LQR, iLQR, MPC

Reinforcement Learning

Roberto Capobianco

SAPIENZA
UNIVERSITÀ DI ROMA

Recap

Policy Iteration

- Outputs policies at every iteration: $\{\pi_0, \pi_1, \pi_2 ... \pi_T\}$
- Different from Value Iteration that was outputting values

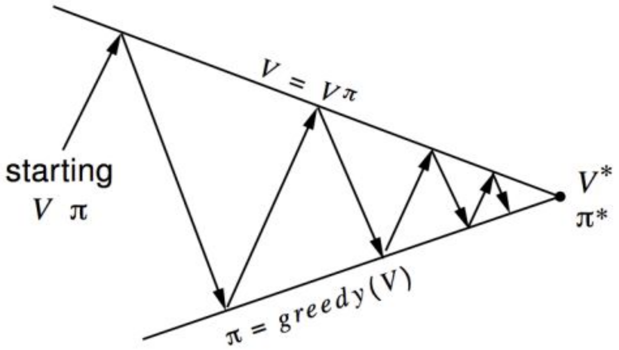
Procedure:

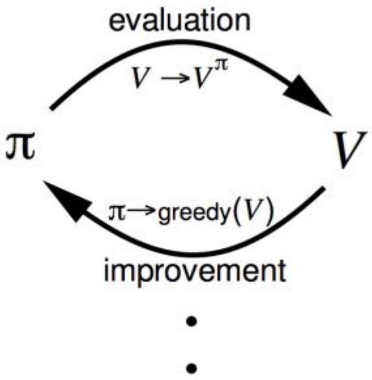
1. Start with a random guess π_0 (can be deterministic or stochastic)
2. For t=0,...,T:
   a. Do **policy evaluation** and compute $Q^{\pi t}$ for all s,a
   b. Do **policy improvement** as $\pi_{t+1}=\text{argmax}_a Q^{\pi t}(s,a)$ for all s

This algorithm only makes progress, and the performance progress of the policy is monotonic

Properties of Policy Iteration

- Monotonic improvement: $Q^{\pi_{t+1}} \geq Q^{\pi_t}$ for all s,a
- Convergence: $\|V^{\pi_i} - V^*\| \leq \gamma^{i+1}\|V^{\pi_0} - V^*\|$





Credits: David Silver

SAPIENZA
UNIVERSITÀ DI ROMA

Properties of Policy Iteration

- Monotonic improvement: $Q^{\pi_{t+1}} \geq Q^{\pi_t}$ for all s,a
- Convergence: $\|V^{\pi_i} - V^*\| \leq \gamma^{i+1}\|V^{\pi_0} - V^*\|$

Is there a max number of iterations of policy iteration?

$|A|^{|S|}$ since that is the maximum number of policies, and as the policy improvement step is monotonically improving, each policy can only appear in one round of policy iteration unless it is an optimal policy

Properties of Policy Iteration

- Monotonic improvement: $Q^{\pi t+1} \geq Q^{\pi t}$ for all s,a
- Convergence: $||V^{\pi i} - V^*|| \leq \gamma^{i+1}||V^{\pi 0} - V^*||$

When do we stop?

if the policy does not change anymore for any state

We Did Dynamic Programming!

Dynamic Programming can be applied if we have:

- *Optimal substructure:* Optimality exists and the optimal solution can be decomposed into subproblems
- *Overlapping subproblems:* Subproblems recur many times and the solutions can be cached and reused

MDPs satisfy both properties: thanks Bellman equation!

We Did Dynamic Programming!

We applied dynamic programming for **planning** as we assumed to know the MDP transition probabilities

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

Credits: David Silver

Primal Linear Program

As an alternative to VI and PI

Consider the Bellman optimality equation

$$V(s) = \max_a\{r_t+\gamma\mathbb{E}_{s'\sim p(.|s,\pi(s))}[V(s')]\}$$

and write it as a linear program:

$$\min V(s)$$

$$\text{such that } V(s)\geq r_t+\gamma\mathbb{E}_{s'\sim p(.|s,\pi(s))}[V(s')] \text{ for all } s,a$$

End Recap

Finite-Horizon MDPs

Slightly different formulation:

$$(S,\ A,\ R,\ T,\ \mathbf{H},\ \boldsymbol{\mu_0})$$

(time-horizon) $\mathbf{H \geq 0}$ **and** $\mathbf{s_0 \sim \mu_0}$ **(initial state distribution)**

Finite-Horizon MDPs

Slightly different formulation:

$$(S, A, R, T, \mathbf{H}, \boldsymbol{\mu_0})$$

(time-horizon) $\mathbf{H \geq 0}$ **and** $\mathbf{s_0 \sim \mu_0}$ **(initial state distribution)**

We consider time-dependent policies $\boldsymbol{\pi}$

$$\boldsymbol{\pi} = \{\pi_0, \pi_1, \pi_2 ... \pi_{H-1}\}$$

Finite-Horizon MDPs

Slightly different formulation:

$$(S, A, R, T, \mathbf{H}, \boldsymbol{\mu}_0)$$

(time-horizon) $\mathbf{H \geq 0}$ and $\mathbf{s_0 \sim \mu_0}$ (initial state distribution)

