Introduction to Reinforcement Learning

recorded by Teng Zeng

July 26, 2019

Optimal Control Formulation

$$\min J = \sum_{k=0}^{N-1} c_k\{x_k, u_k\} + c_N\{x_N\} \tag{1}$$

$$\text{s.t.} \;\; x_{k+1} = f(x_k, u_k); \;\; k = 0, \ldots, N-1 \tag{2}$$

$$x_k \in \mathcal{X}_k, u_k \in \mathcal{U}_k \tag{3}$$

Eqn.3 are feasible or "admissible" sets, Exs:

$$\underline{x} \le x_k \le \bar{x}, \;\; \underline{u} \le u_k \le \bar{u}.$$

Remember our objective: find a control law (a.k.a "policy" of the form $u_k = \pi(x_k)$taht solves Eqns.1-3. Takes the form of a "state feedback" control law/policy. Ex:

- $u_k = -K_L x_L, \;\; K_L \in \mathbf{R}^{p\times n}$(Linear state feedback)
- $u_k = f_{NL}(x_k), \;\; f_{NL} : \mathbf{R}^n \longrightarrow \mathbf{R}^p$(Nonlinear)

Dynamic Programming

Finite Time Horizon Case

Principal of Optimality: Assume at time step k you know the optimal controls from k to N. $u_{k+1}, \ldots, u_{N-1}$ Then the optimal sequence of controls is the best control u_k at time step k paired with the future optimal control actions.

Make precise using an object control to many RL algorithms, called the value function: $V(x) : \mathbf{R}^n \longrightarrow \mathbf{R}$.

Define $v_k(x_k)$ as the optimal "value" from time step k to N, given current state x_k. Bellman's Principle of Optimality Eqn:

$$v_k(x_k) = \min_u \{\underbrace{c_k(x_k, u)}_{\text{cost now!}} + \underbrace{V_{k+1}(x_{k+1})\}}_{\text{minimum cost function } k+1 \text{ to} N}$$

Boundary/final condition:

$$V_N(x_N) = C_N(x_N).$$

Admittedly awkward:

- Recursive,
- Compute it backwards for all states.

$\times$ Focus on special case that is very common in practice: LQR.

$$\min_{x_k, u_k} J = \sum_{k=0}^{N-1} [x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k] + x_N^{\mathrm{T}} Q_f x_N$$
$$\text{s.t.} \;\; x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N = 1,$$
$$\text{where} \;\; Q = Q^{\mathrm{T}} > 0, R = R^{\mathrm{T}} \geq 0, Q_f = Q_f^{\mathrm{T}} \geq 0$$

Objective is convex quadratic and equality constraint is linear.
Step I: Let $V_k(z)$ represent the minimum LQR cost from time step k to N, given we are currently in state z. In math,

$$V_k(z) = \min_{u_k, \ldots, u_{N-1}} \sum_{\tau=k}^{N-1} [x_\tau^{\mathrm{T}} Q x_\tau + u_\tau^{\mathrm{T}} R u_\tau] + x_N^{\mathrm{T}} Q_f x_N,$$

where $x_k = z$, $x_{\tau+1} = x_\tau = Ax_\tau + Bu_\tau$. We will discover that $V_k(z) = z^{\mathrm{T}} P_k z$ where $P_k = P_k^{\mathrm{T}} \geq 0$ and can be computed recursively from P_{k+1}.
Step II: Bellman's Principle of Optimality

$$V_k(z) = \min_w z^{\mathrm{T}} Q z + w^{\mathrm{T}} R w + \underbrace{V_{k+1}(Az + Bw)}_{x_{k+1}}\}$$

$$V_k(z) = z^{\mathrm{T}} Q z + \min_w w^{\mathrm{T}} R w + \underbrace{V_{k+1}(Az + Bw)}_{x_{k+1}}\}$$

Recursive way to compute V_k given V_{k+1}.
Step III: Start recursion at $V_N(z) = z^{\mathrm{T}} Q_f z$. Any minimizer w gives LQR optimal u, $u_k^{\mathrm{lqr}} = \mathrm{argmin}_w \{w^{\mathrm{T}} R w + V_{k+1}(Az + Bw)\}$. Let's begin by assuming $V_{k+1}(z) = z^{\mathrm{T}} P_{k+1} z$ with $P_{k+1} = P_{k+1}^{\mathrm{T}} \geq 0$. Show $V_k(z)$ is quadratic.

