Intorduction to Reinforcement Learning

DOROZHKO Anton

Novosibirsk State University

May 12, 2020

Outline

1. Course overview
2. Introduction
3. Key concepts
4. OpenAI Gym
5. Cross Entropy Method

Class information & Resources

DOROZHKO Anton

Course Instructor

dorozhko.a@gmail.com

Course website : comming soon

Novosibirsk State University

*THE REAL SCIENCE

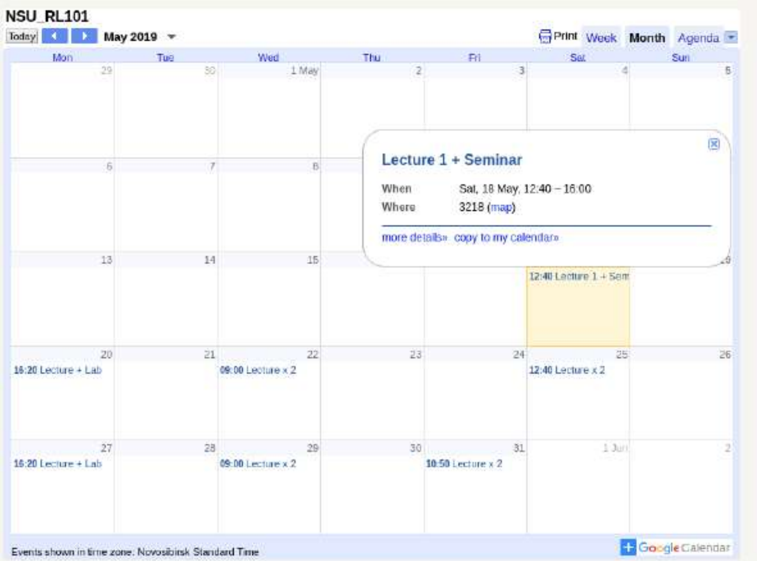
Communications

How to communicate

- We believe students often learn an enormous amount from each other as well as the course staff.
- We will use Piazza to facilitate discussion and peer learning
- Please use Piazza for all questions

Piazza : comming soon

Course logistics



NSU_RL101

Mon	Tue	Wed	Thu	Fri	Sat	Sun
29	30	1 May	2	3	4	5
6	7	8				
13	14	15			12:40 Lecture 1 + Sem	19
20 16:20 Lecture + Lab	21	22 09:00 Lecture x 2	23	24	25 12:40 Lecture x 2	26
27 16:20 Lecture + Lab	28	29 09:00 Lecture x 2	30	31 10:50 Lecture x 2	1 Jun	2

Lecture 1 + Seminar

When Sat, 18 May, 12:40 – 16:00

Where 3218 (map)

more details» copy to my calendar»

Events shown in time zone: Novosibirsk Standard Time

Grading

- Assignment 1 : Math tasks
- Assignment 2 : Q-learning lab
- Assignment 3 : Policy optimization lab
- Project : Read paper + write report (in groups of 2)
- Quiz

Deadlines and Marks to be defined

Preliminary polls

1. What do you know about RL ?
2. Who passed which courses ?
3. What models have you tried to code ?
4. Your level of experience with Python/Tensorflow/PyTorch ?

Outline

1 Course overview

2 Introduction

3 Key concepts

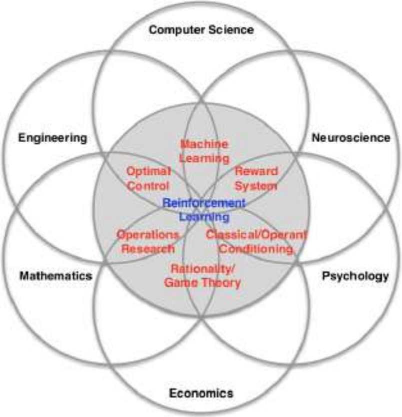
4 OpenAI Gym

5 Cross Entropy Method

What is Reinforcement Learning ?