We consider time-dependent policies $\boldsymbol{\pi}$

$$\boldsymbol{\pi} = \{\pi_0, \pi_1, \pi_2 ... \pi_{H-1}\}$$

Actions might be different for the same state depending on t

Finite-Horizon MDPs

Slightly different formulation:

$$(S, A, R, T, \mathbf{H}, \boldsymbol{\mu_0})$$

(time-horizon) $H \geq 0$ and $s_0 \sim \mu_0$ (initial state distribution)

We consider time-dependent policies π

$$\boldsymbol{\pi} = \{\pi_0, \pi_1, \pi_2 \ldots \pi_{H-1}\}$$

e.g., I could explore more at the beginning

Finite-Horizon MDPs

Slightly different formulation:

$$(S,\ A,\ R,\ T,\ \mathbf{H},\ \boldsymbol{\mu_0})$$

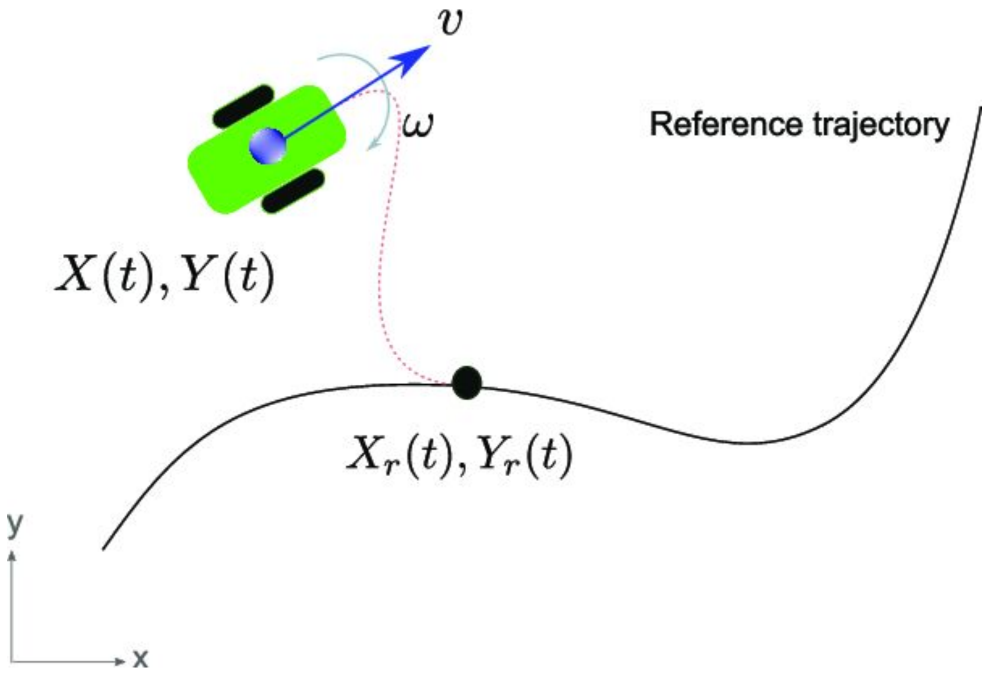
(time-horizon) $\mathbf{H \geq 0}$ and $\mathbf{s_0 \sim \mu_0}$ (initial state distribution)

We consider time-dependent policies π

$$\boldsymbol{\pi} = \{\pi_0,\ \pi_1,\ \pi_2 \ldots \pi_{H-1}\}$$

Very common in control!

Finite-Horizon MDPs - Example



Finite-Horizon MDP & Policy Interaction

$$\text{MDP} = (S, A, R, T, H, \mu_0)$$

$$\pi = \{\pi_0, \pi_1, \pi_2 ... \pi_{H-1}\}$$

$$s_0 \sim \mu_0$$

$$(s_0)$$

Finite-Horizon MDP & Policy Interaction

$$MDP = (S, A, R, T, H, \mu_0)$$

$$\pi = \{\pi_0, \pi_1, \pi_2 ... \pi_{H-1}\}$$

$$s_0 \sim \mu_0$$

$$a_0 = \pi_0(s_0)$$

$$(s_0, a_0)$$

Finite-Horizon MDP & Policy Interaction

$$MDP = (S, A, R, T, H, \mu_0)$$

$$\pi = \{\pi_0, \pi_1, \pi_2 ... \pi_{H-1}\}$$

$$s_1 \sim P(.|s_0, a_0)$$

$$(s_0, a_0, s_1)$$

Finite-Horizon MDP & Policy Interaction

$$MDP = (S, A, R, T, H, \mu_0)$$

$$\pi = \{\pi_0, \pi_1, \pi_2 ... \pi_{H-1}\}$$

$$s_1 \sim P(.|s_0, a_0)$$

$$a_1 = \pi_1(s_1)$$

$$(s_0, a_0, s_1, a_1)$$

Finite-Horizon MDP & Policy Interaction

$$\text{MDP} = (S, A, R, T, H, \mu_0)$$

$$\pi = \{\pi_0, \pi_1, \pi_2 \ldots \pi_{H-1}\}$$

$$(\mathbf{s_0}, \mathbf{a_0}, \mathbf{s_1}, \mathbf{a_1}, \ldots \mathbf{s_{H-1}}, \mathbf{a_{H-1}})$$

Finite-Horizon MDP: V & Q

$$V_h^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=h}^{H-1} r(s_t, a_t)]$$

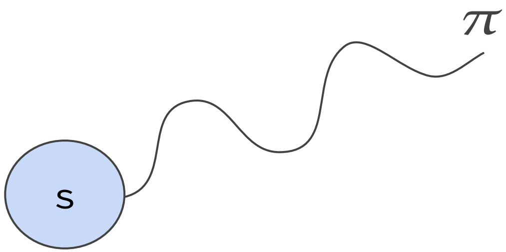
where $s_h = s$, $a_t = \pi_t(s_t)$ and $s_{t+1} \sim P(.|s_t, a_t)$

No discount factor!

