

# Q Learning applied to 421

Guillaume Lozenguez

[@imt-lille-douai.fr](mailto:@imt-lille-douai.fr)



**IMT Lille Douai**  
École Mines-Télécom  
IMT-Université de Lille

# Q-Learnin: the basics

- ▶ Iterative update on (State, Action) interest.
- ▶ Q-value equation:

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

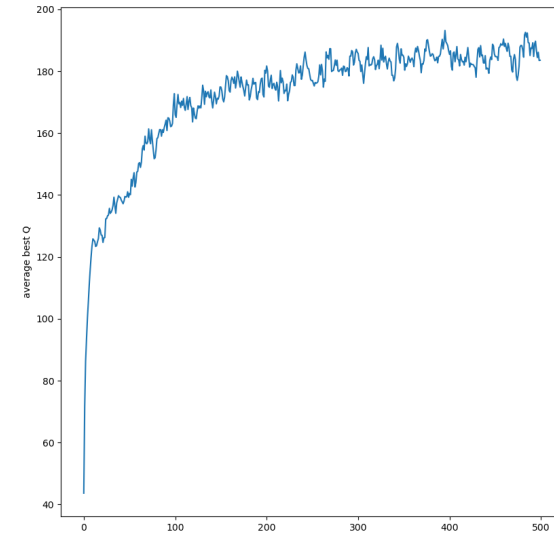
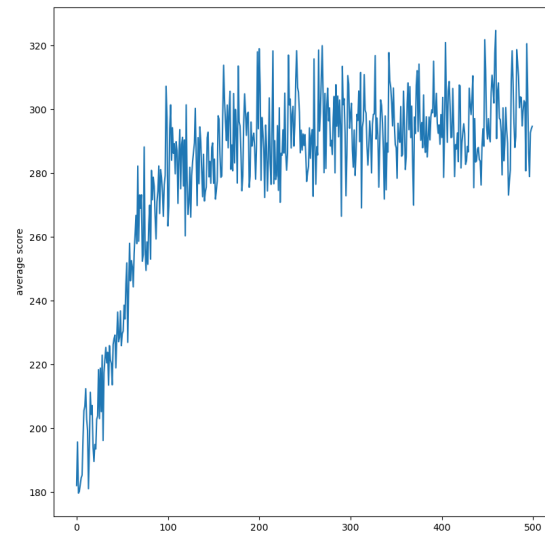
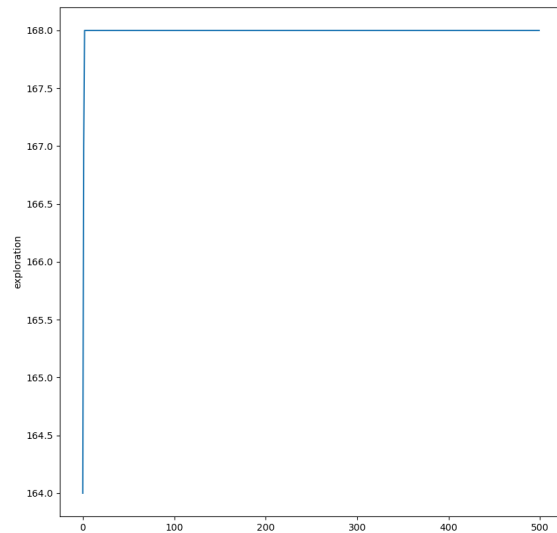
- ▶ Parameters:  
 $\alpha$  - learning rate ;  $\epsilon$  - thexploration-Exploitation ratio ;  $\gamma$  - discount factor

## Q-Learnin: Game 421 (Single PLayer)

- ▶ State Space: Horizon  $\in [2, 0]$ , Dice  $\in [1, 6] \times 3$  : ( $\sim 168$  états)
- ▶ Action Space: **Keep** or **Roll** each dice  $2^3$  : (8 actions)
- ▶ Potentially  $168 \times 8 \times 168$  Transition.
- ▶ Game score (unique final reward): [0 ([2-2-1](#)), 800 ([4-2-1](#))]
- ▶ Random policy score :  **$\sim 170$**
  
