Introduction to Reinforcement Learning
A Short Course

Scott Moura

July 2020

1 Optimal Control

The canonical formulation is given by:

$$\begin{aligned}
\text{minimize}_{x_k,u_k} \quad & J = \sum_{k=0}^{N-1} c_k(x_k, u_k) + c_N(x_N) && (1)\\
\text{subject to:} \quad & x_{k+1} = f(x_k, u_k), \quad k = 0, \cdots, N-1 && (2)\\
& x_0 = x_{\text{init}} && (3)\\
& x_k \in \mathcal{X}_k, \quad u_k \in \mathcal{U}_k, \quad k = 0, \cdots, N && (4)
\end{aligned}$$

Notation: The most fundamental elements

- $x_k \in \mathbb{R}^n$: State variable
- $u_k \in \mathbb{R}^p$: Control variable (a.k.a. "action")

What is a state? Given a trajectory of inputs $u_0, \cdots, u_N$, the (initial) state x_0 is sufficient to completely predict the evolution of a dynamic system.

Equation (1) is the objective function

- $c_k(x_k, u_k)$: instantaneous or stage cost
- c_N : terminal cost

Infinite time horizon:

- discounted cost: $J = \lim_{N\to\infty} \sum_{k=0}^{N} \gamma^k \cdot c(x_k, u_k)$, $\gamma \in [0, 1]$
- average cost: $J = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} c(x_k, u_k)$

Why these formulations?

- Mathematical reason: Ensure cost remains finite.
- Conceptual reason: Stage costs are more uncertain in the future, so we discount their impact relative to immediate stage costs.

Examples:

1. "State regulation": $J = \sum_{k=0}^{\infty} \underbrace{x_k^T Q x_k}_{\text{send } x_k \text{ to } 0} + \underbrace{u_k^T R u_k}_{\text{don't use excessive control effort}}$
2. "State tracking": $J = \sum_{k=0}^{\infty} \underbrace{(x_k - x_k^{\text{ref}})^T Q (x_k - x_k^{\text{ref}})}_{\text{send } x_k \text{ close to } x^{ref}} + u_k^T R u_k$

3. "Minimum-time": $\text{minimize}_{x_k, u_k, N} \quad J = \sum_{k=0}^{N-1} 1$

Equation (2) are the dynamics paired with an initial condition:

$$x_{k+1} = f(x_k, u_k), \quad x_0 = x_{\text{init}} \tag{5}$$

The dynamics are a mathematical model of physical laws (differential equations), e.g.

- Newtonian mechanics
- Maxwell's equations
- Navier-Stokes
- Traffic conservation laws
- Laws of Thermodynamics
- (Electro-)Chemical Processes
- Infectious disease dynamics

Can be known/unknown, deterministic/stochastic

Equation (4) are "admissible" state and control/action sets

- $x_k \in \mathcal{X}_k, \qquad e.g. \underline{x}_k \leq x_k \leq \overline{x}_k$
- $u_k \in \mathcal{U}_k, \qquad \underline{u}_k \leq u_k \leq \overline{u}_k$

Objective: Find a control law (a.k.a. control "policy") of the form

$$u_k = \pi_k(x_k) \tag{6}$$

that solves the optimal control problem. Note that the control law/policy takes a "state feedback" form. That is, it maps states to actions. Examples:

1. Linear state feedback: $u_k = -K \cdot x_k, \qquad K \in \mathbb{R}^{p \times n}$
2. Neural network state feedback: $u_k = f_{NN}(x_k), \qquad f : \mathbb{R}^n \to \mathbb{R}^p$

2 Dynamic Programming

Dynamic programming is an algorithmic technique for solving optimal control problems by breaking it up into a recursion of sub-problems.

Consider the discounted cost optimal control formulation

$$\text{minimize} \quad J = \sum_{k=0}^{\infty} \gamma^k \cdot c(x_k, u_k), \qquad \gamma \in [0, 1] \tag{7}$$

$$\text{subject to:} \quad x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \cdots \tag{8}$$

Definition 1. (Value Function) Define $V(x_k)$ as the "value function," which represents the cumulative cost from time step k onward towards infinity, given the current state is x_k. Furthermore, let $V_\pi(x_k)$ represent the value function corresponding to control policy $u_k \pi(x_k)$, which may or may not be optimal.

