What is Reinforcement, what is Learning?
  - Some examples
  - Course disclaimer
- ## RL problem formulation

  Lots of new terms to be defined and connected to each other
  - RL's basic ingredients
  - RL's problem formulation
  - Exercise
  - Anatomy of a RL solution
- ## Solution of a RL problem

  Zoo of different methods

What is Machine Learning?

Machine Learning: leveraging *data* to perform *tasks*

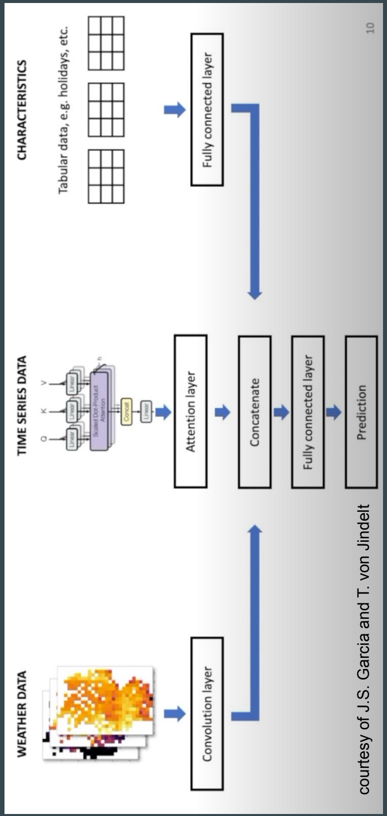
Task

Predicting the consumption and generation of renewable energy

Data

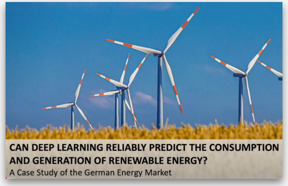
- What should prediction be based on?
    - Weather: wind, temperature, time of sun, pressure,
    - Features: day, week, month, holidays
    - Time series data
- What should the prediction mimic?
    - History
    - Expertise

The knowledge to be *learned*

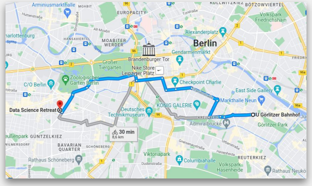


Different Tasks in ML

Machine Learning: leveraging *data* to perform *tasks*



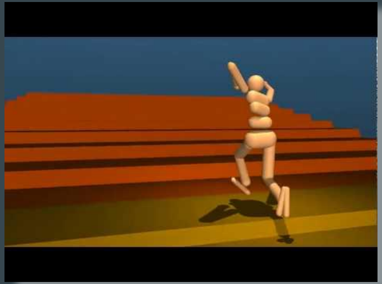
- ***"A"*** **Regression or** ***"A"*** **Classification**



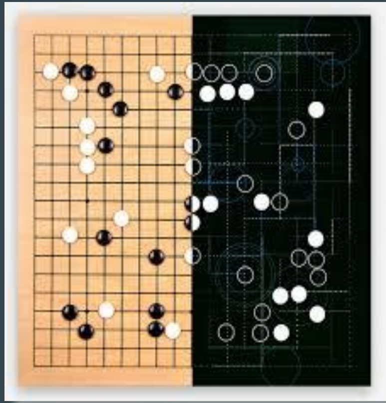
- ***Sequential*** **Decision Making**

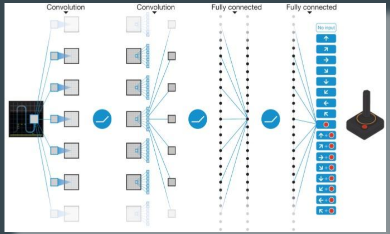
Some examples

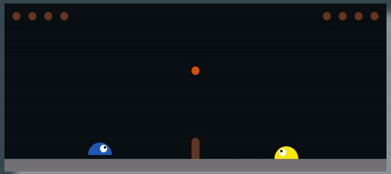
Robotics and Autonomous Systems



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.

