Learning 421 game

Model-Based Learning

Guillaume Lozenguez

@imt-nord-europe.fr

IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

1. **Back to Q-Learning on 421**
2. **ON or OFF policy**
3. **Scale-Up**

Q-Learning: the basics

- Iterative update on (**state**, **action**) evaluation
- Q-Value equation:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left(r + \gamma Q(s',a')\right)$$

- Few parameters:
  α learning rate ; ϵ Exploration-Exploitation ratio and γ discount factor.

Q-Learning: for instance

- Reaching *4-2-1* from *6-2-1* by doing *roll-keep-keep*.

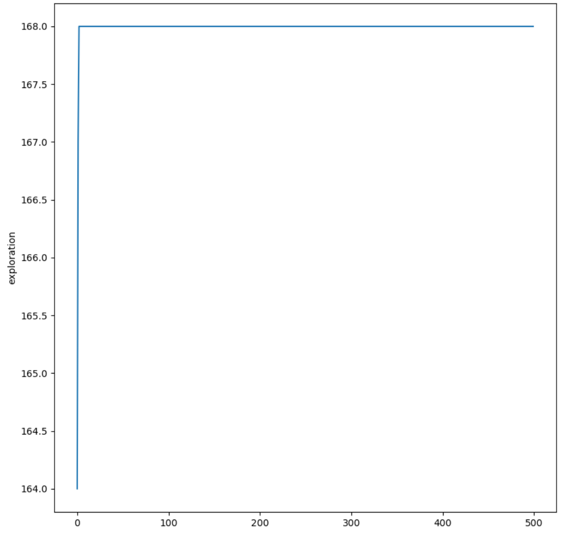
$$Q(\text{6-2-1},\ \text{r-k-k}) = (1-\alpha)Q(\text{6-2-1},\ \text{r-k-k}) + \alpha\left(r + \gamma Q(\text{4-2-1},\ a')\right)$$

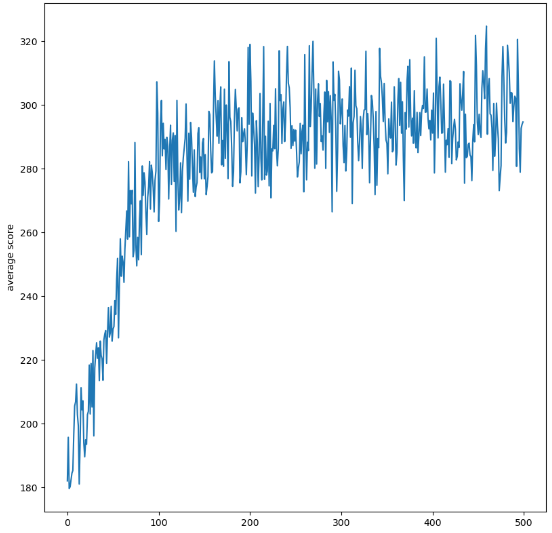
$$Q(\text{2-6-2-1},\ \text{r-k-k}) = (1-\alpha)\,40.0 + \alpha\,(0.0 + 80.0) \quad (a' = \text{keep}^3)$$

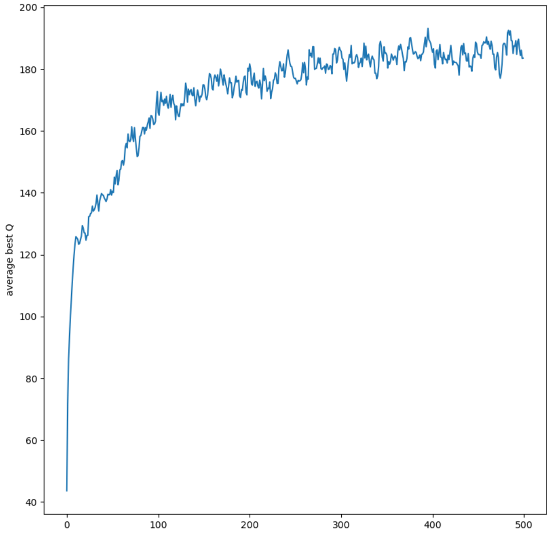
- With α learning rate at *0.1*, $Q(\text{6-2-1},\ \text{r-k-k})$ is now equals to *44*

Q-Learning: the basics

- With 500 steps of 500 games:







- α: 0.1 ; ϵ : 0.1 ; γ : 0.99 ;

Drawing plot in Python: pyplot

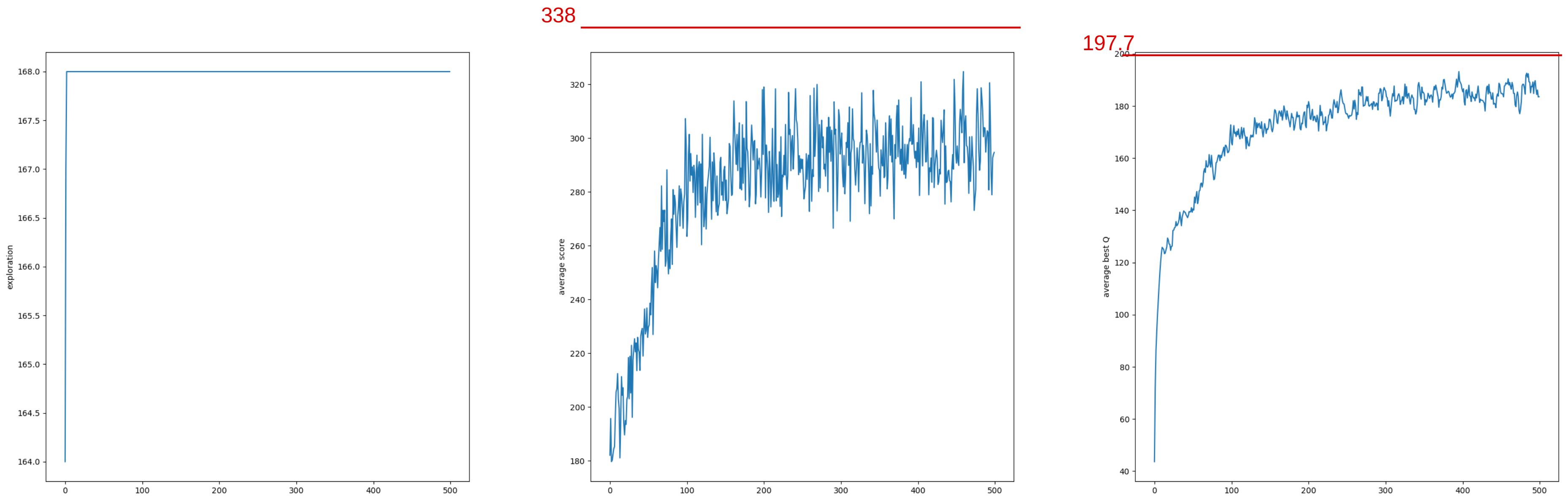
Codes:

```
import matplotlib.pyplot as plt

...

plt.plot( values )
plt.ylabel( "mean of the y value" )
plt.show()
```

- Where `values` is a list of values in $\mathbb{R}$

Q-Learning: the basics

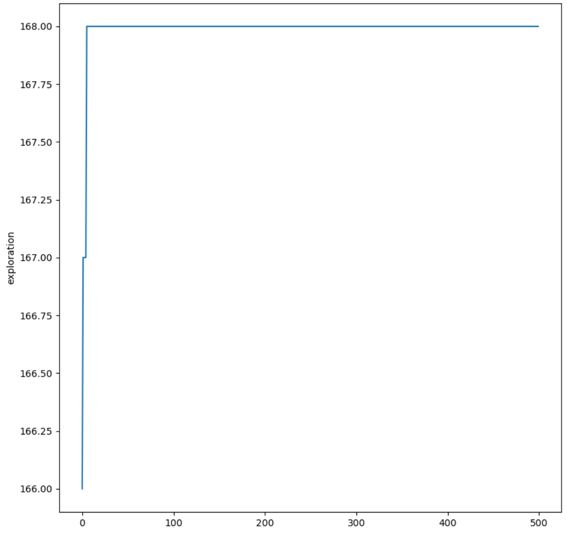
- With 500 steps of 500 games:

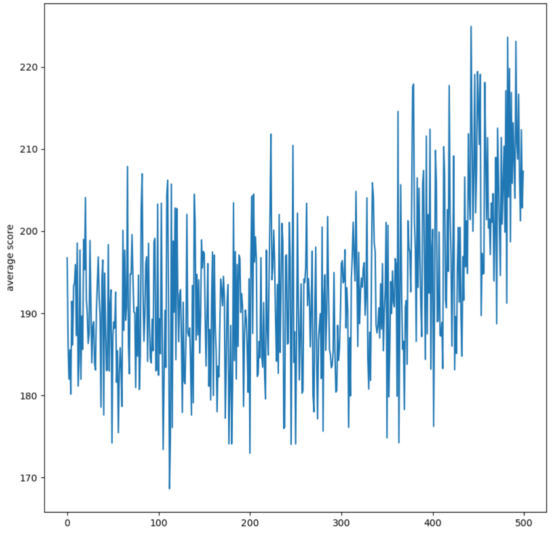


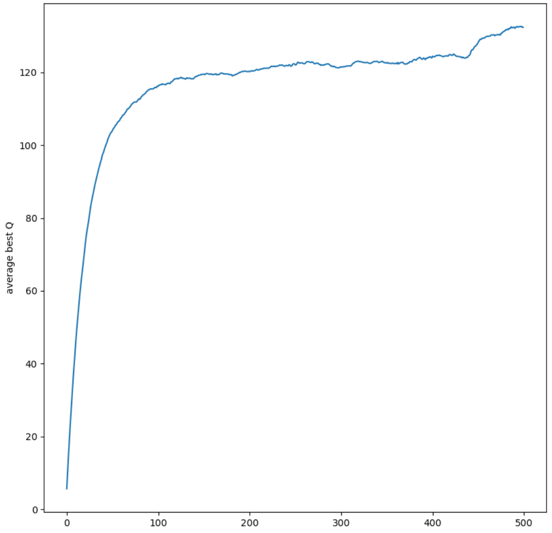
- With optimal threshold

Q-Learning: the basics

- With 500 steps of 500 games:



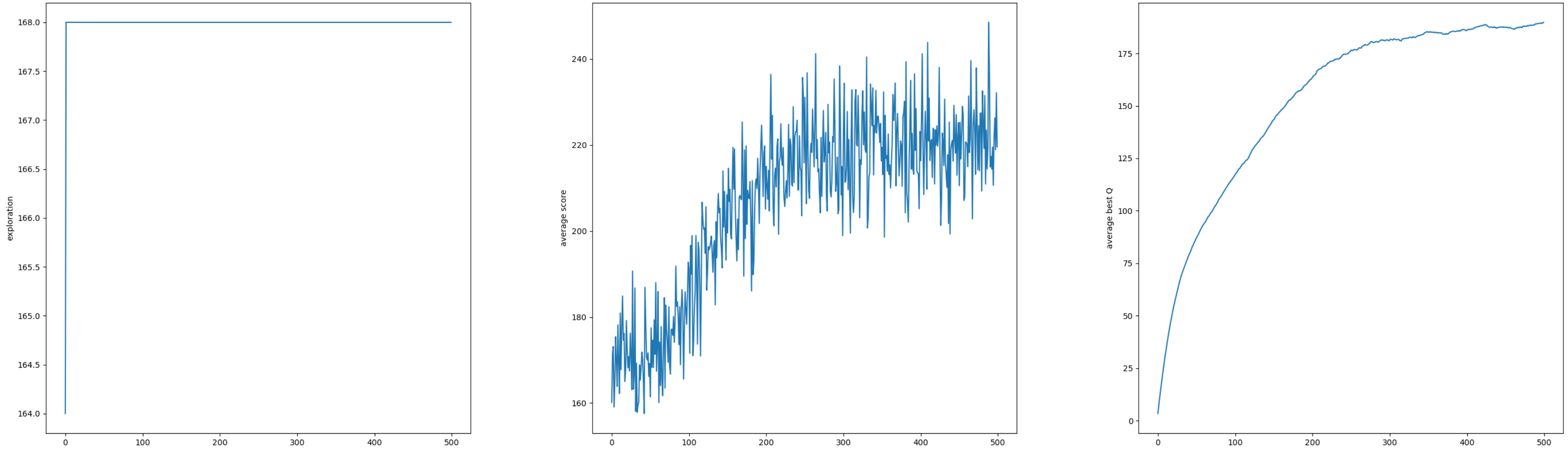




- α: 0.01 ; ϵ : 0.1 ; γ : 0.99 ;

Q-Learning: the basics

- With 500 steps of 500 games:



- α: 0.01 ; ϵ : 0.6 ; γ : 0.99 ;

Playing with the parameters:

- Generate rapidly "good" policies
- Converge on a maximal and stable Q-Values
  (an indicator for optimal policy)
- Potentially: be reactive to system modification (recovery)

Ideally: implement dynamic parameters

1. Back to Q-Learning on 421
2. **ON or OFF policy**
3. Scale-Up

On or Off Policy: The main idea

Does the Q-values match the actual in-use policy ?