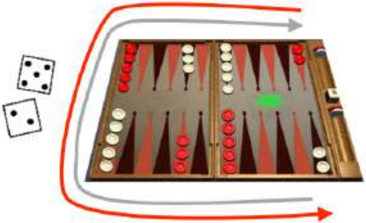
Learn to make good sequence of decisions

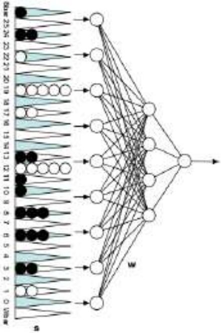
Many Faces of Reinforcement Learning



Example: TD-Gammon

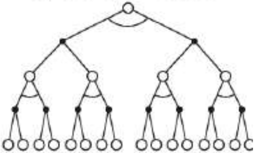
Tesauro, 1992-1995





estimated state value
(≈ prob of winning)

Action selection
by a shallow search



Start with a random Network

Play millions of games against itself

Learn a value function from this simulated experience

Six weeks later it's the best player of backgammon in the world

Originally used expert handcrafted features, later repeated with raw board positions

Why bother learning RL now?

- Interpret rich sensoty inputs
- Choose complex actions

Why bother learning RL now?

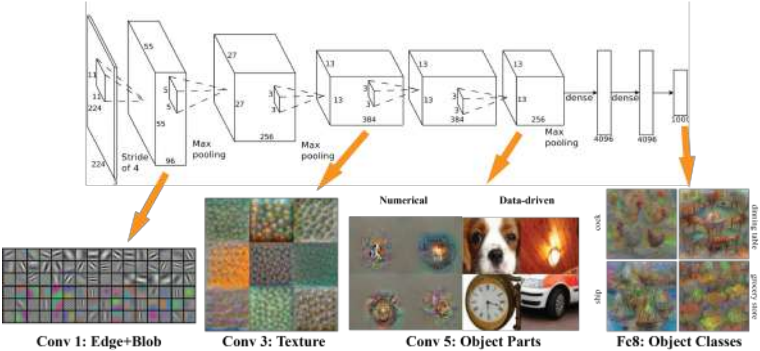
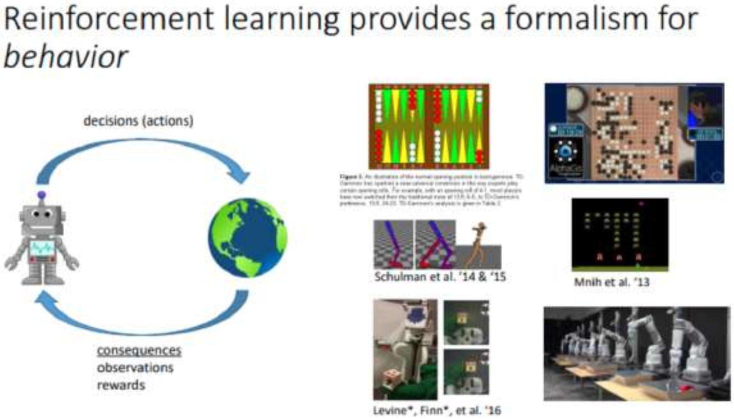


Figure: Deep Learning provides perception

Why bother learning RL now?



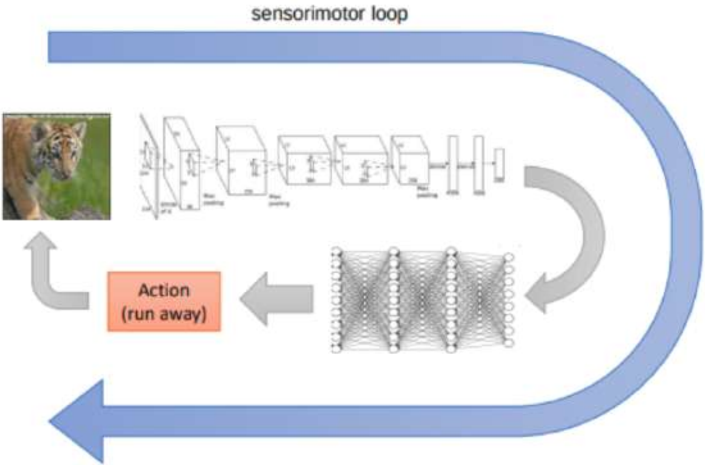
Reinforcement learning provides a formalism for *behavior*