Finite-Horizon MDP: V & Q

$$V_h^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=h}^{H-1} r(s_t, a_t)\right]$$

where $s_h = s$, $a_t = \pi_t(s_t)$ and $s_{t+1} \sim P(.|s_t, a_t)$



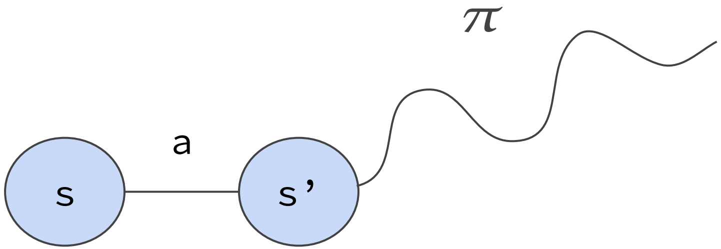
Finite-Horizon MDP: V & Q

$$V_h^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=h}^{H-1} r(s_t, a_t)]$$

where $s_h = s$, $a_t = \pi_t(s_t)$ and $s_{t+1} \sim P(.|s_t, a_t)$

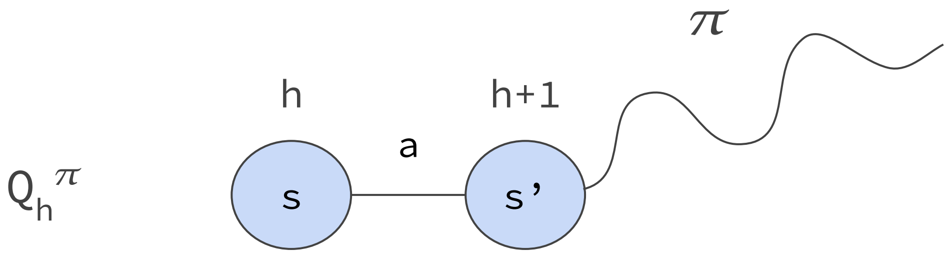
$$Q_h^{\pi}(s,a) = \mathbb{E}_{\pi}[\sum_{t=h}^{H-1} r(s_t, a_t)]$$

where $s_h = s$, $a_h = a$, $a_t = \pi_t(s_t)$ and $s_{t+1} \sim P(.|s_t, a_t)$



Finite-Horizon MDP: Bellman Equation

$$Q_h^{\pi}(s,a) = r(s,a) + \mathbb{E}_{s' \sim p(.|s,a)}[V_{h+1}^{\pi}(s')]$$



Finding the Optimal Policy

$$\pi^* = \{\pi_0^*, \pi_1^*, \pi_2^* ... \pi_{H-1}^*\}$$

Easier problem than infinite horizon!

Finding the Optimal Policy

$$\pi^* = \{\pi_0^*, \pi_1^*, \pi_2^* \ldots \pi_{H-1}^*\}$$

Let’s reason backwards in time and apply dynamic programming:

$Q_{H-1}^*(s,a)$?

Finding the Optimal Policy

$$\pi^* = \{\pi_0^*, \pi_1^*, \pi_2^* \ldots \pi_{H-1}^*\}$$

Let’s reason backwards in time and apply dynamic programming:

$$Q_{H-1}^*(s,a) = r(s,a)$$

Finding the Optimal Policy

$$\pi^* = \{\pi_0^*, \pi_1^*, \pi_2^* \ldots \pi_{H-1}^*\}$$

Let’s reason backwards in time and apply dynamic programming:

$$Q_{H-1}^*(s,a) = r(s,a)$$

$$\pi_{H-1}^*(s)=\mathrm{argmax}_a Q_{H-1}^*(s,a)$$

$$V_{H-1}^*(s)=\max_a Q_{H-1}^*(s,a)=Q_{H-1}^*(s,\pi_{H-1}^*(s))$$

Finding the Optimal Policy

$$Q_{H-1}^*(s,a) = r(s,a)$$

$$\pi_{H-1}^*(s)=\mathrm{argmax}_a Q_{H-1}^*(s,a)$$

$$V_{H-1}^*(s)=\max_a Q_{H-1}^*(s,a)=Q_{H-1}^*(s,\pi_{H-1}^*(s))$$

Now we can reason about H-2!

Finding the Optimal Policy

$$Q_{H-1}^*(s,a) = r(s,a)$$

$$\pi_{H-1}^*(s)=\text{argmax}_a Q_{H-1}^*(s,a)$$

$$V_{H-1}^*(s)=\max_a Q_{H-1}^*(s,a)=Q_{H-1}^*(s,\pi_{H-1}^*(s))$$

Now we can reason about H-2!