Note $V_\pi(x_k) = c(x_k, u_k) + \gamma \cdot \underbrace{\sum_{\tau=k+1}^{\infty} \gamma^{\tau-(k+1)} \cdot c(x_\tau, u_\tau)}_{=V_\pi(x_{k+1})}$

$$V_\pi(x_k) = c(x_k, u_k) + \gamma \cdot V_\pi(x_{k+1}) \tag{9}$$

which can be used to recursively compute the value function corresponding to some policy $u_k = \pi(x_k)$.

We are now positioned to state Bellman's Principle of Optimality Equation:

$$\boxed{\begin{aligned} V(x_k) &= \min_{\pi(\cdot)} \left\{ c(x_k, \pi(x_k)) + \gamma \cdot V(x_{k+1}) \right\} && (10) \\ \text{where} \quad x_{k+1} &= f(x_k, \pi(x_k)) && (11) \end{aligned}}$$

The optimal policy is

$$\pi^\star(x_k) = \arg\min_{\pi(\cdot)} \left\{ c(x_k, \pi(x_k)) + \gamma \cdot V(x_{k+1}) \right\} \tag{12}$$

Remark 1. Bellman's Principle of Optimality Equation is also known as the discrete-time Hamilton-Jacobi-Bellman (HJMB) equation.

Note the following about Bellman's principle of optimality equation:

- The equation is recursive in $V(x_k)$
- Offline "planning" method

2.1 Case Study: Infinite-time Linear Quadratic Regulator (LQR)

Consider the classic and widely used linear quadratic regulator (LQR) optimal control problem

$$\text{minimize} \quad J = \sum_{k=0}^{\infty} \left[x_k^T Q x_k + u_k^T R u_k \right] \qquad \text{Note } \gamma = 1 \tag{13}$$

$$\text{subject to:} \quad x_{k+1} = A x_k + B u_k, \quad k = 0, 1, \cdots \tag{14}$$

$$x_0 = x_{\text{init}} \tag{15}$$

$$\text{where} \quad Q = Q^T \underbrace{\succeq 0}_{\text{p.s.d.}}, \quad R = R^T \underbrace{\succ 0}_{\text{p.d.}}, \quad Q_f = Q_f^T \succeq 0 \tag{16}$$

where "p.s.d." is a positive semi-definite matrix and "p.d." is a positive definite matrix. We will solve the LQR problem via dynamic programming to arrive at the so-called discrete-time algebraic Riccati equation (DARE).

We will discover that

- $V(x_k) = x_k^T P x_k$ (quadratic), $P = P^T \succeq 0$, and $P \in \mathbb{R}^{n \times n}$
- $\pi^\star(x_k) = K \cdot x_k$ (linear), $K \in \mathbb{R}^{p \times n}$

Bellman's (Principle of Optimality) equation for $V(x_k) = x_k^T P x_k$ is ...

$$V(x_k) = c(x_k, u_k) + \gamma \cdot V(x_{k+1}) \tag{17}$$
$$x_k^T P x_k = \left(x_k^T Q x_k + u_k^T R u_k\right) + 1 \cdot x_{k+1}^T P x_{k+1} \tag{18}$$

and substituting $u_k = K x_k$ gives

$$x_k^T P x_k = x_k^T \left[Q + K^T R K + (A + BK)^T P (A + BK)\right] x_k \tag{19}$$

Since this equation must hold for all x_k, we have the following matrix equation:

$$(A + BK)^T P (A + BK) - P + Q + K^T R K = 0 \tag{20}$$

This equation is linear in P, and is known as the "Lypanunov equation" to find P when K is fixed. It yields $P = P^T \succeq 0$ such that $V(x_k) = x_k^T P x_k$. That is:

$$V(x_k) = \sum_{\tau=k}^{\infty} x_\tau^T Q x_\tau + u_k^T R u_k \tag{21}$$
$$= \sum_{\tau=k}^{\infty} x_\tau^T \left[Q + K^T R K\right] x_\tau = x_k^T P x_k \tag{22}$$

To find an expression for K, write Bellman's optimality equation as

$$x_k^T P x_k = \min_w \left\{x_k^T Q x_k + w^T R w + (A x_k + B w)^T P (A x_k + B w)\right\} \tag{23}$$

differentiating with respect to (w.r.t.) w and setting to zero gives

$$2Rw + B^T P (A x_k + B w) = 0 \tag{24}$$
$$\Rightarrow w^\star = \underbrace{-(R + B^T P B)^{-1} B^T P A}_{=K} \cdot x_k \tag{25}$$

and thsu we have $u_k^\star = K \cdot x_k$ where $K = -(R + B^T P B)^{-1} B^T P A$. Substitute back into the Bellman equation and simplify to yield

$$A^T P A - P + Q - A^T P B (R + B^T P B)^{-1} B^T P A = 0 \tag{26}$$

which is quadratic in P and is known as the "Discrete-time Algebraic Riccati Equation" (DARE)