Schulman et al. '14 & '15

Mnih et al. '13

Levine*, Finn*, et al. '16

Deep Reinforcement Learning



Alpha GO and DQN

nature
LEARNING CURVE
Self-taught AI software attains human-level performance in video games
SHARE DATA IN OUTBREAKS
A GIANT IN THE EARLY UNIVERSE
TELEPORTATION FOR TWO

Figure: DQN on Atari games (2015)

Figure: Self-play + MCTS on Go (2016)

OpenAI 5

OpenAI5 blog

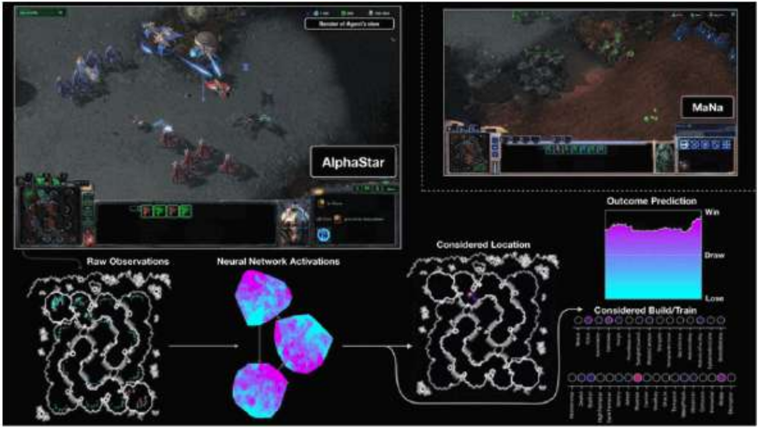
OpenAI 5

OpenAI5 blog

AlphaStar

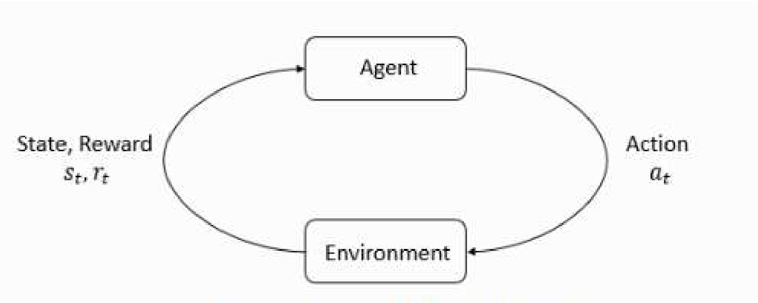
DeepMind blog about AlphaStar

AlphaStar



DeepMind blog about AlphaStar

Environment



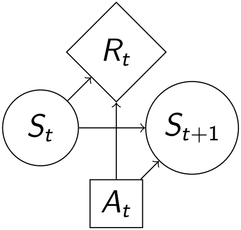
Agent-environment interaction loop.

Reinforcement learning

Markov Decision Process MDP

MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$

1. $\mathcal{S}$ - set of states
2. $\mathcal{A}$ - set of actions
3. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ - *transition function* $p(s_{t+1}|s_t, a_t)$
4. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$- rewards



Markov property

$$p(r_t, s_{t+1}|s_0, a_0, r0, ..., s_t, a_t) = p(r_t, s_{t+1}|s_t, a_t)$$

1

Reinforcement learning

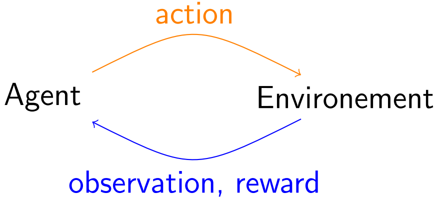
Discounted rewards

$$G_t = R_t + \gamma R_{t+1} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$\max_{\pi_\theta} \mathbb{E}_{\pi_\theta}[G_0]$$

$\pi_\theta : \mathcal{S} \to \mathcal{A}$ - agent policy

Interaction



action

Agent

Environement

observation, reward

- Optimization
- Delayed consequences
- Exploration
- Generalization

Optimization

- Goal is to maximaze the reward
- By finding optimal policy
- Or at least a good policy

Delayed Consequences

- Your current decisions affect your trajectories and future rewards
    - Creating you portfolio
    - Finding key in Montezuma's revenge
- Challenges:
    - Long-term planning
    - Temporal credit assignment (what caused later rewards ?)