Bellman Equation: $Q_h^\pi(s,a) = r(s,a)+\mathbb{E}_{s'\sim p(.|s,a)}[V_{h+1}^\pi(s')]$

SAPIENZA UNIVERSITÀ DI ROMA

Finding the Optimal Policy

$$Q_{H-1}^*(s,a) = r(s,a)$$

$$\pi_{H-1}^*(s)=\text{argmax}_a Q_{H-1}^*(s,a)$$

$$V_{H-1}^*(s)=\max_a Q_{H-1}^*(s,a)=Q_{H-1}^*(s,\pi_{H-1}^*(s))$$

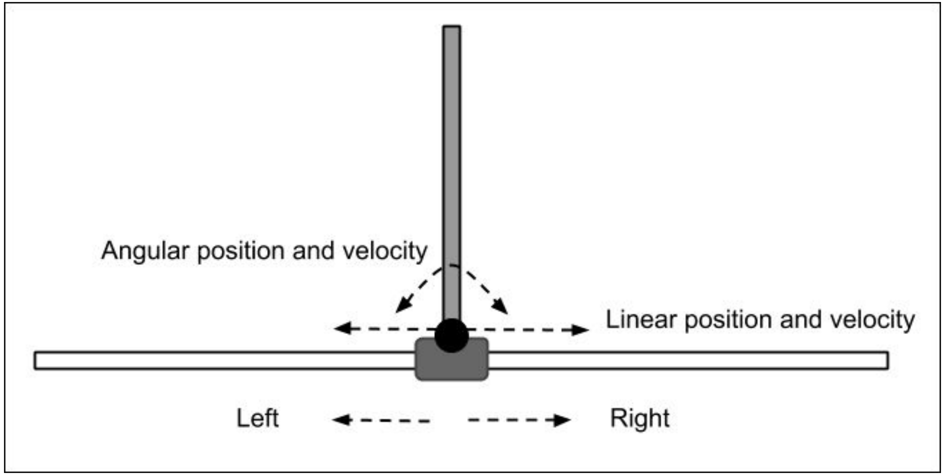
Now we can reason about H-2!

Bellman Equation: $Q_h^\pi(s,a) = r(s,a)+\mathbb{E}_{s'\sim p(.|s,a)}[V_{h+1}^\pi(s')]$

$$\pi_h^*(s)=\text{argmax}_a Q_h^*(s,a)$$

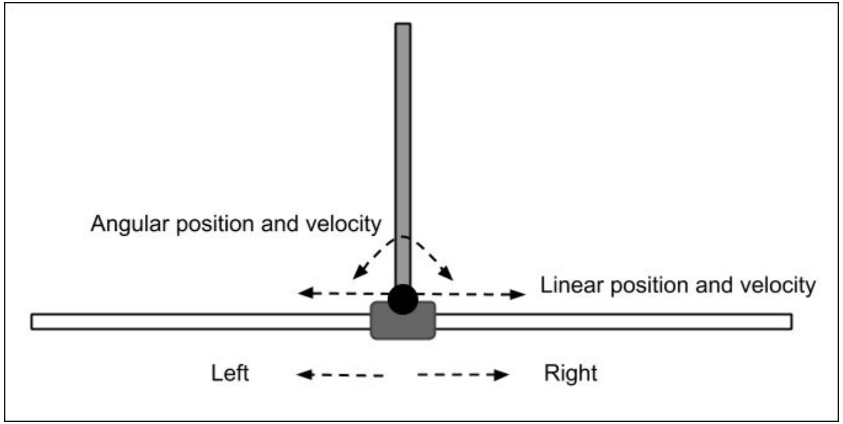
Control Problems

So far, we assumed discrete state and action spaces, but what about cartpole:



Control Problems

So far, we assumed discrete state and action spaces, but what about cartpole:



- **state:** angular pos & vel, linear pos & vel
- **action/control:** force applied on the cart
- **goal:** find the control policy which minimizes the long term cost c

Control Problems

More in general in control problems we have x in $\mathbb{R}^d$ and u in $\mathbb{R}^k$

We denote the state as x and the action as u, as it's typical for control problems

Control Problems

More in general in control problems we have x in $\mathbb{R}^d$ and u in $\mathbb{R}^k$

We also talk about cost instead of reward: we want to minimize the cost instead of maximizing the reward!

Optimal Control

Given a dynamical system with a non-linear transition function f, state x in $\mathbb{R}^d$ and control u in $\mathbb{R}^k$, we want to find a control policy π such that

$$\text{minimize } \mathbb{E}_\pi[c_H(x_H) + \sum_{h=0}^{H-1} c_h(x_h, u_h)]$$

$$\text{where } u_h = \pi(x_h) \text{ and } x_0 \sim \mu_0$$

Optimal Control

Given a dynamical system with a non-linear transition function f, state x in $\mathbb{R}^d$ and control u in $\mathbb{R}^k$, we want to find a control policy π such that

$$\text{minimize } \mathbb{E}_{\pi}[c_H(x_H) + \sum_{h=0}^{H-1} c_h(x_h,u_h)]$$

$$\text{where } u_h=\pi(x_h) \text{ and } x_0 \sim \mu_0$$

Now this seems very familiar! Can we treat it as a Finite-Horizon MDP and use value iteration?