$$\begin{aligned}
V_{k+1}(z) &= z^{\mathrm{T}} Q z + \min_w w^{\mathrm{T}} R w + (Az + Bw)^{\mathrm{T}} P_{k+1} (Az + Bw) \\
\frac{\partial}{\partial w} &\text{set to } 0 : 2Rw^* + 2B^{\mathrm{T}} P_{k+1}(Az + Bw^*) = 0 \\
w^* &= -(R + B^{\mathrm{T}} P_{k+1} B)^{-1} B^{\mathrm{T}} P_{k+1} A z \\
\text{Plug } \; w^* \; \text{ back to } \; V_k(z) &= z^{\mathrm{T}} (Q + A^{\mathrm{T}} P_{k+1} A - A^{\mathrm{T}} P_{k+1} B (R + B^{\mathrm{T}} P_{k+1} B)^{-1} B^{\mathrm{T}} P_{k+1}) z \\
V_k &= z^{\mathrm{T}} P_k z
\end{aligned}$$

Infinite Time Horizon Case + LQR

Consider discounted formulation:

$$\min_{x_k, u_k} \sum_{k=0}^{\infty} \gamma^k \cdot c(x_k, u_k),$$
$$\text{s.t.} \;\; x_{k+1} = f(x_k, u_k) \quad k = 0, 1, \ldots$$

Define $V(x_k)$ as the optimal value function from time step k onward to ∞ given the current state is x_k. Furthermore, denote by $V_\pi(x_k)$ the value function corresponding to policy π, which is not necessarily optimal.

Note:

$$
\begin{aligned}
V_\pi(x_k) &= \sum_{\tau=k}^{\infty} \gamma^{\tau-k} c(x_k, u_k) \\
&= c(x_k, u_k) + \gamma \underbrace{\sum_{\tau=k+1}^{\infty} \gamma^{\tau-(k+1)} c(x_k, u_k)}_{V_\pi(x_{k+1})} \\
&= c(x_k, u_k) + \gamma \cdot V_\pi(x_{k+1}) \\
\text{where} \ \ x_{k+1} &= f(x_k, u_k), u_k = \pi(x_k)
\end{aligned}
$$

Bellman's Optimality Equation:

$$V(x_k) = \min_{\pi(\cdot)} \{c(x_k, \pi(x_k)) + V(x_{k+1})\}$$

Remark: This equation is also known as the discrete time Hamilton-Jacobi-Bellman (HJB) equation. The optimal policy:

$$\pi^*(x_k) = \text{argmin}_{\pi(\cdot)} \{c(x_k, u_k) + \gamma V(x_{k+1})\}$$

Case study: Infinite time LQR

$$
\begin{aligned}
\min \sum_{k=0}^{\infty} [x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k], \quad \gamma = 1 \\
\text{s.t.} \ \ x_{k+1} = A x_k + B u_k, \quad k = 0, 1, \ldots
\end{aligned}
$$

Like before, we will discover,

$$V(x_k) = x_k^{\mathrm{T}} P x_k, u_k = K x_k.$$

Consider an arbitrary K and Bellman equation:

$$x_k^{\mathrm{T}} P x_k = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + (A x_k + B u_k)^{\mathrm{T}} P (A x_k + B u_k).$$

substitute $u_k = K x_k$

$$x_k^{\mathrm{T}} P x_k = x_k^{\mathrm{T}} [Q + K^{\mathrm{T}} R K + (A + BK)^{\mathrm{T}} P (A + BK)] x_k.$$

Then,

$$P^{j+1} = Q + K^{\mathrm{T}} R K + (A + BK)^{\mathrm{T}} P^j (A + BK),$$

linear in P, for some given K. $P^j \longrightarrow P^{\text{old}}$ as $j \longrightarrow \infty$.(???)

Improve control policy by using Bellman Optimality equation

$$x_k^{\mathrm{T}} P x_k = \min_w x_k^{\mathrm{T}} Q x_k + w_k^{\mathrm{T}} R w_k + (A x_k + B w_k)^{\mathrm{T}} P (A x_k + B w_k),$$

differentiate w.r.t w, set to zero, FONC:

$$
\begin{aligned}
2RW + 2B^{\mathrm{T}} P (A x_k + B w) = 0, \\
w^* = \underbrace{-(R + B^{\mathrm{T}} P B)^{-1} B^{\mathrm{T}} P^{\text{old}} A \cdot}_{K^{\text{new}}} x_k. \\
u_k^* = K x_k
\end{aligned}
$$

Substitute u_k^* back into Bellman Eqn and simplifying:

$$A^{\mathrm{T}} P A - P + Q - A^{\mathrm{T}} P B (R + B^{\mathrm{T}} P B)^{-1} B^{\mathrm{T}} P A = 0,$$

which is a quadratic matrix eqn in P called "Ricatti Eqn." For LQR, this is called the Discrete-Time Algebraic Ricatti Eqn (DARE). Solve for P and $V(x_k) = x_k^{\mathrm{T}} P x_k$ is the optimal value function.