Exploration

- Agent learns by making decisions
- Censored data
  - Only have a reward for decision MADE
  - Don't know what would have happened
- Decisions impact learning
  - If we choose to go to another university
  - we will have completely different experience

Generalization

- Policy is mapping: $\mathcal{S} \rightarrow \mathcal{A}$
- Why not just hard code ?

Rewards

- A reward R_t is a scalar feedback
- Indicates how well agent is doing at step t

RL is based on reward hypothesis

Reward hypothesis

All goals can be described by the maximisation of expected cumulative reward

Exaples of Rewards

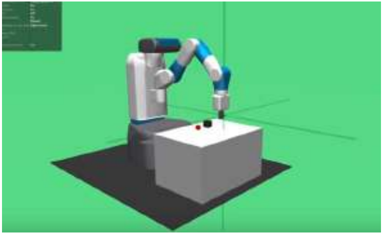
- Fly stunt manoeuvres in helicopter
  - + reward for following desired trajectory
  - - for crashing
- Backgammon
  - + for winning
  - - for losing
- Manage investment portfolio
  - + for making more money
- Make a humanoid robot walk
  - + reward for forward motion
  - - reward for falling over

Teaching agent

- Student initially does not know addition (easier) not subtraction (harder)
- Teaching agent can provide activities about addition or subtraction
- Agent gets rewarded for student performance
  - +1 if student gets problem right
  - -1 if get problem wrong

When optimization gone WRONG

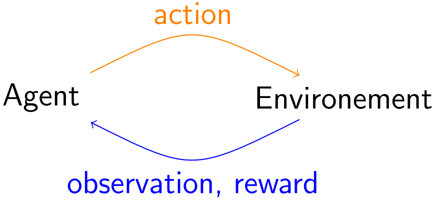
Block moving
A robotic arm trained to slide a block to a target position on a table achieves the goal by moving the table itself.



Other examples: https://bit.ly/2skJE9C

OpenAI Gym [1]

RandomAgent on SpaceInvaders-v0

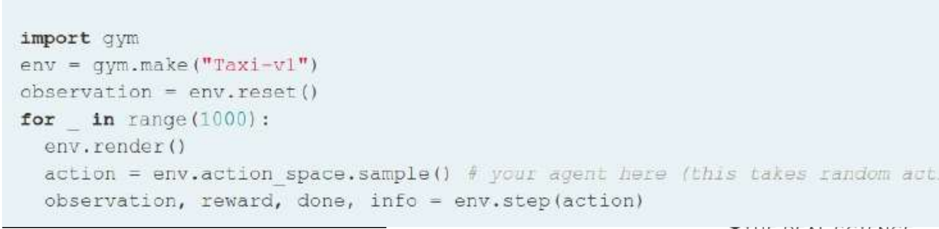


action

Agent

Environement

observation, reward



```
import gym
env = gym.make("Taxi-v1")
observation = env.reset()
for _ in range(1000):
  env.render()
  action = env.action_space.sample() # your agent here (this takes random act
  observation, reward, done, info = env.step(action)
```

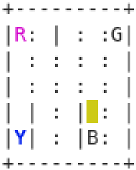
* THE REAL SCIENCE

[1] https://gym.openai.com/

Google Colaboratory

Lab0: https://bit.ly/2YHwUZd

Taxi-v2



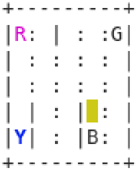
```
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : |█: |
|Y| : |B: |
+---------+
```

Rendering:

- blue: passenger
- magenta: destination
- yellow: empty taxi
- green: full taxi
- other letters (R, G, B and Y): locations

Actions: (0: south, 1: north, 2: east, 3: west, 4: pickup, 5: dropoff)

Taxi-v2



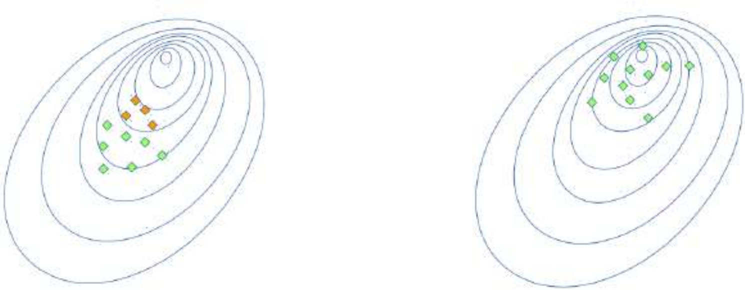
```
+---------+
|R: | : :G|
| : : : : |
| : : : : |
| | : |█: |
|Y| : |B: |
+---------+
```

You receive +20 points for a successful dropoff, and lose 1 point for every timestep it takes. There is also a 10 point penalty for illegal pick-up and drop-off actions.