Sapienza Università di Roma

Optimal Control

Given a dynamical system with a non-linear transition function f, state x in $\mathbb{R}^d$ and control u in $\mathbb{R}^k$, we want to find a control policy π such that

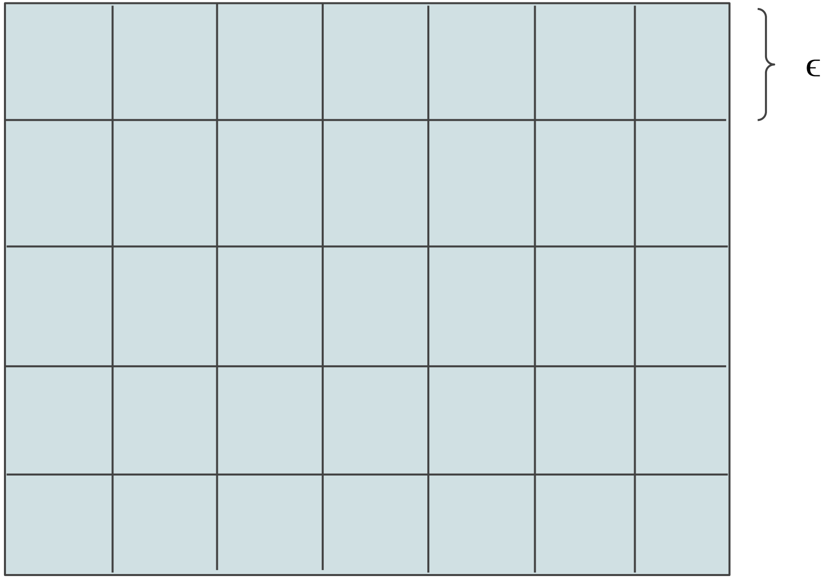
$$\text{minimize } \mathbb{E}_{\pi}[c_H(x_H) + \textstyle\sum_{h=0}^{H-1} c_h(x_h, u_h)]$$

$$\text{where } u_h = \pi(x_h) \text{ and } x_0 \sim \mu_0$$

Now this seems very familiar! Can we treat it as a Finite-Horizon MDP and use value iteration? **YES if we can discretize**

SAPIENZA UNIVERSITÀ DI ROMA

Discretization



X

x in $\mathbb{R}^d$ and u in $\mathbb{R}^k$

Number of total points on the discretized grid increases exponentially and becomes

$$(1/\epsilon)^d + (1/\epsilon)^k$$

Bellman's Curse of Dimensionality

- n-dimensional (discrete) state space
- The number of states grows exponentially in n

In practice discretization is useful, but it is only computationally feasible up to 5 or 6 dimensional state spaces

Bellman's Curse of Dimensionality

- n-dimensional (discrete) state space
- The number of states grows exponentially in n

In practice discretization is useful, but it is only computationally feasible up to 5 or 6 dimensional state spaces

Let's try to work directly in continuous space, starting from simplified problems

Linear Systems

Consider a system of this kind:

$$x_{t+1}=Ax_t+Bu_t$$

- x_t state at time t
- u_t control (i.e., action) at time t

A in $\mathbb{R}^{dxd}$, B in $\mathbb{R}^{dxk}$

Linear Systems

Consider a system of this kind:

$$\boxed{x_{t+1}=Ax_t+Bu_t}$$

This is our transition function!

- x_t state at time t
- u_t control (i.e., action) at time t

A in $\mathbb{R}^{dxd}$, B in $\mathbb{R}^{dxk}$

Quadratic Cost Function

Consider a cost function of this kind

$$c(x_t,u_t)=x_t^TQx_t+u_t^TRu_t$$

alternative notation
$g(x_t,u_t)$

- Q in $\mathbb{R}^{dxd}$ and R in $\mathbb{R}^{kxk}$ square matrices
- Q and R positive definite

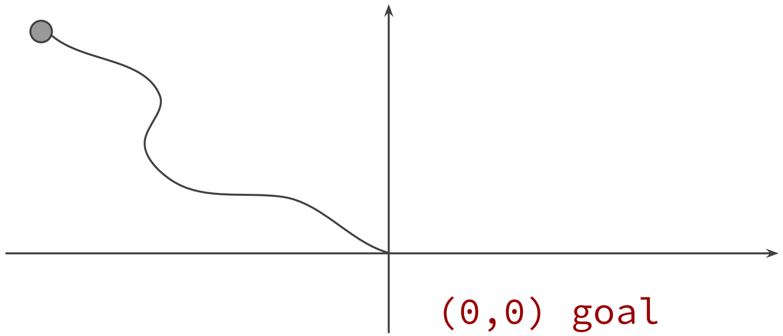
As a result, there is a non-zero cost for any non-zero state with all-zero control

Quadratic Cost Function

Consider a cost function of this kind

$$c(x_t,u_t)=x_t^TQx_t+u_t^TRu_t$$

As a result, there is a non-zero cost for any non-zero state with all non-zero control



Q Functions for Linear Systems and Quadratic Cost

$$Q_h^*(x,u) = c(x,u)+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')]$$

Q Functions for Linear Systems and Quadratic Cost

$$Q_h^*(x,u) = c(x,u) + \mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')]$$

$$c(x_t,u_t) = x_t^T Q x_t + u_t^T R u_t$$

Q Functions for Linear Systems and Quadratic Cost

$$Q_h^*(x,u) = c(x,u) + \mathbb{E}_{x' \sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^T Q x_t + u_t^T R u_t + \mathbb{E}_{x' \sim p(.|x,u)}[V_{h+1}^*(x')]$$