Course outline

- Introduction

  What sort of problems you can solve with it? How is it new to you?
  - What is Machine Learning?
  - Different tasks in ML
  - What is Reinforcement, what is Learning?
  - Some examples
  - Course disclaimer
- RL problem formulation

  Lots of new terms to be defined and connected to each other
  - RL’s basic ingredients
  - RL’s problem formulation
  - Exercise
  - Anatomy of a RL solution
- Solution of a RL problem

  Zoo of different methods

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?

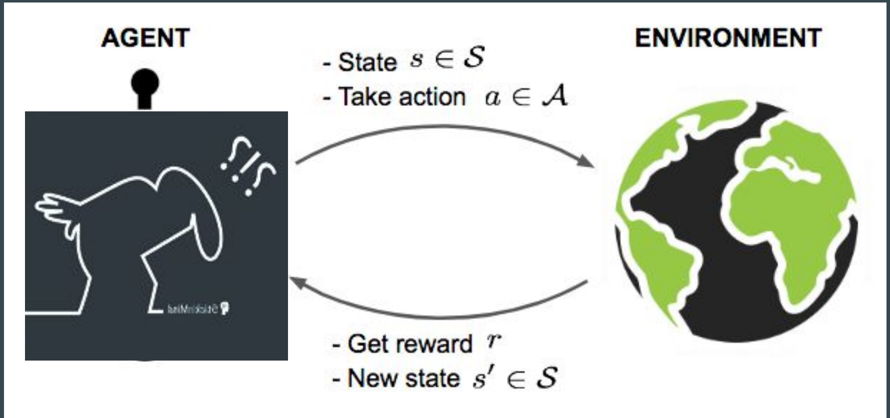
- How rewarding/costly is each action?
  REWARD
- Where do I end up after this action?

Environment

Policy

The mapping between the state and the actions

State —Policy→ Action



RL agent's job

Finding a behavior for maximizing the sum of all rewards, i.e. current reward and what comes after.

Examples

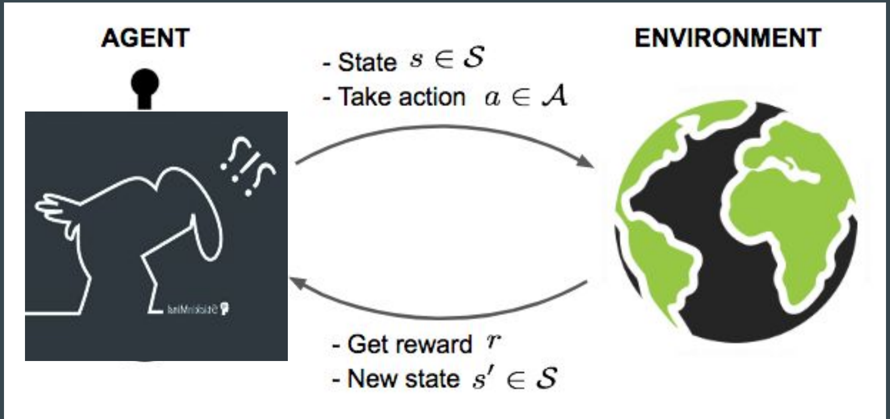
Give me examples!

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

What is RL "Hello World!"?

RL’s problem formulation

Can you cast your decision making problem into a problem for the RL agent?



Reward hypothesis

“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).” *Richard Sutton*

Reward mess up

- Can you think of a badly designed reward?

Anatomy of a RL solution

Find the optimal behavior

- Start with a policy
- Make it better (?!)
  until you reach the optimal policy.

The questions will answer by the end of the course

- How good is a policy? (policy evaluation)
  - Use to policy in interaction with the env.
  - Measure certain quantities to evaluate your policy
- How can you make a policy better? (policy improvement)

Anatomy of a RL solution

Find the optimal behavior

- Start with a policy
- Make it better (?!)
  until you reach the optimal policy.