How do we solve it?

- Play a few rollouts
- Update your policy
- Repeat

CEM visualization [2]



[2] Yandex Practical RL

- Initialize policy (e.g. uniformly)
- Repeat:
  - Sample N rollouts
  - Pick M best
  - Update policy to prioritize best (states, actions)

CEM tabular case

- Policy is a matrix:

$$\pi(a|s) = \mathbb{P}(\text{make action a in state s})$$

CEM tabular case

- Policy is a matrix:

$$\pi(a|s) = \mathbb{P}(\text{make action a in state s})$$

- Sample N games with that policy
- Get best games

$$[(s_0, a_0), (s_1, a_1), \dots, (s_k, a_k)]$$

CEM tabular case

- Policy is a matrix:

$$\pi(a|s) = \mathbb{P}(\text{make action a in state s})$$

- Sample N games with that policy
- Get best games

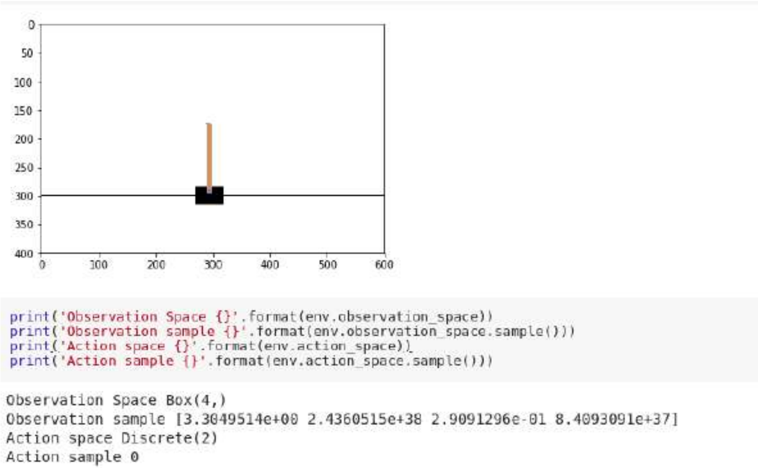
$$[(s_0, a_0), (s_1, a_1), ..., (s_k, a_k)]$$

- Update policy

$$\pi_{t+1}(a|s) = \frac{\sum_{(s,a)\in best}[s_t = s][a_t = a]}{\sum_{(s,a)\in best}[s_t = s]}$$

CartPole-v0

Infinite/large/continuous state space



```
print('Observation Space {}'.format(env.observation_space))
print('Observation sample {}'.format(env.observation_space.sample()))
print('Action space {}'.format(env.action_space))
print('Action sample {}'.format(env.action_space.sample()))
```

```
Observation Space Box(4,)
Observation sample [3.3049514e+00 2.4360515e+38 2.9091296e-01 8.4093091e+37]
Action space Discrete(2)
Action sample 0
```

Approximate Crossentropy

- Approximate function $\pi_\theta(a|s)$
- Linear model / Random Forest / NN

Approximate Crossentropy

- Best state action pairs

$$[(s_0, a_0), (s_1, a_1), ..., (s_k, a_k)]$$

- Maximize likelihood of those tuples

$$\pi = arg \max \sum log \pi(a_i|s_i)$$

Approximate Crossentropy

Initialize NN $w_0 \leftarrow$ **random**

- Sample N rollouts
- Best (s,a) $= [(s_0, a_0), (s_1, a_1), ..., (s_k, a_k)]$
- $w_{i+1} = w_i + \alpha \nabla \sum log \pi(a_i|s_i)$

Approximate Crossentropy

Initialize NN nn = MLPClassifier(...)

- Sample N rollouts
- Best (s,a) $= [(s_0, a_0), (s_1, a_1), ..., (s_k, a_k)]$
- nn.fit(states, actions)