Control theory and Reinforcement Learning - Lecture 3

Carlos Esteve Yague

Universidad Autónoma de Madrid - Fundación Deusto

September 2020

Consider the following optimal control problem:

$$\underset{u_0,u_1,u_3\dots}{\text{minimize}} \quad \sum_{t=0}^{\infty} e^{-t\,h} C(x_t, u_t) \quad \text{subject to} \quad \begin{cases} x_{t+1} = x_t + h\, f(x_t, u_t) & t = 0, 1, 2, 3, \dots \\ x_0 = x \in \mathbb{R}^n. \end{cases}$$

The **value function**:

$$V(x) = \min_{\pi(\cdot)} \sum_{t=0}^{\infty} e^{-t\,h} C(x_t, u_t)$$

Bellman equation:

$$V(x) = \min_{u} \{C(x, u) + \gamma V(x + h\, f(x, u))\}$$

Value iteration: Set an initial guess $V_0(x)$, and then improve the approximation

$$V_{k+1}(x) := \min_{u} \{C(x, u) + \gamma V_k(x + h\, f(x, u))\}$$

By letting $h \to 0^+$ we obtain the continuous version of the OCP:

$$\underset{u(\cdot)}{\text{minimize}} \quad \int_0^\infty \gamma^t C(x(t), u(t)) \quad \text{subject to} \quad \begin{cases} x'(t) = f(x(t), u(t)) & t > 0 \\ x(0) = x \in \mathbb{R}^n. \end{cases}$$

The **value function**:

$$V(x) = \min_{\pi(\cdot)} \int_0^\infty e^{-t} C(x(t), u(t))$$

Bellman equation:

$$V(x) = H(x, \nabla V(x, t))$$

where $H(x, p) := \min_p \{p \cdot f(x, u) + C(x, u)\}$.

Value iteration: Set an initial guess $V_0(x)$, and then improve the approximation with $V(x, t)$ with t large

$$\begin{cases} \partial_t V(x, t) = H(x, \nabla V(x, t)) - V(x) \\ V(x, 0) = V_0(x). \end{cases}$$

Consider the following stochastic optimal control problem:

$$\underset{u_0,u_1,u_3\dots}{\text{minimize}} \quad \mathbb{E}_w\left[\sum_{t=0}^{\infty} e^{-t\,h} C(x_t,u_t)\right] \quad \text{subject to} \quad \begin{cases} x_{t+1} = x_t + h\,f(x_t,u_t) + \sqrt{h}\,w_t \\ x_0 = x \in \mathbb{R}^n. \end{cases}$$

where $w_t \sim \mathcal{N}(0,\, \sigma\, I_n)$.

The **value function**:

$$V(x) = \min_{\pi(\cdot)} \mathbb{E}_w\left[\sum_{t=0}^{\infty} e^{-t\,h} C(x_t,u_t)\right]$$

Bellman equation:

$$V(x) = \min_u \left\{ h\,C(x,u) + \frac{e^{-h}}{\sqrt{(2\pi)^n \sigma}} \int_{\mathbb{R}^n} e^{-\frac{|y|^2}{2\sigma}}\, V(x + h\,f(x,u) + \sqrt{h}y)dy \right\}$$

Value iteration: Set an initial guess $V_0(x)$, and then improve the approximation

$$V_{k+1}(x) := \min_u \mathbb{E}_w \left\{ C(x,u) + \gamma\, V_k(x + h\,f(x,u) + \sqrt{h}w) \right\}$$

By letting $h \to 0^+$ we obtain the continuous version of the stochastic OCP:

$$\underset{u(\cdot)}{\text{minimize}} \quad \mathbb{E}_w \left[\int_0^\infty \gamma^t C(x(t), u(t)) dt \right] \quad \text{subject to} \quad \begin{cases} x'(t) = f(x(t), u(t)) + \sigma \dot{\xi}(t) \\ x(0) = x \in \mathbb{R}^n. \end{cases}$$

The **value function**:

$$V(x) = \min_{\pi(\cdot)} \mathbb{E}_w \left[\int_0^\infty e^{-t} C(x(t), u(t)) \right]$$

Bellman equation:

$$V(x) - \sigma \Delta V(x) = H(x, \nabla V(x, t))$$

where $H(x, p) := \min_p \{p \cdot f(x, u) + C(x, u)\}$.

Value iteration: Set an initial guess $V_0(x)$, and then improve the approximation with $V(x, t)$ with t large

$$\begin{cases} \partial_t V(x, t) - \sigma \Delta_x V(x, t) = H(x, \nabla V(x, t)) - V(x) \\ V(x, 0) = V_0(x). \end{cases}$$

Q-learning

Recall: If we have the value function $V(x)$ (or an approximation of it) we can design an optimal (or nearly optimal) feedback control as

$$u^*(x) = \text{argmin}_u \{C(x,u) + \gamma V(x + h f(x,u))\}$$

However, if the dynamics f is unknown, we cannot make this choice efficiently.

The Q-function

For each state x and control $u \in \mathcal{U}$. We define the Q-function as

$$Q(x,u) := C(x,u) + \gamma V(x + h f(x,u))$$

If we known the Q-function, we can construct the optimal policy without using the dynamics.

Optimal feedback control

$$u^*(x) = \text{argmin}_u Q(x,u)$$

The Q-function

$$Q(x,u) = C(x,u) + \gamma V(x + h f(x,u))$$

represents the total cost if the initial position is x, the first action is u and then we follow an optimal strategy.