- ▶ Correction: [playerQ.py\(raw file\)](#)

# Q-Learnin: Game 421 (Single PLayer)

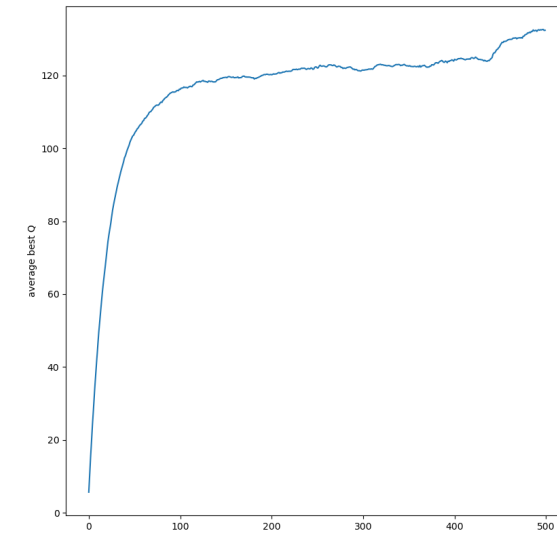
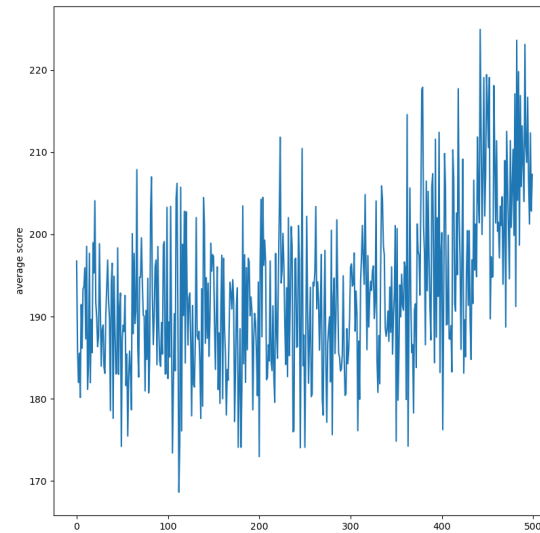
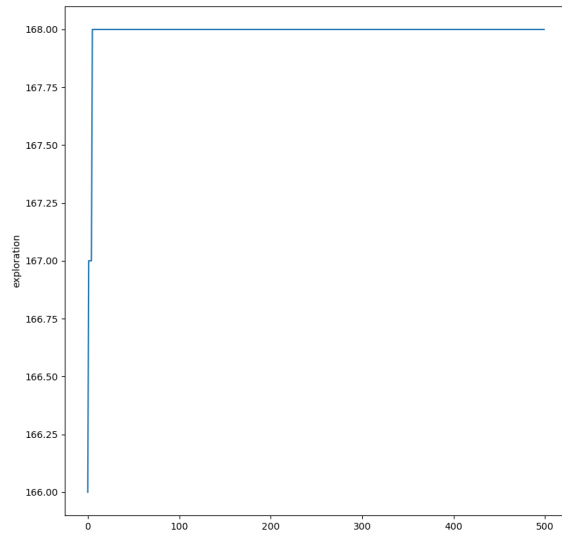
- ▶ With 500 steps of 500 games:



- ▶  $\alpha : 0.1$  ;       $\epsilon : 0.1$  ;       $\gamma : 0.99$

# Convergence: effect of the learning rate

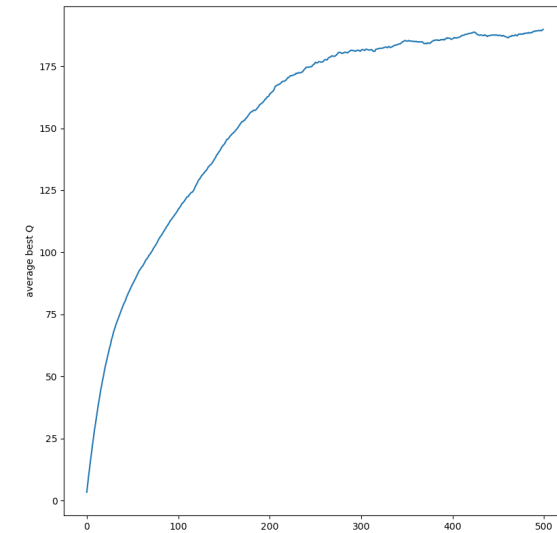
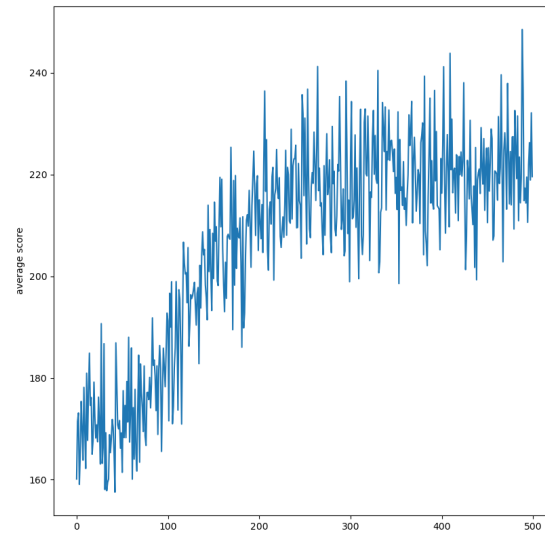
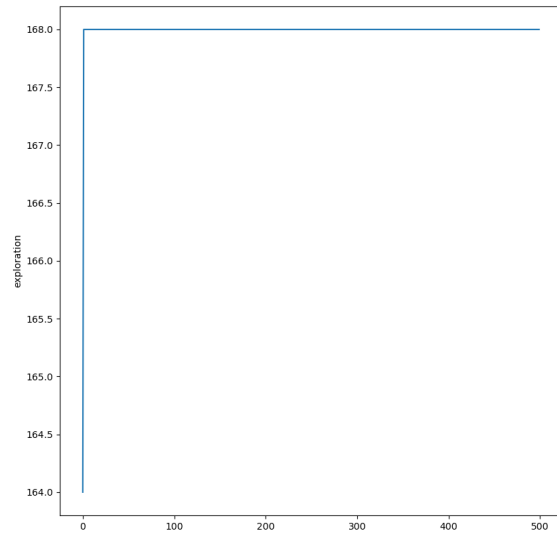
- ▶ With **500** steps of **500** games:



- ▶  $\alpha : 0.01$  ;       $\epsilon : 0.1$  ;       $\gamma : 0.99$

# Convergence: effect of the exploration ratio

- ▶ With 500 steps of 500 games:



- ▶  $\alpha : 0.01$  ;       $\epsilon : 0.6$  ;       $\gamma : 0.99$

# Playing with the parameters:

- ▶ Generated rapidly "good" policies
- ▶ Converge on maximal and stable Q values  
(an indicator for optimal policy)
- ▶ Be reactive to system modification (recovery)  
(no more equiprobable dice for instance)

# Optimize Q-Learning:

## A first solution: use dynamic parameters

- ▶ Balance **learning rate** and **exploration ratio** (or use a stochastic policy) by taking into account known and unknown areas:

*Typically:* Count the number of performed transitions, for each couple of (state, action)

*Problem:* The dynamic will depend on other parameters

*Danger:* Quid of the recovery mode



# Optimize Q-Learning:

## A second solution: use expert knowledge

- ▶ Drive the exploration with an expert knowledge.

*Typically*: initialize the  $Q(s, a)$  with coherent value to take advantage of exploitation from the very beginning.

*Problem*: calibrate the "weight" of the initial knowledge.

*Danger*: Wrong initialization could slow down the learning process.



**About learning directly the model...**