Policy Iteration

So far, offline planning. Now, we show how the Bellman equation given fixed points equation that enable online methods. What's coming:

- Policy evaluation
- Policy improvement

<u>Policy evaluation</u>
Recall Bellman equation given some arbitrary policy π.

$$V_\pi(x_k) = c(x_k, u_{k)} + \gamma V_\pi(x_{k+1}),$$

where $x_{k+1} = f(x_k, u_k), u_k = \pi(x_k)$. Observation and a question: Implicit in V_π. Can we make iterative scheme?
For j=0, 1, ...

$$V_\pi^{j+1}(x_k) = c(x_k, u_k) + \gamma V_\pi^j(x_{k+1})$$

Does V_π^j converge as $j \longrightarrow \infty$? A: YES.
Iterative Policy Evaluation Algorithm: given arbitrary policy π, find V_π.

- Initialize $V_\pi^0, \ \forall x_k \in \mathcal{X}$
- For j=0, 1, ...,

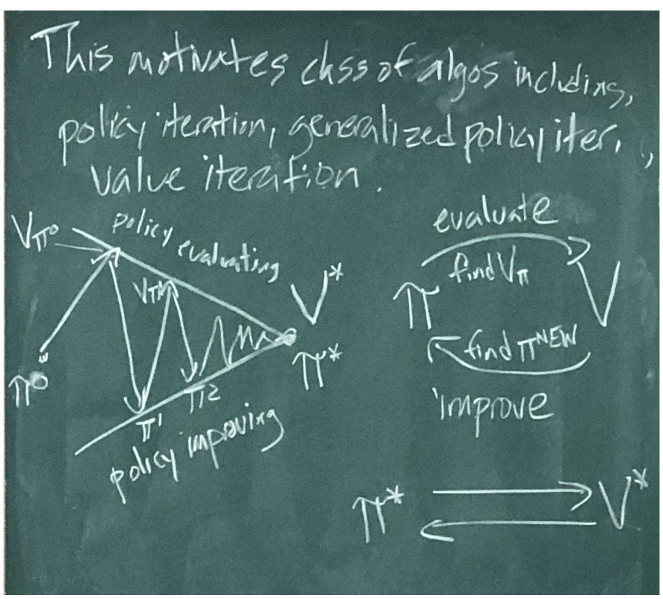
$$V_\pi^{j+1}(x_k) = c(x_k, u_k) + \gamma V_\pi^j(x_{k+1})$$

<u>Policy Improvement</u>
How to improve π? Use Bellman's Optimality equation. Given π^{old}

$$\pi^{\text{new}} = \text{argmin}_{\pi(\cdot)}\{c(x_k, u_k) + \gamma V_{\pi^{\text{old}}}(x_{k+1})\}$$

[Bertsekas 1996] has proved π^{new} is an improvement over π^{old} in the sense $\pi^{\text{new}} \leq \pi^{\text{old}} \ \ \forall x_k$

This motivates class of algorithm includes policy iteration, generalized policy iteration, value iteration.



Policy Iteration Algorithm:

1. Initialize with admissible policy
2. Policy evaluation
3. Policy improvement

Does j needs to go to ∞? No!
What if $j = 0, 1, \cdots < \infty$? Generalized policy iteration (GPI)
What if $j = 0$? Value Iteration (VI).
This converges!

Approximate Dynamic Programming

Now, we finally arrive at online adaptive optimal control methods (aka RL). You will see:

- Use data collected online from the system trajectories.
- integrate supervised learning (namely regression)

The methods are called: heuristic DP (Werbos '91, '92) and neuro-DP (Bersekas '96).
We need two more small concepts:

- temporal difference (TD) error
- value function approximation

Temporal Difference (TD) Error:
Recall Bellman Equation:

$$V_\pi(x_k) = c(x_k, \pi(x_t)) + \gamma \cdot V_\pi(x_{k+1})$$

To get an online adaptive method, consider the time-varying residual:

$$e_k = c(x_k, \pi(x_t)) + \gamma \cdot V_\pi(x_{k+1}) - V_\pi(x_k)$$

- If $e_k = 0$ for a given V_π, then it satisfies the Bellman equation and is consistent with π. Idea! Fit $V_\pi(\cdot)$ s.t. residuals are small, i.e. $\sum_k e_k^2$.