Observe that

$$V(x) = \min_u Q(x,u)$$

Dynamic Programming for Q

For all $t = 0, 1, 2, 3, \ldots$

$$Q(x,u) = C(x,u) + \gamma \min_v Q(x + h f(x,u), v).$$

Bellman operator for Q

$$\mathcal{T}Q(x,u) = C(x,u) + \gamma \min_v Q(f(x,u),\ v)$$

We can prove that $\mathcal{T}$ is a contraction in $L^\infty(\Omega)$, with Ω bounded.

$$\begin{aligned}
\mathcal{T}Q_1(x,u) - \mathcal{T}Q_2(x,u) &= \gamma\left(\min_v Q_1(f(x,u),v) - \min_w Q_2(f(x,u),w)\right) \\
&= \gamma\left(\min_v Q_1(f(x,u),v) - Q_2(f(x,u),u^*)\right) \\
&\leq \gamma\left(Q_1(f(x,u),u^*) - Q_2(f(x,u),u^*)\right) \\
&\leq \gamma \left\|Q_1 - Q_2\right\|_{L^\infty}
\end{aligned}$$

By replacing the roles of Q_1 and Q_2 we obtain

$$\left\|\mathcal{T}Q_1 - \mathcal{T}Q_2\right\|_{L^\infty} \leq \gamma \left\|Q_1 - Q_2\right\|_{L^\infty}$$

Q-iteration algorithm

Initialize $Q_0 \in L^\infty(\mathbb{R}^n \times \mathcal{U})$ arbitrarily, and then iterate

$$Q_{k+1}(x,u) = \mathcal{T}Q_k(x,u) = C(x,u) + \gamma \min_v Q_k(f(x,u),\ v)$$

Since $\mathcal{T}$ is a contraction in $L^\infty(\mathbb{R}^n \times \mathcal{U})$, the algorithm converges to the unique fix point of $\mathcal{T}$, which is the solution to the Bellman equation for Q.

$$\lim_{k\to\infty} Q_k(x,u) = Q(x,u).$$

Computationally too expensive! Even when X and $\mathcal{U}$ are finite.

There are pairs $(x,u) \in X \times \mathcal{U}$ that we do not really need to approximate.

Q-iteration algorithm

Initialize $Q_0 \in L^\infty(\mathbb{R}^n \times \mathcal{U})$ arbitrarily, and then iterate

$$Q_{k+1}(x,u) = \mathcal{T}Q_k(x,u) = C(x,u) + \gamma \min_v Q_k(f(x,u),\ v)$$

Since $\mathcal{T}$ is a contraction in $L^\infty(\mathbb{R}^n \times \mathcal{U})$, the algorithm converges to the unique fix point of $\mathcal{T}$, which is the solution to the Bellman equation for Q.

$$\lim_{k\to\infty} Q_k(x,u) = Q(x,u).$$

Computationally too expensive! Even when X and $\mathcal{U}$ are finite.

There are pairs $(x,u) \in X \times \mathcal{U}$ that we do not really need to approximate.

Q-learning, Watkins 1989

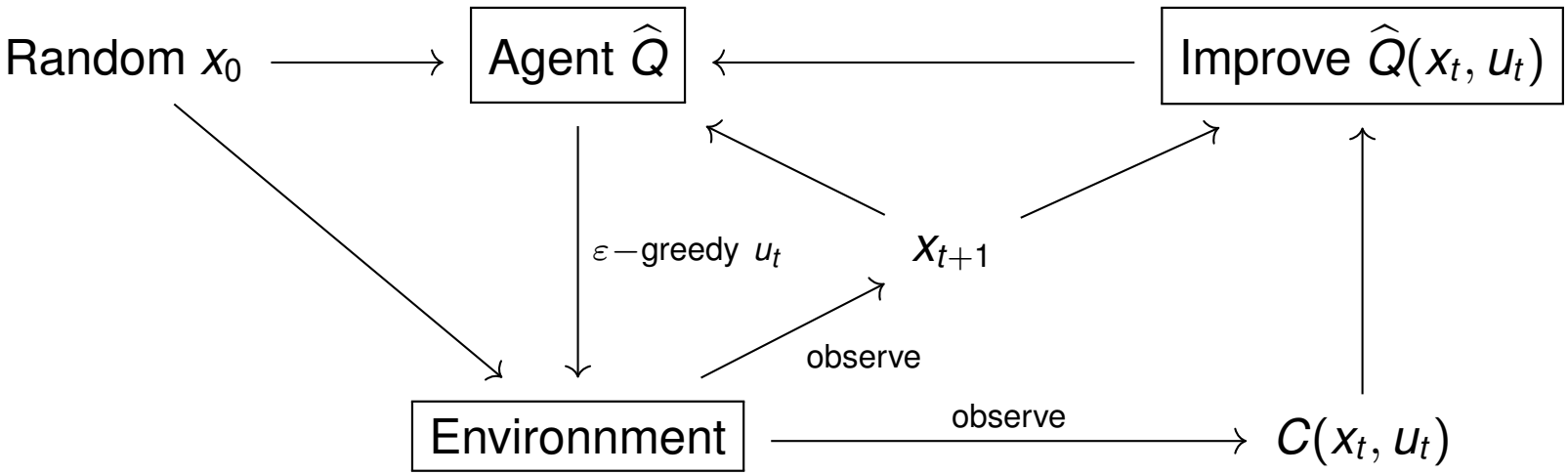
- Initialize an arbitrary Q-function, $\widehat{Q}_0(x, u)$.
- Run a number N of experiments (episodes) of T steps each one.
- In total there will be $N\,T$ number of steps.
- **ε-greedy policy:**

$$u_t = min_u \widehat{Q}_k(x_t,\ u) \quad \text{with probability } 1-\varepsilon$$

and u_k is chosen randomly with probability ε (exploration vs exploitation).

- **Improvement of $\widehat{Q}(x_t, u_t)$:**

$$\widehat{Q}_{k+1}(x_t, u_t) = (1-\alpha)\widehat{Q}_k(x_t, u_t) + \alpha\big(C_t + \gamma \min_u \widehat{Q}_k(x_{t+1}, u)\big)$$



Q-learning (Watkins, 1989)

Algorithm parameters:

- step size $\alpha \in (0, 1]$ and small $\varepsilon \in (0, 1)$
- Number of episodes N and number of steps in each episodes T.