The questions will answer by the end of the course

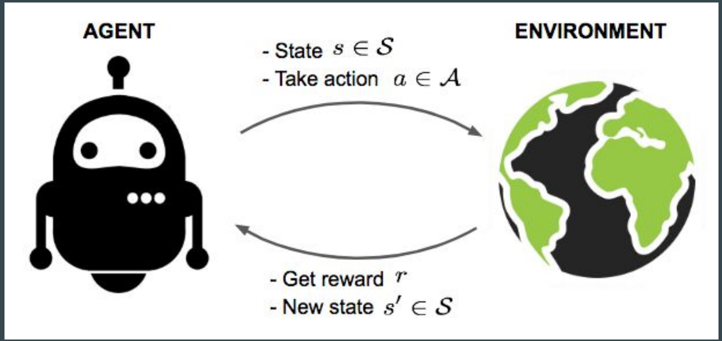
- How good is a policy? (policy evaluation)
  - Use to policy in interaction with the env.
  - Measure certain quantities to evaluate your policy
- How can you make a policy better? (policy improvement)

The rest of the course

- Interaction with Environment
  - The interface between agent and environment
  - Where are the env.? How can I write my own?
- Your 1st Agent
  - Random agent
  - Learning agent
- Policy Evaluation
  - How to represent policy in a computer?
  - Measure of a good policy, Value function
  - Why values are important?
  - Ways to calculate those measures
  - Trial and error: Monte-Carlo
- Policy Improvement
  - A simple algorithm
  - Revisiting policy evaluation (SARSA)
  - Combining eval. and improv. (Q-learning)
- Introducing Deep Neural Nets
  - DQN: back to supervised learning

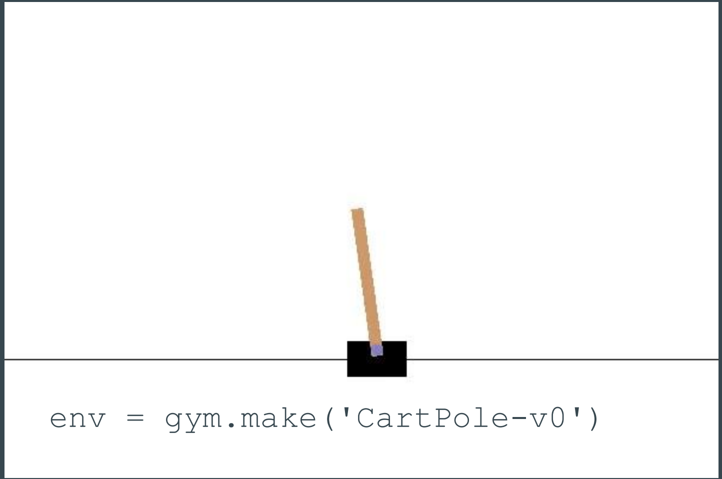
Interaction with Env.

- Let's find a *minimalistic*
  Agent <--> Environment *interface* together!
  Gedankenexperiment: controlling from far far away!

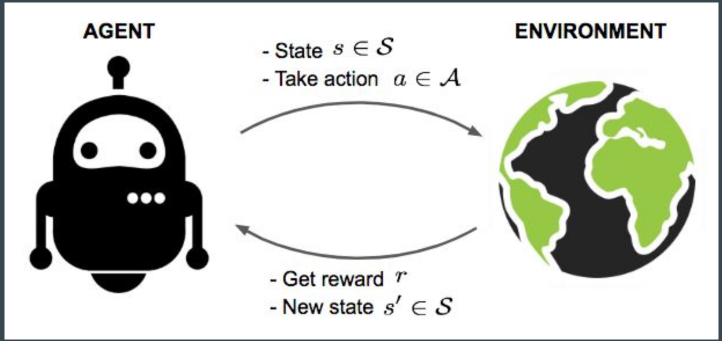


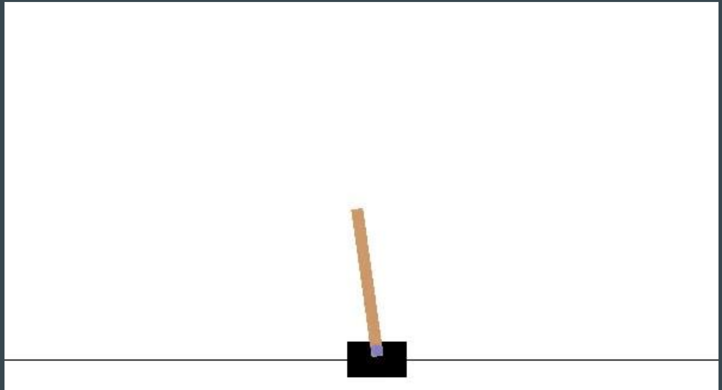
- Where can I find an environment?
- When do I need to write my own?
- Properties of a good env.?