Q Functions for Linear Systems and Quadratic Cost

$$Q_h^*(x,u) = c(x,u)+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^TQx_t+u_t^TRu_t+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')]$$

$$x_{t+1}=Ax_t+Bu_t$$

(deterministic for now)

Q Functions for Linear Systems and Quadratic Cost

$$Q_h^*(x,u) = c(x,u)+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^T Q x_t + u_t^T R u_t + \mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^T Q x_t + u_t^T R u_t + V_{h+1}^*(Ax_t + Bu_t)$$

Inductive Step: Assumption

$$Q_h^*(x,u) = c(x,u)+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^TQx_t+u_t^TRu_t+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^TQx_t+u_t^TRu_t+V_{h+1}^*(Ax_t+Bu_t)$$

Assume $V_{h+1}^*(x_t)=x_t^TP_{h+1}x_t$ with P in $\mathbb{R}^{dxd}$

Inductive Step: Question

$$Q_h^*(x,u) = c(x,u)+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^TQx_t+u_t^TRu_t+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^TQx_t+u_t^TRu_t+V_{h+1}^*(Ax_t+Bu_t)$$

Can we show that also $V_h^*(x_t)=x_t^TP_hx_t$ with P in $\mathbb{R}^{dxd}$?

Inductive Step: Implement the Assumption

$$Q_h^*(x,u) = c(x,u)+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^TQx_t+u_t^TRu_t+\mathbb{E}_{x'\sim p(.|x,u)}[V_{h+1}^*(x')] =$$

$$x_t^TQx_t+u_t^TRu_t+V_{h+1}^*(Ax_t+Bu_t) =$$

$$\mathbf{x_t^TQx_t+u_t^TRu_t+(Ax_t+Bu_t)^TP_{h+1}(Ax_t+Bu_t)}$$

Finding the Best Action

We are now interested in finding the $\pi_h^*(x)=\text{argmin}_u Q_h^*(x,u)$

$$Q_h^*(x_t,u_t) = x_t^T Q x_t + u_t^T R u_t + (Ax_t+Bu_t)^T P_{h+1}(Ax_t+Bu_t)$$

Finding the Best Action

We are now interested in finding the $\pi_h^*(x)=\text{argmin}_u Q_h^*(x,u)$

$$Q_h^*(x_t,u_t) = x_t^TQx_t+u_t^TRu_t+(Ax_t+Bu_t)^TP_{h+1}(Ax_t+Bu_t)$$

How do you find minima?

Finding the Best Action

We are now interested in finding the $\pi_h^*(x)=\text{argmin}_u Q_h^*(x,u)$

$$Q_h^*(x_t,u_t) = x_t^T Q x_t + u_t^T R u_t + (Ax_t+Bu_t)^T P_{h+1}(Ax_t+Bu_t)$$

Set $\nabla_u Q_h^*(x,u)=0$ and solve for u

Inductive Step: Proof

Assume $V_{h+1}{}^* = x_t{}^T P_{h+1} x_t{}^T$, can we show that also $V_h{}^*(x_t) = x_t{}^T P_h x_t$

Inductive Step: Proof

Assume $V_{h+1}^* = x_t^T P_{h+1} x_t$, can we show that also $V_h^*(x_t) = x_t^T P_h x_t$

1. Set $Q_h^*(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t + (A x_t + B u_t)^T P_{h+1} (A x_t + B u_t)$

Inductive Step: Proof

Assume $V_{h+1}^{*} = x_t^T P_{h+1} x_t$, can we show that also $V_h^{*}(x_t) = x_t^T P_h x_t$

1. Set $Q_h^{*}(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t + (Ax_t + Bu_t)^T P_{h+1} (Ax_t + Bu_t)$
2. $\nabla_u Q_h^{*}(x_t, u_t) = 2Ru_t + 2B^T P_{h+1} (Ax_t + Bu_t) = 0$

Inductive Step: Proof

Assume $V_{h+1}^* = x_t^T P_{h+1} x_t^T$, can we show that also $V_h^*(x_t) = x_t^T P_h x_t$

1. Set $Q_h^*(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t + (Ax_t + Bu_t)^T P_{h+1} (Ax_t + Bu_t)$
2. $\nabla_u Q_h^*(x_t, u_t) = 2Ru_t + 2B^T P_{h+1} (Ax_t + Bu_t) = 0$
3. Solve for $u = \pi_h^*(x_t) = -(R + B^T P_H B)^{-1} B^T P_{h+1} A x_t = K_h^* x_t$

Inductive Step: Proof

Assume $V_{h+1}^* = x_t^T P_{h+1} x_t^T$, can we show that also $V_h^*(x_t)=x_t^T P_h x_t$

- $u = \pi_h^*(x_t) = -(R+B^T P_H B)^{-1} B^T P_{h+1} A x_t = K_h^* x_t$
- We know that $V_h^*(x)=\max_u Q_h^*(x,u)=Q_h^*(x,\pi_h^*(x)) = Q_h^*(x,K_h^* x)$

We can replace it in

$Q_h^* = x_t^T Q x_t + x_t^T K_h^{*T} R K_h^* x_t + (Ax_t + BK_h^* x_t)^T P_{h+1} (Ax_t + BK_h^* x_t)$

Inductive Step: Proof

Assume $V_{h+1}^* = x_t^T P_{h+1} x_t^T$, can we show that also $V_h^*(x_t)=x_t^T P_h x_t$