Initialization: Set $\widehat{Q}(x, u)$ arbitrary for all $x \in \mathbb{R}^n$ and $u \in \mathcal{U}$.
For each episode from 1 to N
Initialize $x_0 \in \mathbb{R}^n$
Loop for each step t of the episode, 1 to T
Choose $u_t \in \mathcal{U}$ using an ε-greedy policy.
Input the action u_t and observe C_t and x_{t+1}
$\widehat{Q}(x_t, u_t) \leftarrow (1 - \alpha) Q(x_t, u_t) + \alpha\big(C_t + \gamma \max_u \widehat{Q}(x_{t+1}, u)\big)$
$x_t \leftarrow x_{t+1}$

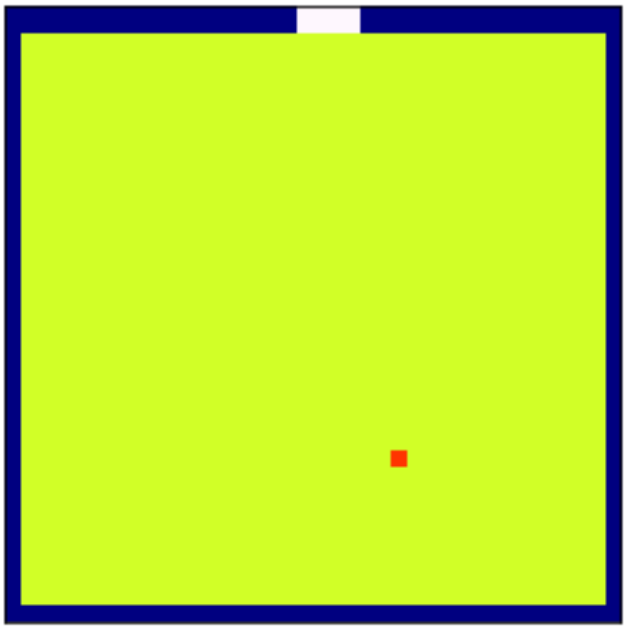
Shortest path to a target

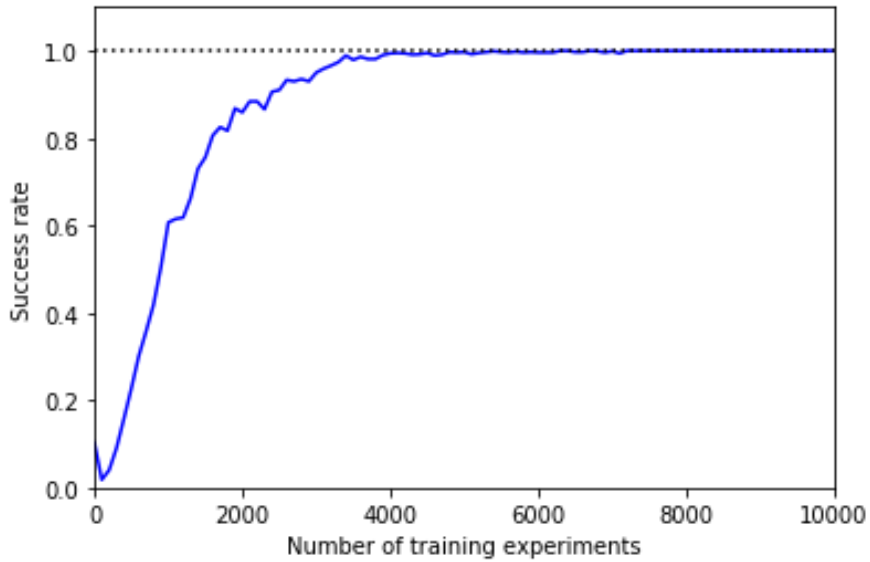
- Set of states: $X = \{1, 2, 3, \ldots, 40\}^2$
- Set of controls: $\mathcal{U} = \{$'up', 'down', 'left', 'right'$\}$
- Dynamics:

$$x_{t+1} = x_t + u_t$$
$$x_0 \text{ randomly chosen.}$$

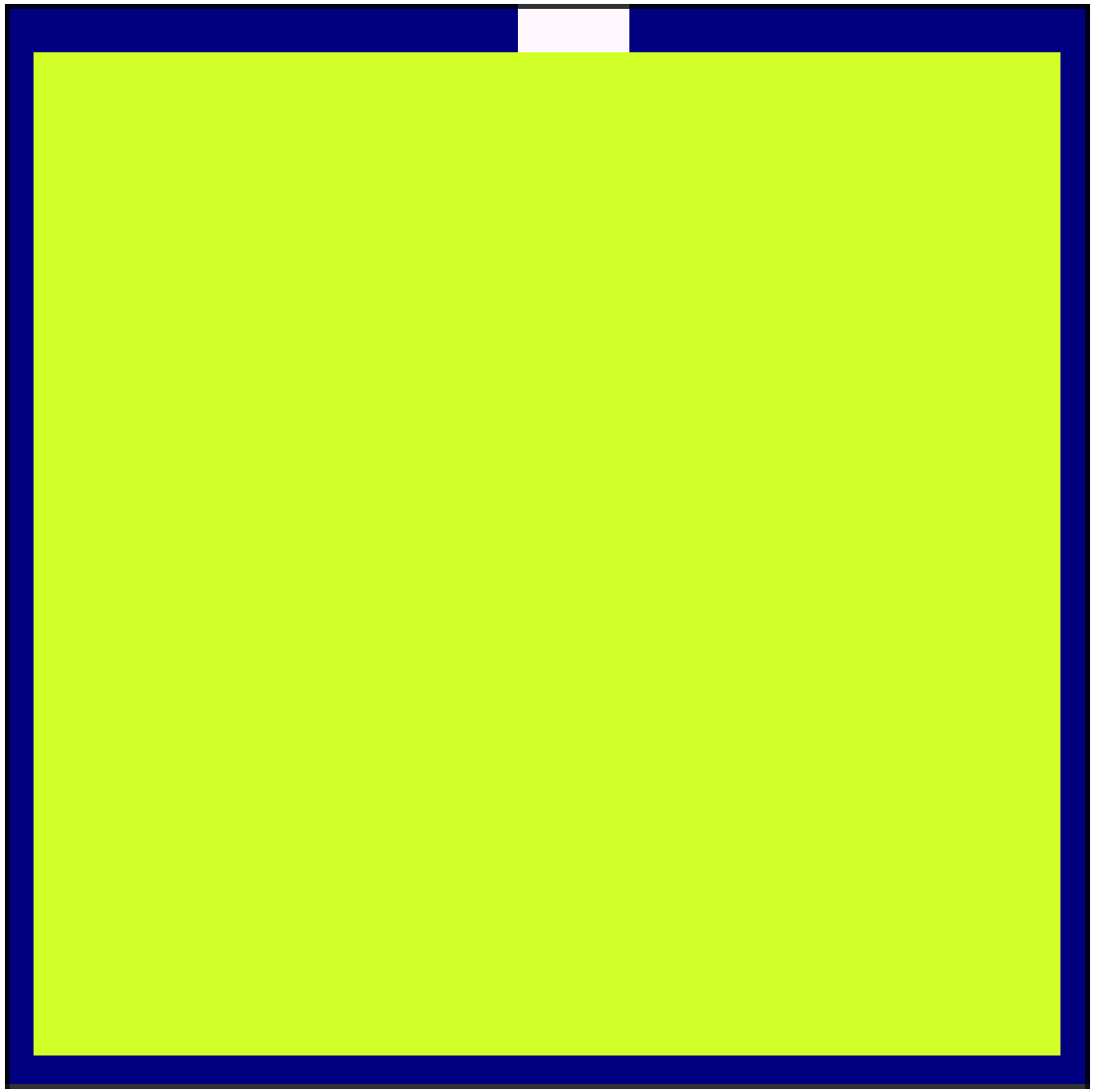
- Cost:

$$C(x_t,\, u_t) := \begin{cases} 1 & \text{if } x_t \text{ is in a green swaure,} \\ 100 & \text{if } x_t \text{ is in a blue square} \\ -100 & \text{if } x_t \text{ is in a white square} \end{cases}$$

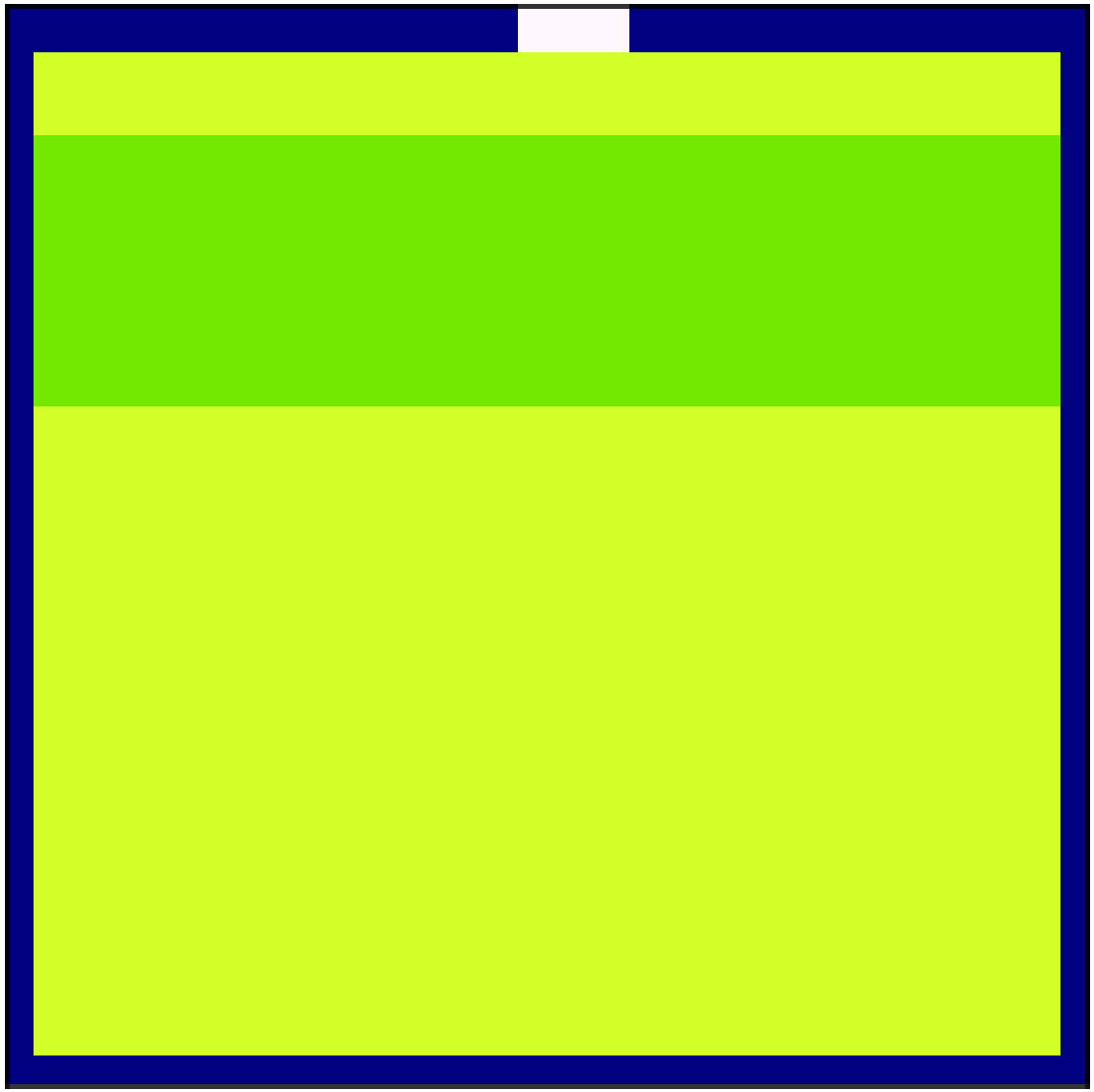




After applying $Q-$learning with 10000 experiments (about 5 seconds), with discount factor $\gamma = 0.9$, learning rate $\alpha = 0.9$ and $\varepsilon-$greedy policy with $\varepsilon = 0.9$.



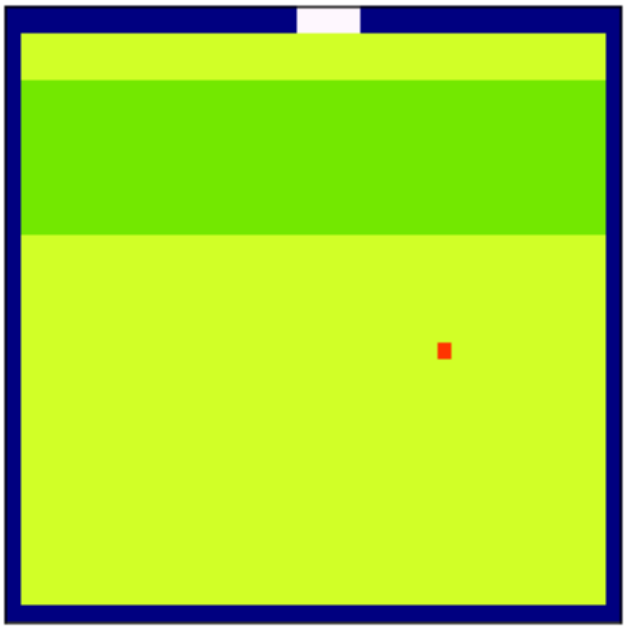
Here we added a ”wind” to the right in the dark area, so the agent often fails to get to the target.