- Let's investigate an environment together!



Your 1st agents

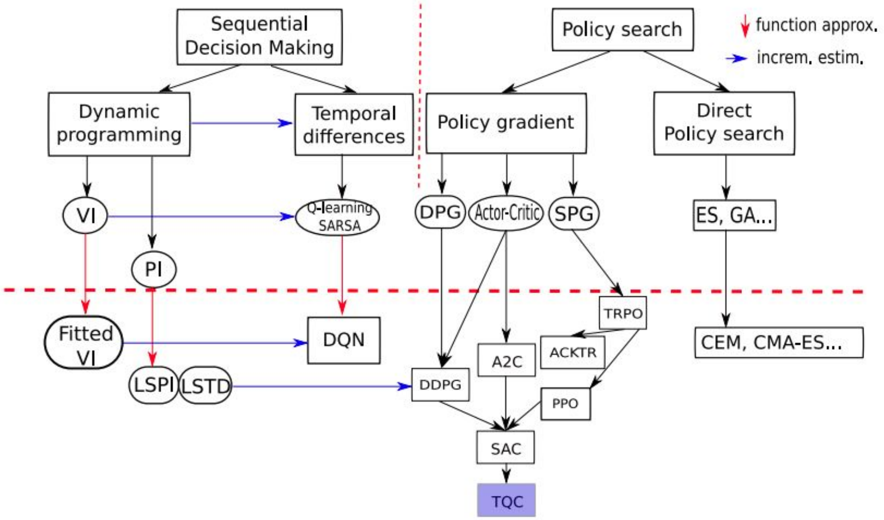




```
env = gym.make('CartPole-v0')
```