- $u = \pi_h^*(x_t) = -(R+B^T P_H B)^{-1} B^T P_{h+1} A x_t = K_h^* x_t$
- We know that $V_h^*(x)=\max_u Q_h^*(x,u)=Q_h^*(x,\pi_h^*(x)) = Q_h^*(x,K_h^* x)$

We can replace it in

$$Q_h^* = x_t^T \mathbf{Q} x_t + x_t^T \mathbf{K_h^{*T} R K_h^*} x_t + (\mathbf{A} x_t + \mathbf{B K_h^*} x_t)^T \mathbf{P_{h+1}} (\mathbf{A} x_t + \mathbf{B K_h^*} x_t) =$$

$$x_t^T \mathbf{(Q + K_h^{*T} R K_h^* + (A + B K_h^*)^T P_{h+1} (A + B K_h^*))} x_t = x_t^T \mathbf{P_h} x_t$$

Cost at H

Recall our goal is to minimize $\mathbb{E}_\pi[c_H(x_H) + \sum_{h=0}^{H-1} c_h(x_h,u_h)]$

So, at H, we only have $c_H(x_H)$. This is typically 0 or $x_H^TQx_H$

Cost at H

Recall our goal is to minimize $\mathbb{E}_{\pi}[c_H(x_H) + \sum_{h=0}^{H-1} c_h(x_h,u_h)]$

So, at H, we only have $c_H(x_H)$. This is typically 0 or $x_H^T Q x_H$

This is our base-case and we set P_H to 0 or Q or Q_{final}

LQR algorithm

- Initialize P_H (at 0 or Q)
- Starting from h = H−1, backwards
  - Set $K_h^* = -(R+B^T P_H B)^{-1} B^T P_{h+1} A$
  - Compute $u = \pi_h^*(x_t) = K_h^* x_t$
  - Set $P_h = (Q + K_h^{*T} R K_h^* + (A + BK_h^*)^T P_{h+1} (A + BK_h^*))$
  - Set $V_h^* = x_t^T P_h x_t$

LQR algorithm

- Initialize P_H (at 0 or Q)
- Starting from h = H-1, backwards
  - Set $K_h^* = -(R+B^TP_HB)^{-1}B^TP_{h+1}A$
  - Compute $u = \pi_h^*(x_t) = K_h^*x_t$
  - Set $P_h = (Q + K_h^{*T}RK_h^* + (A + BK_h^*)^TP_{h+1}(A + BK_h^*))$
  - Set $V_h^* = x_t^TP_hx_t$

This is the Value Iteration update done in closed form: it is always the same and solves this particular continuous-state system with a quadratic cost

LQR algorithm

- Initialize P_H (at 0 or Q)
- Starting from h = H-1, backwards
  - Set $K_h^* = -(R+B^TP_HB)^{-1}B^TP_{h+1}A$
  - Compute $u = \pi_h^*(x_t) = K_h^*x_t$
  - Set $\mathbf{P_h = (Q + K_h^{*T}RK_h^* + (A + BK_h^*)^TP_{h+1}(A + BK_h^*))}$
  - Set $V_h^* = x_t^TP_hx_t$

Riccati Equation

LQR algorithm

- Initialize P_H (at 0 or Q)
- Starting from h = H-1, backwards
  - Set $K_h^* = -(R+B^TP_HB)^{-1}B^TP_{h+1}A$
  - Compute $u = \pi_h^*(x_t) = K_h^*x_t$
  - Set $P_h = (Q + K_h^{*T}RK_h^* + (A + BK_h^*)^TP_{h+1}(A + BK_h^*))$
  - Set $\mathbf{J_h}^* = x_t^TP_hx_t$

Since V is the value, we will rename it to J, for this kind of problems, as the "cost-to-go"

LQR Extensions

Extensions to the LQR make it more generally applicable to:

- Affine systems
- Systems with stochasticity
- Regulation around non-zero fixed point for non-linear systems
- Trajectory following for non-linear systems
- ...

LQR for Affine Systems

Affine system: $Ax_t + Bu_t$ **+ c**

Cost: $c(x_t,u_t)=x_t^TQx_t+u_t^TRu_t$

Solve it **in the same way** by simply re-defining the state and transition function as

$$z_t = [x_t;\ 1]$$

$$z_{t+1} = \begin{bmatrix} x_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A & c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t = A' z_t + B' u_t$$

LQR for Stochastic Systems

Stochastic system: $Ax_t + Bu_t$ **+ w_t with w_t zero-mean Gaussian noise**

$$\mathbf{w_t \sim \mathcal{N}(0,\sigma^2 I)}$$

Cost: $c(x_t,u_t)=x_t^TQx_t+u_t^TRu_t$

The optimal control policy is the same; cost-to-go has an extra term depending on the variance of the syste and that cannot be controlled: $J_h^*= x_t^TP_hx_t + p_h$