Value Function Approximation:
To perform least square regression with TD errors on V_π. we need to parameterize $V_\pi(x)$. Consider Weierstass Approximation Theorem

$$V_\pi(x) = \sum_{i=1}^{\infty} w_i \phi_i(x) = \sum_{i=1}^{L} w_i \phi_i(x) + \underbrace{\sum_{i=L+1}^{\infty} w_i \phi_i(x)}_{\epsilon_L}$$

$$\begin{aligned} V_\pi(x) &= W^{\mathrm{T}} \phi(x) + \epsilon_L \\ \text{where } \; \phi(x) &= [\phi_1(x), \phi_2(x), \ldots, \phi_L(x)]^{\mathrm{T}} \\ W &= [w_1, w_2, \ldots, w_L]^{\mathrm{T}} \end{aligned}$$

$\epsilon_L \longrightarrow 0$ uniformly in x as $L \longrightarrow \infty$. One of the main contributions of Werbes + Bertsekas was to used analyzed this approach where $\phi(x)$ is a neural network.
EX: LQR

We know $V(x_k) = x_k^{\mathrm{T}} P x_k, u_k = K x_k$ The TD error of LQR:

$$e_k = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + x_{k+1}^{\mathrm{T}} P x_{k+1} - x_k^{\mathrm{T}} P x_k$$

which is linear in P. Let's rewrite $V(x_k) = x_k^{\mathrm{T}} P x_k$ to apply linear least squares. $V(x_k) = x_k^{\mathrm{T}} P x_k = W^{\mathrm{T}} \phi(x)$ where $W = \mathrm{vec}(P)$, $\phi(x_k) = x_k \bigotimes x_k$, quadratic monomials of elements of x. E.g.

$$x_k = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, P = \begin{bmatrix} p_{11} & p_{12} \\ * & p_{22} \end{bmatrix},$$

$$\begin{aligned} x_k^{\mathrm{T}} P x_k &= p_{11} x_1^2 + 2 p_{12} x_1 x_2 + p_{22} x_2^2 \\ &= \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{22} \end{bmatrix}}_{W^{\mathrm{T}}} \underbrace{\begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}}_{\phi(x)} \end{aligned}$$

Note, because P is symmetric, we have $\frac{n(n+1)}{2}$ parameters. Using this parameterization. the LQR TD error is

$$\begin{aligned} e_k &= \underbrace{x_k^{\mathrm{T}} Q x_k^{\mathrm{T}} + u_k^{\mathrm{T}} R u_k^{\mathrm{T}}} + W^{\mathrm{T}} \phi(x_{k+1}) - W^{\mathrm{T}} \phi(x_k) \\ &= c(x_k, u_k) + W^{\mathrm{T}} [\gamma \phi(x_{k+1}) - \phi(x_k)] \end{aligned}$$

$Ax = b$

$$\begin{bmatrix} \gamma \phi(x_{k+1}) - \phi(x_k) \\ \gamma \phi(x_k) - \phi(x_{k-1}) \\ \dots \end{bmatrix}^{\mathrm{T}} \underbrace{W}_{L \times 1} = \begin{bmatrix} -c(x_k, u_k) \\ -c(x_{k-1}, u_{k-1}) \\ \dots \end{bmatrix}$$

Note: we have bypassed curse of dimensionality using TD error + value function approximation.

Online Approximate Dynamic Programming

Now we are positioned to write our first online RL algorithm.

