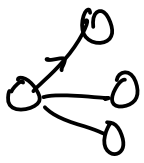
Q-learning

optimal control $\quad \underset{}{\text{argmin}} \sum_{t=0}^{T} J_t(x_t, u_t)$

s.t. $x_{t+1} = f(x_t, u_t)$ — unknown

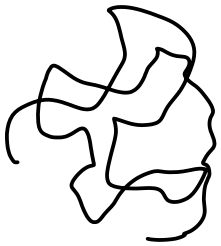
LQR we assumed system dynamics to be known and linear

Dykstra or A-star we assumed dynamics to be deterministic and known



Learning function from data

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{matrix} x_t + v\cos\theta_t dt \\ y_t + v\sin\theta_t dt \\ \theta_t + \omega_t dt \end{matrix}$$



Data D

$$\begin{cases} x_t, u_t \longrightarrow x_{t+1} \\ x_1, u_1 \longrightarrow x_2 \\ x_2, u_2 \longrightarrow x_3 \\ \vdots \\ x_{100}, u_{100} \longrightarrow x_{101} \end{cases}$$

Solution: learn the function f = $Ax_t + Bu_t$ (To be learnt)

$$\min_{A,B} \sum_{x_t, u_t, x_{t+1} \in D} \| x_{t+1} - (Ax_t + Bu_t) \|_2^2$$

$f = NN \longrightarrow$ LQR, Dykstra, A-star

Model-based learning

system dynamics

Model-free learning

Solving the optimal control problem without learning system dynamics

In LQR

Cost to go (Value function)

$$V_T(x_T) = J_T(x_T)$$

$$V_t^*(x_t) = \sum_{k=t}^{T} J_k(x_k, \pi^*(x_k))$$

Optimal cost to go

$$V_t^*(x_t) = \min_{u_t} J_t(x_t, u_t) + V_{t+1}^*(x_{t+1}) \qquad f(x_t, u_t)$$

LQR Dijkstra

Bellman equation

The main idea is called Dynamic programming

$$f_i = f_{i-1} + f_{i-2}$$

Recursion inefficient

```
def fib(n):
    if n == 1
        return 1
    else
        return fib(n-1)
               + fib(n-2)
```

DP efficient

```
f_obs = [1, 1]
for i in range(n)
    f[i+2] = f[i] + f[i+1]
```

Q-function = Action value function (max)

cost = min
rewards = max

$$V_t^*(x_t) = \sum_{k=t}^{T} J_k(x_k, \pi^*(x_k))$$

$$Q_t^*(x_t, u_t) = J_t(x_t, u_t) + \sum_{k=t+1}^{T} J_t(x_k, \pi^*(x_k))$$

$$V_t^*(x_t) = \min_{u_t} Q^*(x_t, u_t)$$

$$Q_t^*(x_t, u_t) = J_t(x_t, u_t) + V_{t+1}^*(x_{t+1})$$

policy function $\rightarrow$

$$\pi^*(x_t) = \arg\min_{u_t} Q^*(x_t, u_t) \quad \checkmark$$

$$= \arg\min_{u_t} J_t(x_t, u_t) + V_t^*(x_{t+1}) \quad \times$$

functional form of the cost

$$V_t^*(x_t) = \min_{u_t} J_t(x_t, u_t) + V_{t+1}^*(x_{t+1})$$

$$Q_t^*(x_t, u_t) = J_t(x_t, u_t) + V_{t+1}^*(x_{t+1})$$

$$Q_t^*(x_t, u_t) = J_t(x_t, u_t) + \min_{u_{t+1}} Q(x_{t+1}, u_{t+1})$$

need not be functional

$$J_t(x_t, u_t) = \|x_t - x_g\|_2^2$$

Q-learning is about learning the Q-function directly from data

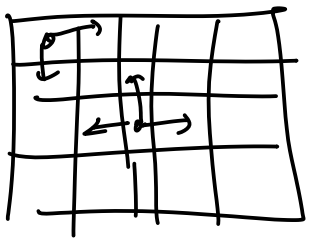
$$D \begin{cases} x_1, u_1 \rightarrow x_2, j_1 \\ x_2, u_2 \rightarrow x_3, j_2 \\ x_3, u_3 \rightarrow x_4, j_3 \end{cases}$$

$j_t \sim J_t(x_t, u_t)$ cost function from data

$j_1 \sim J_1(x_1, u_1)$

$j_2 \sim J_2(x_2, u$

How to represent Q function



① Tabular Q function

$$Q\begin{bmatrix} x_t & u_t \\ 0,0, & 0 \\ 0,0, & 1 \\ 0,0, & 2 \\ & 3 \end{bmatrix} \longrightarrow Q^*(x_t, u_t)$$

② Q-function as a NN

$$Q^*(x_t, u_t; W) = w_2^T \sigma\left(W_1 \begin{bmatrix} x_t \\ u_t \end{bmatrix}\right)$$

optimal action $\rightarrow \pi^*(x_t) = \min_{u_t} Q(x_t, u_t)$

Q-learning algorithm

① Exploration (Random actions)

Initialize $Q(x_t, u_t) = 0.2$ | optimistic encourages exploration

for tabular Use bellman equations to update

② Exploitation / learn Q function

$t \rightarrow$ 1 . . 100

99% $\rightarrow$ 1% of actions of to be random

ϵ-greedy policy

Neural Network $Q(x_t, u_t, W) = 0.2$

Replay buffer + shuffle

Loss function

gradient on this

$$L(W) = \left\| Q_t^*(x_t, u_t; W) - \left(r_t + \min_{u_{t+1}} Q(x_{t+1}, u_{t+1}; W) \right) \right\|_2^2$$

assume fixed

×

freeze

target network

$$W_{i+1} = W_i - \alpha \nabla_W L(W)$$

gradient descent