Summary of Infinite-time LQR:

$$u_k^{lqr} = K \cdot x_k \tag{27}$$
$$\text{where} \quad K = -(R + B^T P B)^{-1} B^T P A \tag{28}$$

and P solves:

$$A^T P A - P + Q - A^T P B (R + B^T P B)^{-1} B^T P A = 0 \tag{29}$$
$$\text{Value Function:} \quad V(x_k) = x_k^T P x_k \tag{30}$$

3 Policy Iteration & Value Iteration

So far, we have discussed offline designs via dynamic programming. We are also interested in online learning algorithms. Next we show that the Bellman equations are fixed point equations that enable forward-in-time methods for online learning.

Consider the discounted cost optimal control formulation over infinite time

$$\text{minimize} \quad \sum_{k=0}^{\infty} \gamma^k c(x_k, u_k) \qquad \gamma \in [0,1] \tag{31}$$

$$\text{subject to:} \quad x_{k+1} = f(x_k, u_k) \tag{32}$$

Define $V_\pi(x)$ as the value function corresponding to policy π (which may or may not be optimal). Note:

$$V_\pi(x_k) = \sum_{\tau=k}^{\infty} \gamma^{\tau-k} \cdot c(x_\tau, u_\tau) \tag{33}$$

$$= c(x_k, u_k) + \gamma \cdot \underbrace{\sum_{\tau=k+1}^{\infty} \gamma^{\tau-(k+1)} c(x_\tau, u_\tau)}_{=V_\pi(x_{k+1})} \tag{34}$$

$$V_\pi(x_k) = c(x_k, u_k) + \gamma \cdot V_\pi(x_{k+1}) \tag{35}$$

all where $u_k = \pi(x_k)$.

Observation & Question: This equation is implicit in $V_\pi(\cdot)$ and suggests the iterative scheme:

$$V_\pi^{j+1}(x_k) = c(x_k, u_k) + \gamma \cdot V_\pi^j(x_{k+1}) \qquad j = 0, 1, \cdots \tag{36}$$

$$V_\pi^0(x_k) = 0 \qquad \forall\, x_k \in \mathcal{X} \tag{37}$$

Q: Does V_π^j converge as $j \to \infty$? A: YES!

Algorithm 1. (Iterative Policy Evaluation Algorithm) To compute the value function corresponding to some arbitrary policy:

For $j = 0, 1, \cdots$,

$$V_\pi^{j+1}(x_k) = c(x_k, u_k) + \gamma \cdot V_\pi^j(x_{k+1}) \qquad \forall\, x_k \in \mathcal{X} \tag{38}$$

where $u_k = \pi(x_k)$,

$$V_\pi^0(x_k) = 0 \quad \forall\, x_k \in \mathcal{X} \tag{39}$$

Sutton & Barto refer to $V_\pi^j(x_k)$ as $j \to \infty$ as a "full backup".

Now that we have a method to evaluate a given policy, we wish to improve it. An intuitive idea is

$$\pi^{\text{NEW}} = \arg\min_{\pi(\cdot)} \{c(x_k, \pi(x_k)) + \gamma \cdot V_{\pi^{\text{OLD}}}(x_{k+1})\} \tag{40}$$

$$\text{where} \quad x_{k+1} = f(x_k, \pi(x_k)) \tag{41}$$

Bertsekas [1996] has proven that π^{NEW} is improved from π^{OLD} in the sense that $V_{\pi^{\text{NEW}}}(x_k) \le V_{\pi^{\text{OLD}}}(x_k)$ $\forall$ $x_k \in \mathcal{X}$. We call this step "policy improvement".

SUMMARY

Policy Evaluation
Given an arbitrary policy π, find V_π
For $j = 0, 1, \cdots$
$V_\pi^{j+1} = c(x_k, u_k) + V_\pi^j(x_{k+1})$
$V_\pi^0(x_k) = 0 \;\forall\, x_k \in \mathcal{X}$
where $u_k = \pi(x_k)$, $x_{k+1} = f(x_k, u_k)$

Policy Improvement
Given $V_{\pi^{\text{OLD}}}$ for some arbitrary policy π^{OLD}, find improved policy π^{NEW} such that $V_{\pi^{\text{NEW}}}(x_k) \le V_{\pi^{\text{OLD}}}(x_k) \;\forall\, x_k \in \mathcal{X}$
$\pi^{\text{NEW}} = \arg\min_{\pi(\cdot)} \{c(x_k, \pi(x_k)) + \gamma \cdot V_{\pi^{\text{OLD}}}(x_{k+1})\}$
where $x_{k+1} = f(x_k, \pi(x_k))$