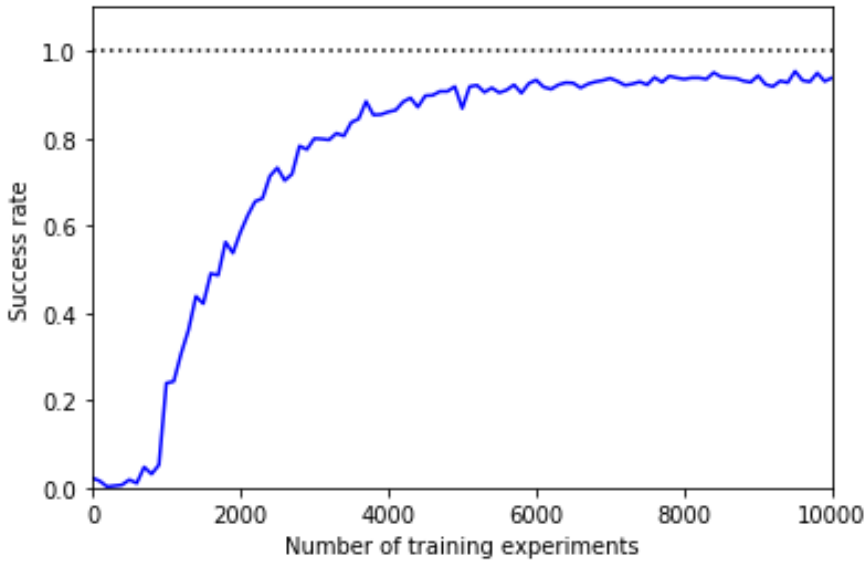


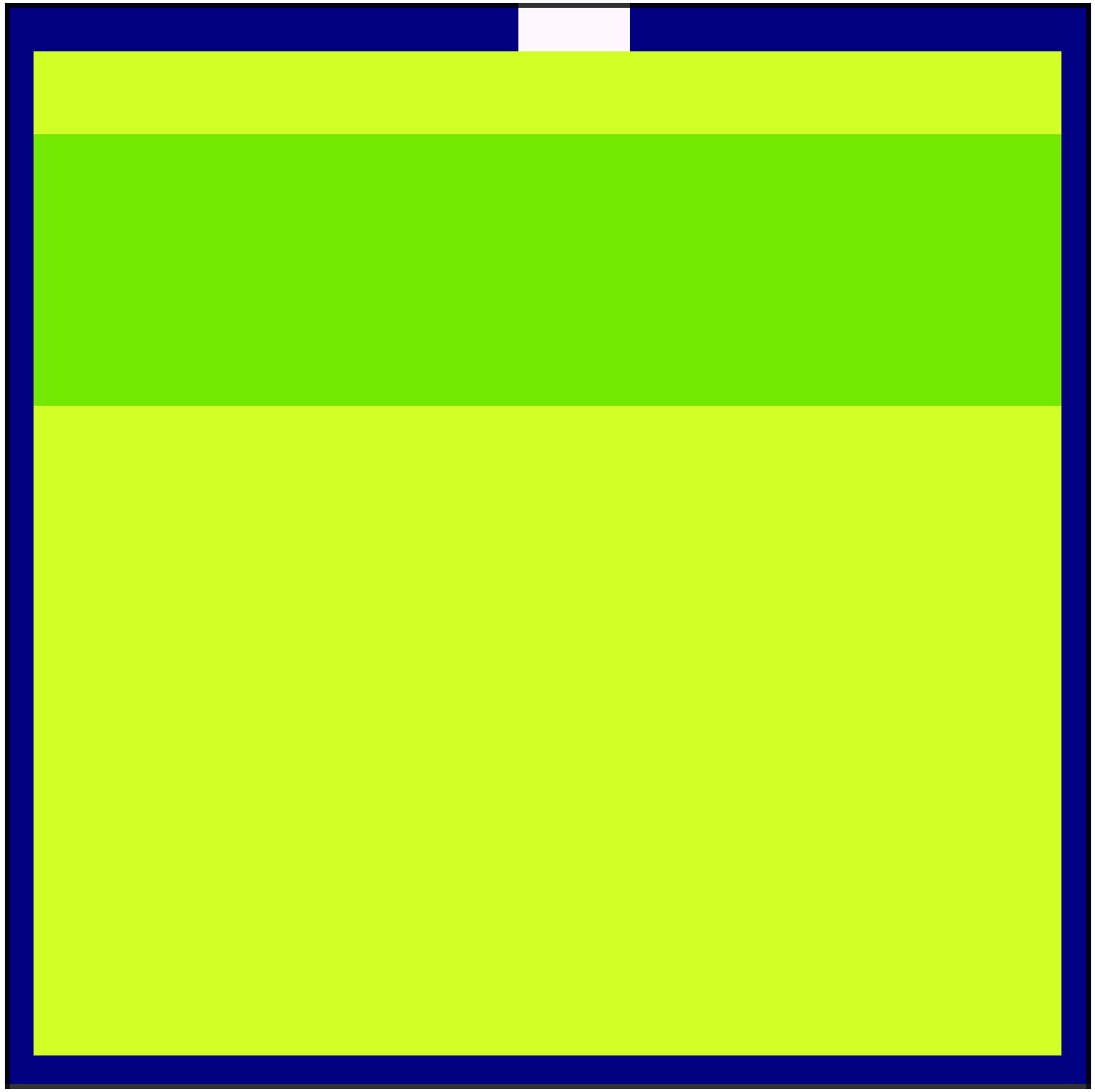
Get to the target with wind

Here we consider the following nonlinear dynamics:

$$x_{t+1} = \begin{cases} x_t + u_t & \text{if } x_t \text{ is in the light area} \\ x_t + 3 + u_t & \text{if } x_t \text{ is in the dark area} \end{cases}$$