An overview of RL-algorithms taxonomy

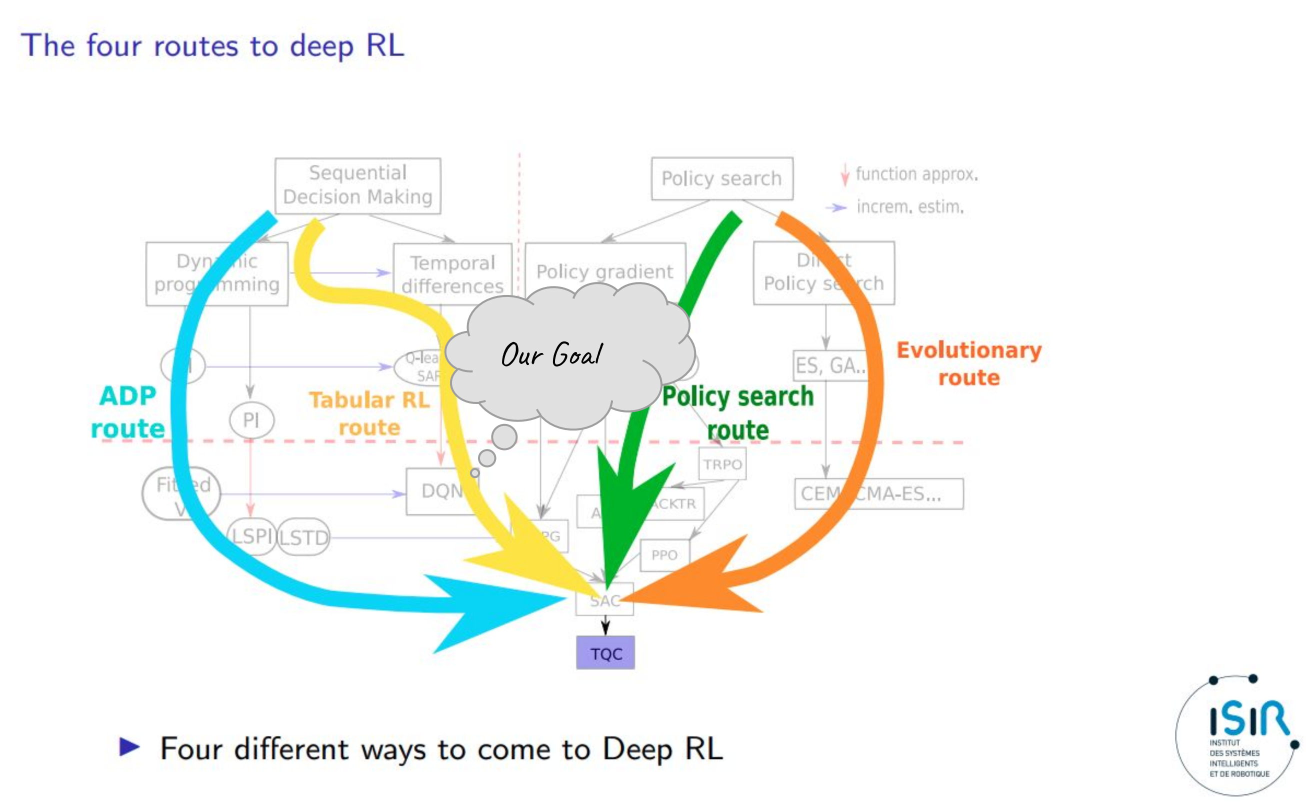
The Big Picture



- A very partial view of the whole RL literature

Olivier Sigaud

An overview of RL-algorithms taxonomy



How to represent a policy?

State $\xrightarrow[\text{Policy}]{?}$ Action

Some possibilities

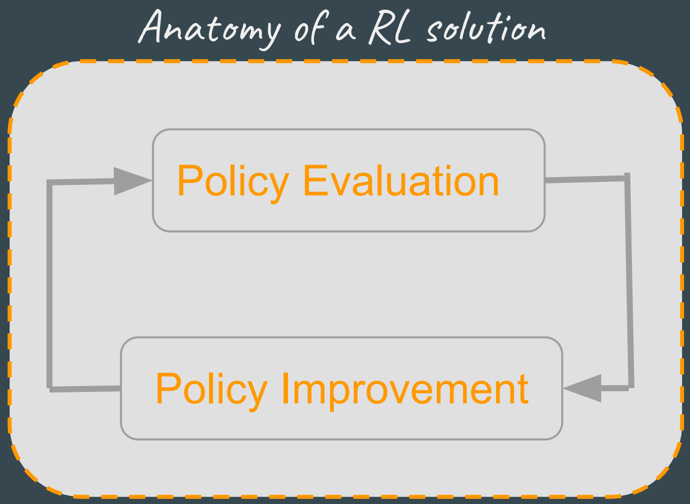
- Table,
- Function, or
- Deep neural network (*Deep RL*)

Different policy types

- Deterministic
- Stochastic

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!

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).

Policy Evaluation Cntd.

Different policy types

- Deterministic
- Stochastic

Q1: How do you evaluate a deterministic policy?

Don’t think of computational cost, we deal with that later.

Q2: What about a stochastic policy?

Q3: Why are stochastic policy important even for deterministic problem?

Policy Improvement

Q-values

Quality of an action:

How good is action **a** from state **s**, if I follow policy **pi** after that action?

Policy improvement

At one state change the policy to argmax Q(s,a)!

Congratulations! You arrived at Monte-Carlo Control

Q1: By the suggested policy improvement, the policy becomes deterministic. Can you suggest some way of making the policy better but staying stochastic?

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.)

<u>Exercise</u>

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

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r \mid s,a)\big[r + \gamma V(s')\big]$
   $\quad$ 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 $\rightarrow$ 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

Monte-Carlo Control

- Choose a policy
- For many iterations:
  - Evaluate the Q values of this policy using MC
  - Improve the policy using the Qs
  - Make the new policy “soft”

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!

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

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

But we can also do better: Q-learning!

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

Q-learning (Cntd.)

Exercise

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 $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-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:

$N\times$

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 $\mathcal{B}$

$K\times$

3. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
4. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \underline{[r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}'_i, \mathbf{a}'_i)]})$

targets don't change in inner loop!

supervised regression