- **ON** the *Q-values* and the in-use policy π are aligned.
- **OFF** the learning *Q-values* model something else.
  - Generally π is ϵ-greedy and *Q-values* optimals

On or Off Policy: more formally

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a' \in A} Q(s',a')\right)$$

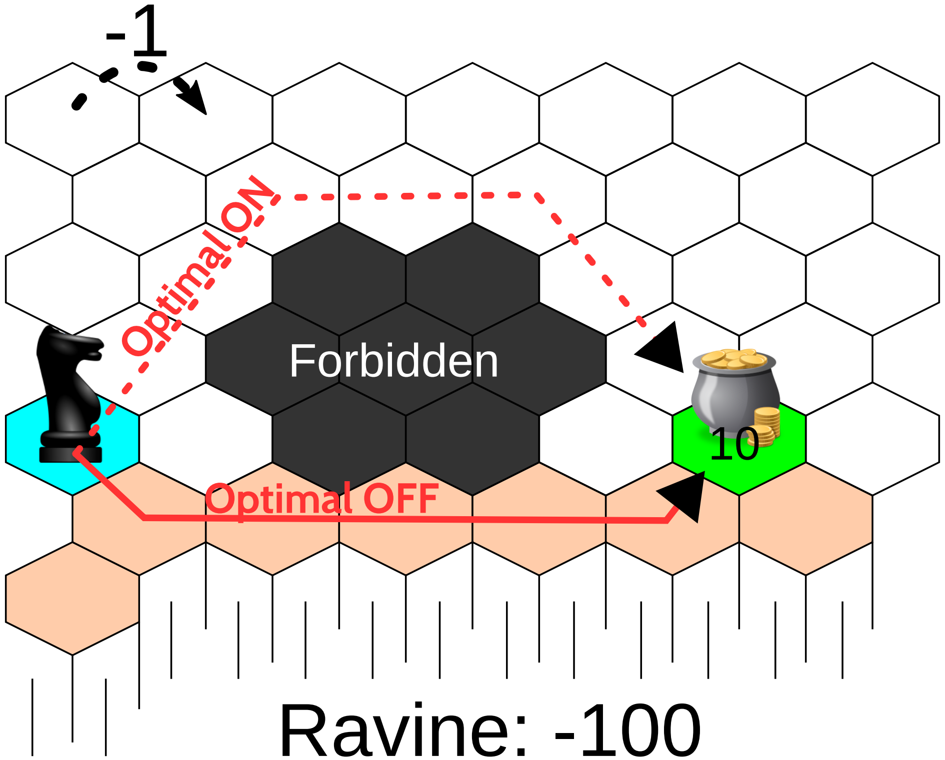
- **ON:**

$$a' = \pi(s') \quad (\text{with} \quad \pi \quad \text{the } \epsilon\text{-greedy policy for instance})$$

- **OFF:**

$$a' = \max_{a' \in A} Q(s',a') \quad \left(\text{and} \quad \pi^*(s) = \arg\max_{a \in A} Q(s,a)\right)$$

On or Off Policy: with an example:



- Deterministic action outcome but 10% to random action.

1. Back to Q-Learning on 421
2. ON or OFF policy
3. **Scale-Up**

Scale-Up:

The central difficulty in Machine-Learning

- You are encouraged to try *QLearning* on 421 Duo mode...
  - From scratch to win random AI.
  - From 421-Solo to win 421-Solo playing Duo.

Scale-Up:

Heuristic values

- Expert-oriented reward function.
- Expert-based qvalue initialization.

Iterative Learning

- Start with few state and action and grow the model.

Hybrid Approach

- Process Q-Learning a record in the same time
- Use punctually heavy 'classical' learning
  - Typically estimate V or T with [illegible]