$$p_h = tr(\sigma^2 P_{h+1}) + p_{h+1}$$

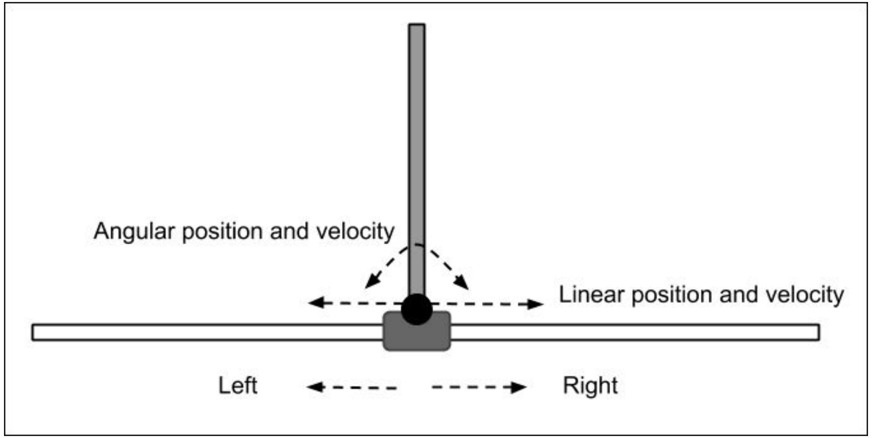
$$\text{base-case } p_H = 0$$

SAPIENZA
UNIVERSITÀ DI ROMA

LQR for Non-Linear Systems

Non-linear system: $\mathbf{f(x_t, u_t)}$

Cost: $c(x_t,u_t)=x_t^TQx_t+u_t^TRu_t$



We have a goal: x^*, u^*

We want to stabilize the system around it, such that:

$$x^* = f(x^*, u^*)$$

LQR for Non-Linear Systems

Non-linear system: $\mathbf{f(x_t, u_t)}$

Cost: $c(x_t,u_t)=x_t^TQx_t+u_t^TRu_t$

We can linearize the dynamics around $f(x^*, u^*)$ using Taylor expansion:

$$x_{t+1} \approx f(x^*,u^*) + \nabla_x f(x^*,u^*)(x_t-x^*) + \nabla_u f(x^*,u^*)(x_t-x^*)$$

LQR for Non-Linear Systems

Non-linear system: $\mathbf{f(x_t, u_t)}$

Cost: $c(x_t,u_t)=x_t^TQx_t+u_t^TRu_t$

We can linearize the dynamics around $f(x^*, u^*)$ using Taylor expansion:

$$x_{t+1} \approx f(x^*,u^*) + \nabla_x \mathbf{f(x^*,u^*)}(x_t-x^*) + \nabla_u \mathbf{f(x^*,u^*)}(x_t-x^*)$$

$$x_{t+1} - f(x^*,u^*) \approx + \mathbf{A}(x_t-x^*) + \mathbf{B}(x_t-x^*)$$

LQR for Non-Linear Systems

Non-linear system: $\mathbf{f(x_t, u_t)}$

Cost: $g(x_t,u_t)$

We can linearize the dynamics around $f(x^*, u^*)$ using Taylor expansion:

$$x_{t+1} \approx \mathbf{f(x^*,u^*)} + \nabla_{\mathbf{x}}\mathbf{f(x^*,u^*)}(x_t-x^*) + \nabla_{\mathbf{u}}\mathbf{f(x^*,u^*)}(x_t-x^*)$$

$$x_{t+1} - x^* \approx + \mathbf{A}(x_t-x^*) + \mathbf{B}(x_t-x^*)$$

Let $z_t = x_t - x^*$ and $v_t = u_t - u^*$

LQR for Non-Linear Systems

Non-linear system: $\mathbf{f(x_t, u_t)}$

Assume cost is: $c(x_t,u_t)=x_t^TQx_t+u_t^TRu_t$

$$\text{Let } z_t = x_t - x^* \text{ and } v_t = u_t - u^*$$

$$z_{t+1} = Az_t+Bv_t$$

Solve it the usual way: $v_t = Kz_t \rightarrow u_t = u^* + K(x_t - x^*)$

LQR for Non-Linear Systems

Non-linear system: $\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$

Cost: $g(x_t, u_t)$

We can also linearize the cost using Taylor expansion (do it at home), but it becomes something like:

$x_t^T \mathbf{Q} x_t + u_t^T \mathbf{R} u_t + u_t^T \mathbf{M} x_t + x_t^T \mathbf{q} + u_t^T \mathbf{r} + c$

LQR for Trajectory Following

Non-linear system: **f(x, u)**

Trajectory: $x_0^*, \ldots, x_T^*$

We want controls such that $x_{t+1}^* = f(x_t^*, u_t^*)$ for all the trajectory states

LQR for Trajectory Following

Non-linear system: **f(x, u)**

Trajectory: x_0^*, ..., x_T^*

We want controls such that $x_{t+1}^* = f(x_t^*, u_t^*)$ for all the trajectory states

In other words we are doing:

$$\min_{u_0, u_1, \ldots, u_{H-1}} \sum_{t=0}^{H-1} (x_t - x_t^*)^\top Q (x_t - x_t^*) + (u_t - u_t^*)^\top R (u_t - u_t^*)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

LQR for Trajectory Following

Non-linear system: **f(x, u)**

Trajectory: x_0^*, ..., x_T^*

We want controls such that $x_{t+1}^*=f(x_t^*,u_t^*)$ for all the trajectory states

We use the same trick as before but with x_{t+1}^* and x_{t+1}^*

$$x_{t+1} - x_{t+1}^* \approx + \mathbf{A}(x_t - x_t^*) + \mathbf{B}(x_t - x_t^*)$$