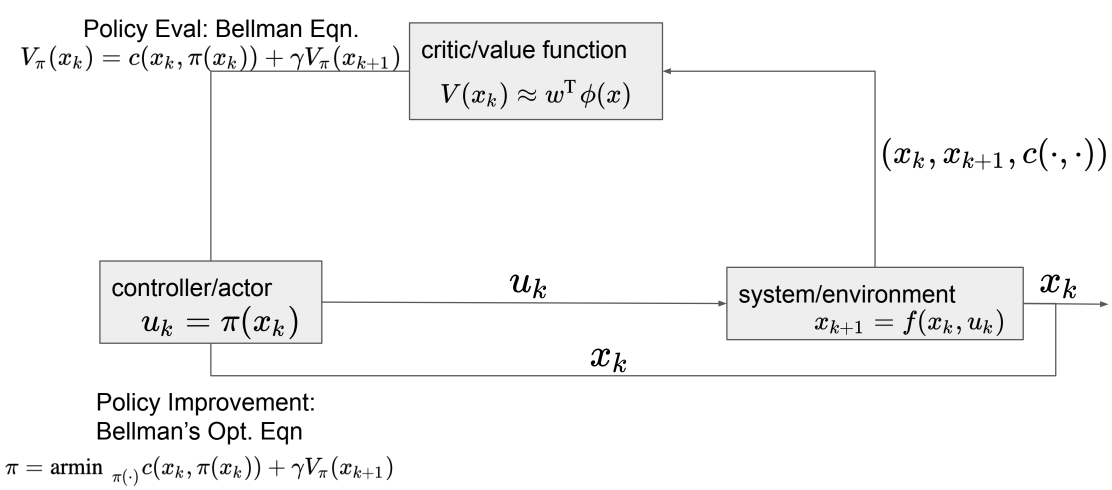


Figure 1: Flow Diagram

Remark 1:
Regression in the policy evaluation step can be executed using

Online Policy Iteration	Online Value Iteration
Intuition: Select any initial admissible control policy π^0. set $m = 0$ 1. Policy evaluation (fixed point iteration): $(x_k, x_{k+1}, c(x_k, \pi^m(x_k)))$ Collect measured data Find least square solution to $W^{\mathrm{T}}[\phi(x_k) - \gamma\phi(x_{k+1})] = c(x_k, \pi^m(x_k))$ 2. Policy improvement: $\pi^{m+1} = \text{argmin}_{\pi(\cdot)} \{c(x_k, \pi(x_k) + \gamma W_m^{\mathrm{T}} \phi(x_{k+1})\}$ where $x_{k+1} = f(x_k, \pi(x_k))]\forall x_k$	Intuition: Select any admissible π^0 set $m = 0$ $W_0 = 0$ 1. Policy evaluation: Collect measured data $(x_k, x_{k+1}, c(x_k, \pi^m(x_k)))$ Find least square solutions to $W_m^{\mathrm{T}}\phi(x_t) = c(x_k, \pi^m(x_k)) + \gamma W_{m-1}^{\mathrm{T}}\phi(x_{k+1})$ 2. Policy Improvement: Same

Figure 2: Summary

- batch least square
- recursive least square
- gradient method

<u>Remark 2:</u> In online PI,

$$\begin{bmatrix} \gamma\phi(x_{k+1}) - \phi(x_k) \\ \gamma\phi(x_k) - \phi(x_{k-1}) \\ \dots \end{bmatrix}^{\mathrm{T}} \underbrace{W}_{L\times 1} = \begin{bmatrix} -c(x_k, u_k) \\ -c(x_{k-1}, u_{k-1}) \\ \dots \end{bmatrix}$$

Actor-critic Method

Introduce a second function approximator for the control policy

$$u_k = \pi(x_k) = \mathbf{U}^{\mathrm{T}}\sigma(x_k)$$

where $\sigma(x_k) = [\sigma_1(x_k), \dots, \sigma_n(x_k)]^{\mathrm{T}}$, $\mathbf{U}^{\mathrm{T}} \in \mathbf{R}^{P\times M}$, are policy weights learned online. $\sigma(\cdot) : \mathbf{R}^n \longrightarrow \mathbf{R}^m$. Focus on 2.Policy improvement:

$$\begin{aligned} \min_{\mathbf{U}} \; & c(x_k, \mathbf{U}\sigma(x_k)) + \gamma W^{\mathrm{T}}\phi(x_{k+1}) \\ \text{s.t.} \quad & x_{k+1} = f(x_k, \mathbf{U}^{\mathrm{T}}\phi(x_k)) \end{aligned}$$

Define

$$T(\mathbf{U}) = c(x_k, \mathbf{U}\sigma(x_k)) + \gamma W^{\mathrm{T}}\phi(f(x_k, \mathbf{U}^{\mathrm{T}}\phi(x_k)))$$