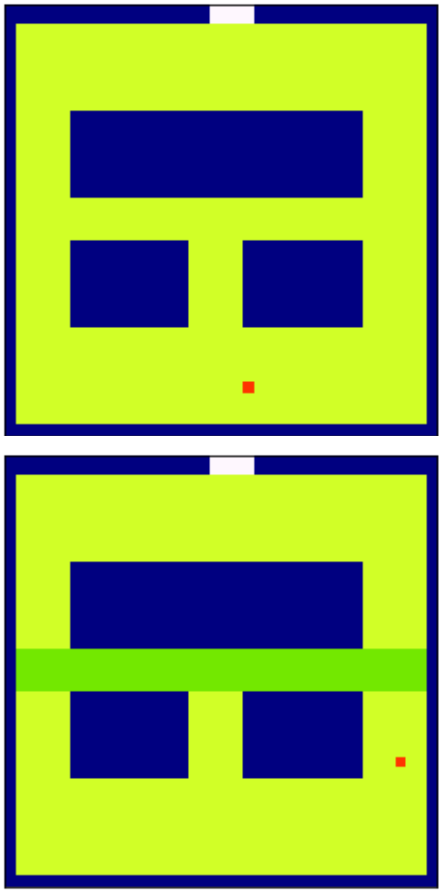


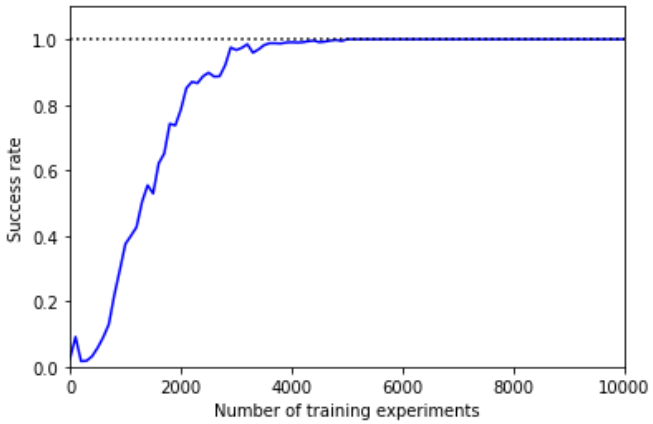


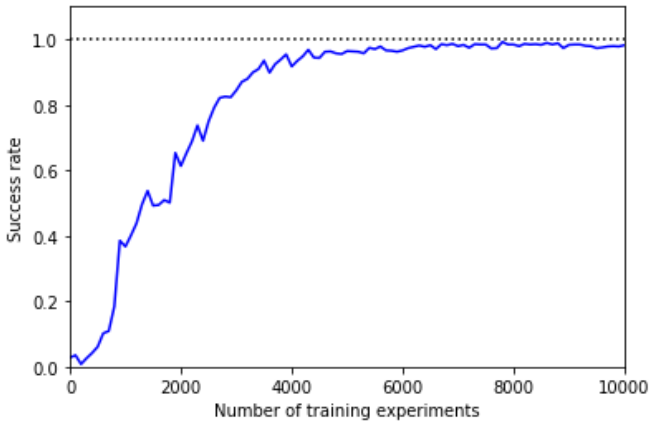


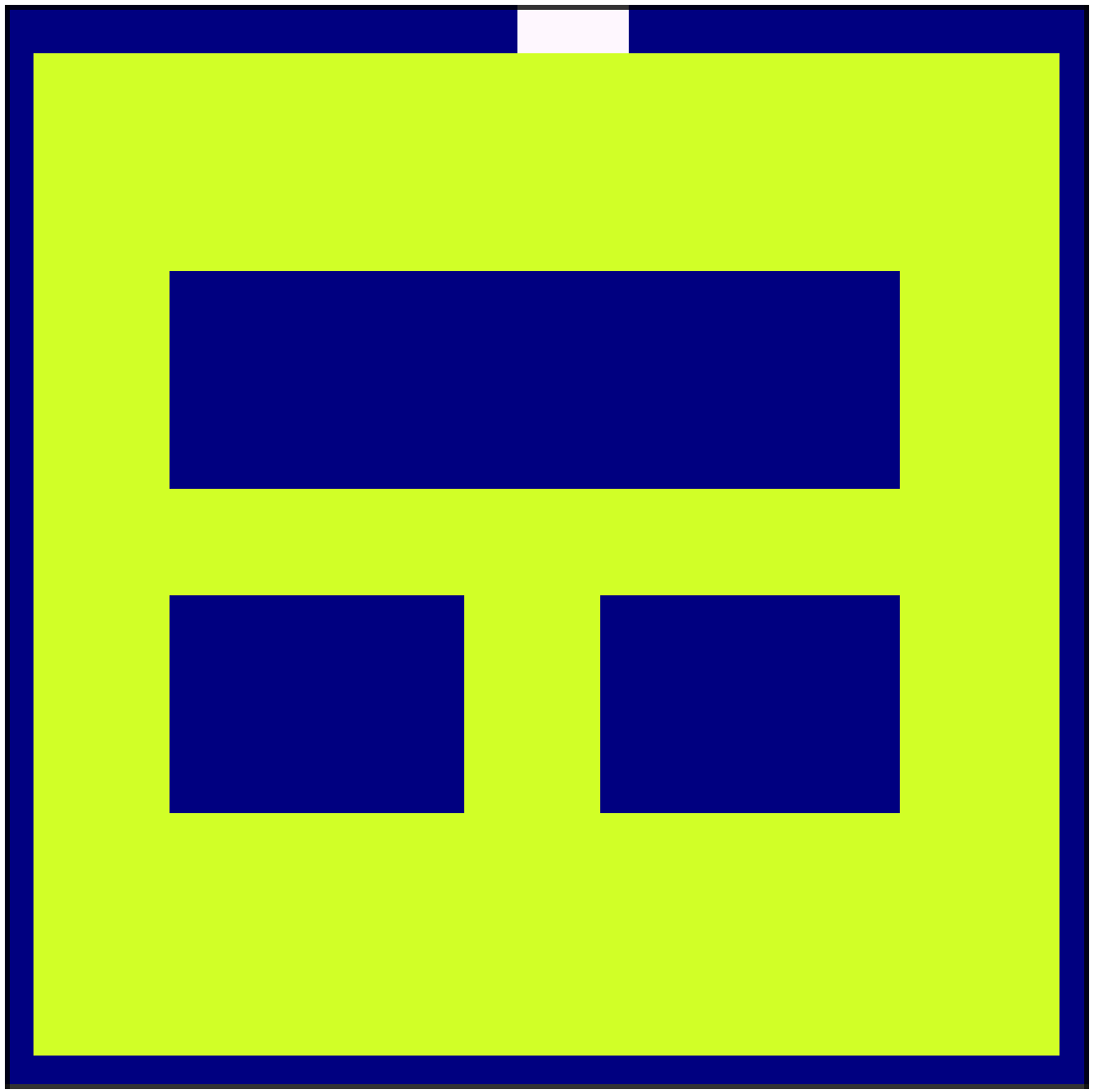
We can put obstacles in the domain by modifying the cost function:

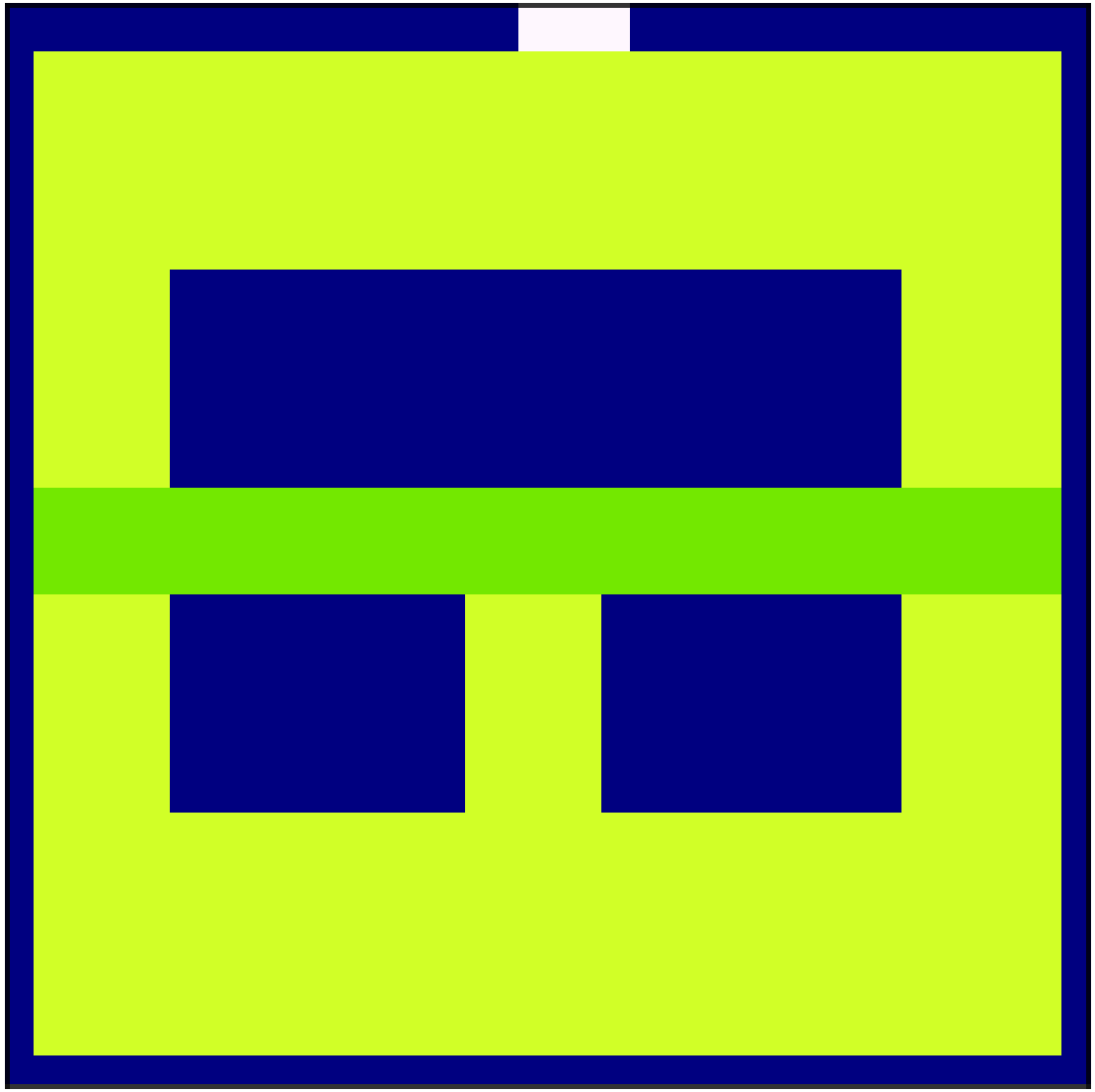
$$C(x_t,\, u_t) := \begin{cases} 1 & \text{if } x_t \text{ is in a green swaure,} \\ 100 & \text{if } x_t \text{ is in a blue square} \\ -100 & \text{if } x_t \text{ is in a white square} \end{cases}$$





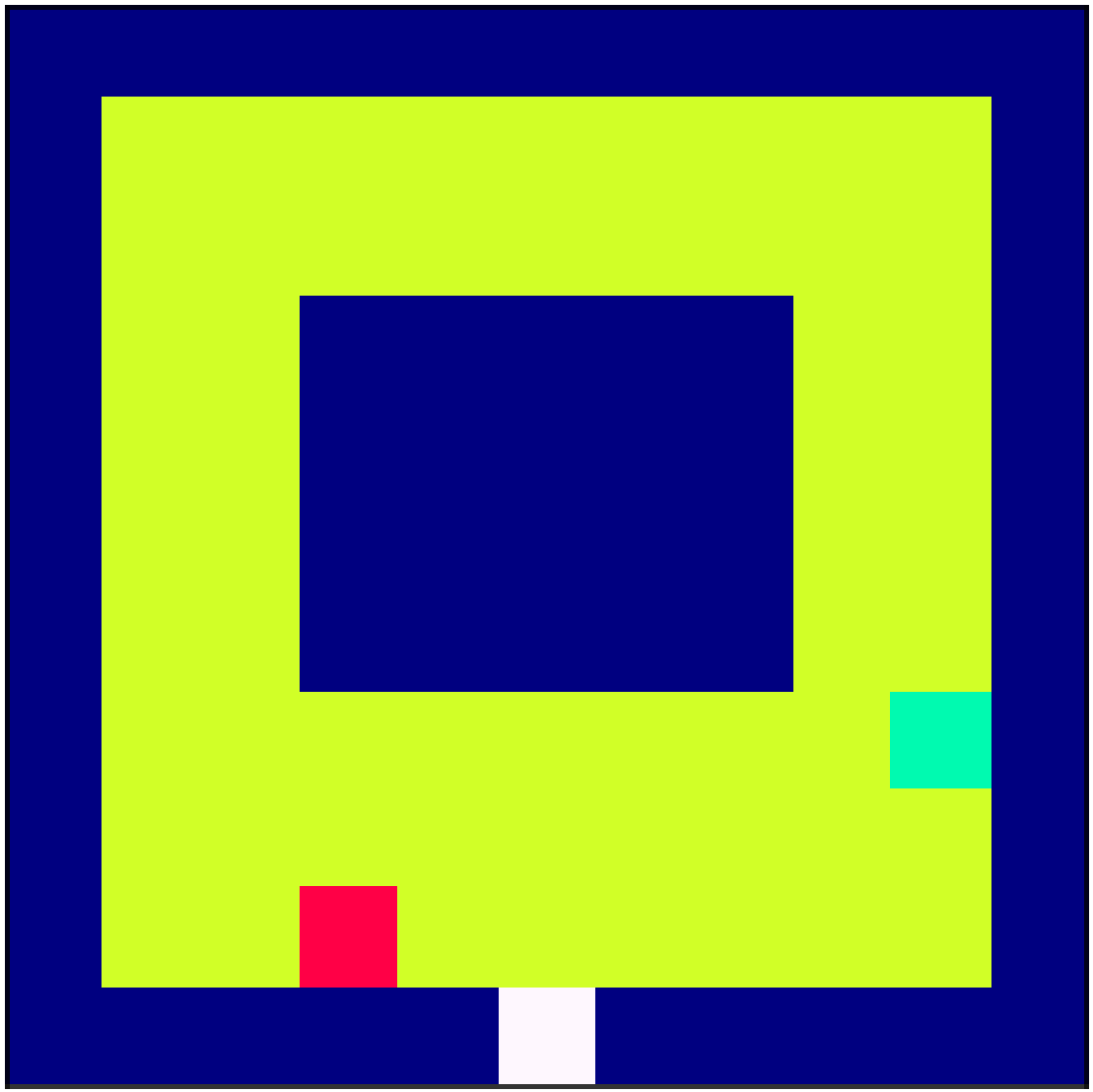






We can use two Q-functions to make two agents compete for two different goals. Here, green agent's goal is two reach the yellow square, and the red's agent goal is to chase the green agent.

It needs however a lot more experiments in the training.



$Q-$learning for the linear quadratic regulator

Consider the optimal control problem

$$\underset{u_0,u_1,u_3\dots}{\text{minimize}} \quad \sum_{t=0}^{\infty} \gamma^{-t}(x_t^* L x_t + u_t^* R u_t) \quad \text{subject to} \quad \begin{cases} x_{t+1} = Ax_t + Bu_t & t = 0, 1, 2 \dots \\ x_0 = x \in \mathbb{R}^n. \end{cases}$$

where $L \in \mathcal{M}_n(\mathbb{R})$ and $R \in \mathcal{M}_m(\mathbb{R})$ are definite positive matrices.

The value function:

$$V(x) = \inf_u \{x^* L x + u^* R u + \gamma V(Ax + Bu)\} = x^* K x,$$

for some definite positive matrix $K \in \mathcal{M}_n(\mathbb{R})$.

The Q-function:

$$\begin{aligned} Q(x,u) &= C(x,u) + \gamma \min_v Q(f(x,u), v) \\ &= x^* L x + u^* R u + \gamma (Ax + Bu)^* K (Ax + Bu) \\ &= [x\ u]^* \begin{bmatrix} L + \gamma A^* K A & \gamma A^* K B \\ \gamma B^* K A & R + \gamma B^* K B \end{bmatrix} [x\ u] \end{aligned}$$

Q-earning for LQR problem

We can write the Q-function as

$$Q(x,u) = [x\ u]^* H [x\ u],$$

where $H \in \mathcal{M}_{m+n}(\mathbb{R})$ is a symmetric matrix given by

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} = \begin{bmatrix} L + \gamma A^* K A & \gamma A^* K B \\ \gamma B^* K A & R + \gamma B^* K B \end{bmatrix}$$

Optimal policy

$$\pi^*(x) = \text{argmin}_u Q(x,u) = \text{argmin}_u \left[x^* H_{11} x + 2x^* H_{21} u + u^* H_{22} u\right]$$

$$\pi^*(x) = -H_{22}^{-1} H_{21} x$$

Then we only need to approximate H from experiments.

$$[x_t\ u_t]^* H [x_t\ u_t] = C_t + \gamma \min_v \{[x_{t+1}\ v]^* H [x_{t+1}\ v]\}$$

Q-learning for the LQR problem

Algorithm parameters:

- step size $\alpha \in (0, 1]$ and small $\varepsilon \in (0, 1)$
- Number of episodes N and number of steps in each episodes T.

Initialization: Choose an arbitrary matrix $H_0 \in \mathcal{M}_{n+m}(\mathbb{R})$.

Iteration: For each episode from 1 to N

Initialize $x_0 \in \mathbb{R}^n$ randomly

Loop for each step t of the episode, 1 to T

Choose $u_t \in \mathcal{U}$ using an ε-greedy policy ($u_t = -H_{22}^{-1} H_{21} x_t$).

Input the action u_t and observe C_t and x_{t+1}

Store all the inputs output pairs in a list

$$\{(x_t, u_t, x_{t+1}, c_t]\}_{t=0}^{T}$$

Using least squares, choose the parameters of the matrix H_{k+1} that better approximate the equation

$$[x_t\ u_t]^* H_{k+1} [x_t\ u_t] = (1-\alpha)[x_t\ u_t]^* H_k [x_t\ u_t] + \alpha \left(C_t + \gamma \min_v [x_{t+1}\ v]^* H_k [x_{t+1}\ v] \right)$$