Idea, gradient descent:

$$\underbrace{\mathbf{U}^{j+1}}_{M\times p} = \underbrace{\mathbf{U}^{j}}_{M\times p} - \beta \underbrace{\frac{dT}{d\mathbf{U}}}_{M\times p}(\mathbf{U})$$

$$\frac{dT}{d\mathbf{U}} = \sigma(x_k)[\frac{\partial c}{\partial \mathbf{U}}(x_k, \mathbf{U}^{\mathrm{T}}\sigma(x_k)) + \gamma W^{\mathrm{T}}\nabla\phi(x_{k+1})\frac{\partial f}{\partial \mathbf{U}}(x_k, \mathbf{U}^{\mathrm{T}}\sigma(x_k))]$$

Ex: LQR

$$\begin{aligned}
c(x,u) &= x_k^{\mathrm{T}} Q x_k + (\mathbf{U}^{\mathrm{T}}\sigma(x_k))^{\mathrm{T}} R(\mathbf{U}^{\mathrm{T}}\sigma(x_k)) \\
f(x,u) &= Ax_k + B\mathbf{U}^{\mathrm{T}}\sigma(x_k) \\
\frac{dT}{d\mathbf{U}} &= \sigma(x_k)[2R\mathbf{U}^{\mathrm{T}}\sigma(x_k) + \gamma W^{\mathrm{T}}\nabla\phi(x_{k+1})B]
\end{aligned}$$

Observation: "Model-based" Actor-critic is actually partially model free!

$$T(\mathbf{U}) = x_k^{\mathrm{T}} Q x_k + (\mathbf{U}^{\mathrm{T}}\sigma(x_k))^{\mathrm{T}} R(\mathbf{U}^{\mathrm{T}}\sigma(x_k)) + \gamma W^{\mathrm{T}}\phi(Ax + B\mathbf{U}^{\mathrm{T}}\sigma(x))$$

For $T(\mathbf{U})$ to be convex, $\mathbf{U}$ we need

- $R > 0$,
- $\phi(Ax+b)$ is convex in x, if ϕ is convex in argument.

Q-function, Q-learning

This section discuss model-free Rl. Key object is a generalization of the value function, called "Q-function." "Q" stands for "quality."
Introduced by Werbos 1974, '89, '91, '92 called "action-dependant" heuristic DP. Then [Watkins '89] proved convergence of Q-learning for discrete-time-value Markov Decision Processes.

Definition: Q-learning
Define $Q_\pi(x_k, u_k)$ as the Q-function associated with policy π, which gives cost from time step k onward, given the current state is x_k. You take a given control action u_k, then you follow policy π afterwards.i.e. at k given x_k

$$\begin{aligned}
k+1 &\quad x_{k+1} = f(x_k, u_k) \\
k+2 &\quad x_{k+2} = f(x_k, \pi(x_{k+1})) \\
k+3 &\quad x_{k+3} = f(x_k, \pi(x_{k+2})) \\
\cdots &
\end{aligned}$$

Note, in contrast to $V_\pi(x)$, $Q_\pi(x,u)$ depends on the control action.

$V(x_k)$	$Q(x_k, u_k)$
• Value function • State value function • Value • Cost-to-go $V(\cdot) : \mathbf{R}^n \longrightarrow \mathbf{R}$	• Quality function • State-action value function • Q-function $Q(\cdot,\cdot) : \mathbf{R}^n(x) \times \mathbf{R}^n(\mathcal{U}) \longrightarrow \mathbf{R}$

Figure 3: Table of Jargon

Key feature of $Q_\pi(x_k, u_k)$: It exposes current control action as a variable, enables model-free methods. But first, some identifies to relate $Q_\pi(x_k, u_k)$ to $V_\pi(x_k)$

1. $Q_\pi(x_k, \pi(x_k)) = V_\pi(x_k)$
2. Bellman's equation for Q-function. $Q_\pi(x_k, u_k) = c(x_k, u_k) + \gamma V_\pi(\underbrace{x_{k+1}}_{f(x_k,u_k)}) \ \forall x_k \in \mathcal{X}, \ \ u_k \in \mathcal{U}$.

Denote: $Q^*(x_k, u_k)$ the optimal Q-function, which is the minimal cost for k onwards, assuming we start in state x_k take given action u_k and follow the optimal policy afterwards. - Special case of Q_π which specifically minimizes our cost. We have from 2.

$$Q^*(x_k, u_k) = c(x_k, u_k) + \gamma V^*(\underbrace{x_{k+1}}_{f(x_k,u_k)})$$

Using Q^*, the Bellman's Optimally equation is simple! $V^*(x_k) = \min_u Q^*(x_k, u)$ and $\pi^*(x_k) = \text{argmin}_u Q^*(x_k, u)$.

EX. Q-function for LQR Consider the Q-function for a given control policy $u_k = \pi(x_k)$ (we know it's given linear for now $u_k = Kx_k$) for the LQR problem. The corresponding Q-function must satisfy the Bellman equation, namely

$$Q_\pi(x_k, u_k) = x_k^{\mathrm{T}} \overbrace{Q}^{\in \mathbf{R}^{\mathbf{n}\times\mathbf{n}}} x_k + u_k^{\mathrm{T}} R u_k + V_\pi(x)(\underbrace{x_{k+1}}_{=Ax_k+Bu_k})$$

We know for LQR,

$$V(x_k) = x_k^{\mathrm{T}} \underbrace{P}_{\text{solve a Riccati Eqn.}} x_k = Ax_k + Bu_k$$

$$Q_\pi(x_k, u_k) = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + (Ax_k + Bu_k)^{\mathrm{T}} P (Ax_k + Bu_k)$$

$$\text{Rewrite: } Q_\pi(x_k, u_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q + A^{\mathrm{T}}PA & B^{\mathrm{T}}PA \\ A^{\mathrm{T}}PB & R + B^{\mathrm{T}}PB \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

$$\text{Define: } Q_\pi(x_k, u_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} S_{xx} & S_{xu} \\ \times & S_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

$$\text{Apply: } \frac{\partial Q_\pi}{\partial u}(x_k, u) = 0 \ \text{ yield:}$$

$$\begin{aligned} u_k^* &= -S_{uu}^{-1} S_{xu} x_k \\ &= -(R + B^{\mathrm{T}}PB)^{-1}(B^{\mathrm{T}}PA)x_k \end{aligned}$$

This is intriguing, we somehow find a way to bypass ...
Case Study: Load frequency Control in Power System (MATLAB example)
Objective: Regulate frequency around a nominal val (e.g. 50Hz) by controlling a generator's output.
Model: Power systems are nonlinear system. We are going to use a linearize model to represent dynamics in local neighborhood of nominal state.

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$\text{where } A = \begin{bmatrix} -\frac{1}{T_p} & \frac{K_p}{T_p} & 0 & 0 \\ 0 & -\frac{1}{T_T} & -\frac{1}{T_T} & 0 \\ -\frac{1}{RT_k} & 0 & -\frac{1}{T_k} & -\frac{1}{T_E} \\ K_E & 0 & 0 & 0 \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{T_G} \\ 0 \end{bmatrix}$$

State is $x(t) = [\underbrace{\Delta f(t)}_{\text{incremental change in freq. around nominal value}}, \ \Delta P_g(t), \ \Delta X_g(t), \ \Delta E(t)]$

$$\min \sum_{k=0}^{\infty} [x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} u_k]$$

Policy iteration with Q-function	Value Iteration with Q-function
1. Policy evaluation: Set $Q_\pi^0(x_k,u_k)=0$ For given policy $\pi=\pi^m$ For j = 0,1,... $Q_\pi^{j+1}=c(x_k,u_k)+\gamma Q_\pi^j(x_{k+1},u_{k+1})$ Where $x_{k+1}=f(x_k,u_k)$ $u_{k+1}=\pi(x_k,u_k),\forall x_k\in\mathcal{X},u_k\in\mathcal{U}$ 2. Policy improvement: $\pi^{m+1}=\text{argmin}_u Q_{\pi^m}(x_k,u_k),\forall x_k$ $m\longleftarrow m+1$ Go to step 1	1. Policy evaluation: 1 step only! $Q_\pi^{m+1}=c(x_k,u_k)+\gamma Q_\pi^m(x_{k+1},u_{k+1})$ $x_{k+1}=f(x_k,u_k)$ $u_{k+1}=\pi^m(x_k,u_k),\forall x_k\in\mathcal{X},u_k\in\mathcal{U}$ 2. Policy improvement: Refer to left. Note: 1. And 2. Can be combined, $Q_\pi^{m+1}=c(x_k,u_k)+\gamma\min_u Q_\pi^m(x_{k+1},u_{k+1})$

Online method	Offline method
• Onboard • Synchronous • In operation	• Offboard • A-synchronous • Planning

Watkin's Q-learning Algorithm (1989)
- Let $\alpha_k\in[0,1]$ be "learning parameter"
- Idea: Update Q-function by taking convex combination of previous Q and a new Q suggested by value iteration.

$$\begin{aligned} Q^m(x_k,u_k) &= (1-\alpha_k)Q^{m-1}+\alpha_k\cdot Q^{m-1}_{\text{VI}} \\ &= (1-\alpha_k)Q^{m-1}+\alpha_k[c(x_k,u_k)+\gamma\min_u Q^{m-1}(x_{k+1},u)] \quad \underline{\text{- new Q according to VI}} \end{aligned}$$

often seen as : $Q^m(x_k,u_k)=Q^{m-1}(x_k,u_k)+\alpha_k[c(x_k,u_k)+\gamma\min_u Q^{m-1}(x_k,u)-Q^{m-1}(x_k,u_k)]$

Watkins' proved convergence to global optimum provided that

1. <u>all</u> state-action pairs are visited infinitely often over infinite time
2. $\sum_{k=1}^\infty \alpha_k=\infty,\ \sum_{k=1}^\infty\alpha_k^2<\infty$ - "Robbins-Moore Sequence"

Q: how to ensure (1)?
A: Randomized policy/control action. "ϵ-Greedy" policy is a common example that helps satisfy 1.

$$\underbrace{\Pi}_{Pr[u_k=u|x_k=x]}(u|x)=\begin{cases}(1-\epsilon)+\frac{\epsilon}{m} & \text{if } u^*=\text{argmin}_u Q(x,u)\\ \frac{\epsilon}{m}\ \text{ o.w. where } m=|\mathcal{U}| & \end{cases}$$

<u>Ex. Online Q-learning for LQR</u>
Objective: Find a feedback control policy $u_k=\pi(x_k)$ that solves $\min\sum_{k=1}^\infty x_k^{\mathrm{T}}Qx_+k+u_k^{\mathrm{T}}Ru_k$, subject to $x_kk+1=Ax_k+Bu+k$.
Learn optimal control online without knowledge of (A,B) from data $(x_k,x_{k+1},c(x_k,u_k))$.
We have seen that $Q(x,u)$ is quadratic

$$Q(x_k,u_k)=\begin{bmatrix}x_k\\u_k\end{bmatrix}^{\mathrm{T}}S\begin{bmatrix}x_k\\u_k\end{bmatrix}=z_k^{\mathrm{T}}Sz_k=Q(z_k)$$

Learning $Q_{x,u}$ comes down to learning matrix S. Now, S can be computed directly if we know (A, B), but in this case we will learn S from data. Write Q in parametric form:

$$Q(z_k) = \underbrace{W^{\mathrm{T}}}_{\text{vector of elements of } S} \underbrace{\phi(z_k)}_{\text{feature/basis vectors, quadratic terms in } z_k}$$

The Q-learning Bellman Equation

$$W_{j+1}^{\mathrm{T}}\phi(z_k) = x_k^{\mathrm{T}} Q x_+ k + u_k^{\mathrm{T}} R u_k + W_{j+1}^{\mathrm{T}}\phi(z_{k+1})$$
$$W_{j+1}^{\mathrm{T}}(\underbrace{\phi(z_k) - \phi(z_{k=1}}_{\text{regressor}}) = x_k^{\mathrm{T}} Q x_+ k + u_k^{\mathrm{T}} R u_k$$

Initialize: Select initial feedback gain $u_k = K^0 x_k$ at $j = 0$
Step j:

1. Learn Q-function online via least square
   - collect at k: $(x_k, x_{k+1}, u_k, u_{k+1}$ where $u_{k+1} = K x_{k+1}$
   - compute basis vectors: $\phi(z_t), \phi(z_{t+1})$
   - perform 1-step update of W using recursive least square (RLS) $W_{j+1}^{\mathrm{T}}(\phi(z_k) - \phi(z_{k=1}) = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k$
   - Repeat at time step $k+1$ with new data $(x_{k+1}, x_{k+2}, u_{k+1}, u_{k+2}$, until RLS converges $\longrightarrow W_{j+1}$.
2. Update the control policy
   - Unpack W_{j+1} into kernal matrix

$$Q(x_k, u_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} S_{xx} & S_{xu} \\ * & S_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

   - Update policy: $K^{j+1} = S_{uu}^{-1} S_{xu}$, $u_k =$ TBD

Class slides..