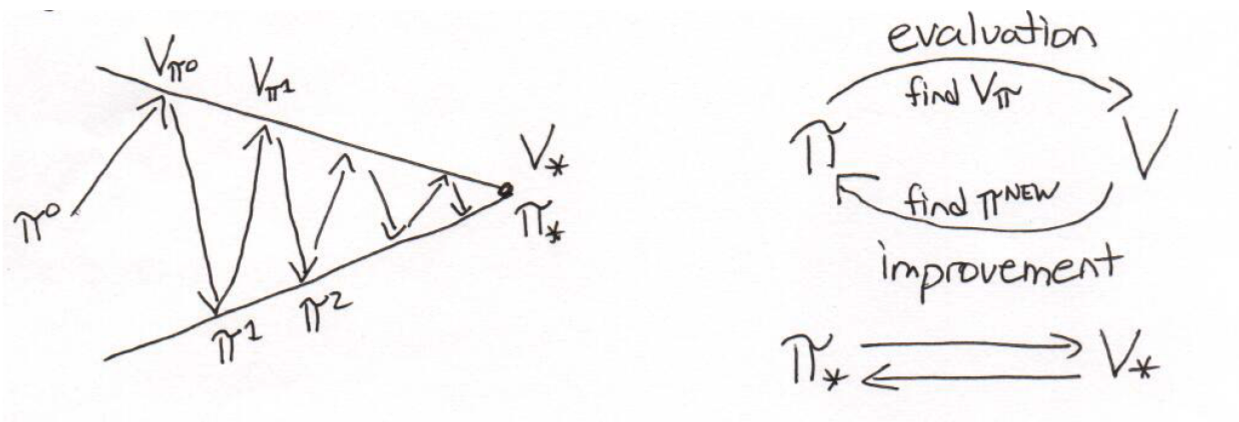

Figure 1: Schematic of Policy Iteration Algorithm

This motivated a class of algorithms called Policy Iteration (PI) and Generalized Policy Iteration (GPI), and Value Iteration (VI).

Algorithm 2. (Policy Iteration (PI) Algorithm [Sutton & Barto, Bertsekas])

1. Initialize admissible policy π^0. Set $m = 0$.
2. Policy Evaluation: Set $V_\pi(x_k) = 0 \ \forall \ x_k \in \mathcal{X}$. $\pi \leftarrow \pi^m$.
   for $j = 0, 1, \cdots$
$$V_\pi^{j+1} = c(x_k, u_k) + V_\pi^j(x_{k+1}), \quad \forall \ x_k \in \mathcal{X} \tag{42}$$
$$\text{where } u_k = \pi(x_k), \quad x_{k+1} = f(x_k, u_k) \tag{43}$$
3. Policy Iteration: Set $V_{\pi^{\mathrm{OLD}}} \leftarrow V_\pi^{j+1}$
$$\pi^{\mathrm{NEW}} = \arg\min_{\pi(\cdot)} \{c(x_k, \pi(x_k)) + \gamma \cdot V_{\pi^{\mathrm{OLD}}}(x_{k+1})\} \tag{44}$$
$$\text{where } x_{k+1} = f(x_k, \pi(x_k)) \tag{45}$$
   Set $\pi^{m+1} = \pi^{\mathrm{NEW}}$, $m \leftarrow m + 1$.
   Go to Step 2.

Remark 2. The policy evaluation step can be computationally onerous if $j \to \infty$, and we must perform each iteration for all $x_k \in \mathcal{X}$.

- Q: Can we truncate to M iterations? A: YES! Generalized Policy Iteration (GPI) algorithm.
- Q: Can we truncate to 1 iteration? A: YES! Value Iteration (VI) algorithm.
- Ref: [Howard 1960], [Puterman 1978] for theory. [Sutton & Barto, Bertsekas] for textbooks.

3.1 Case Study: LQR

Consider the linear quadratic regulator (LQR) problem:

$$\text{minimize} \quad \sum_{k=0}^{\infty} \left[x_k^T Q x_k + u_k^T R u_k\right] \tag{46}$$
$$\text{subject to:} \quad x_{k+1} = A x_k + B u_k, \quad k = 0, 1, \cdots \tag{47}$$

Recall that

- $V(x_k) = x_k^T P x_k$
- $u_k = K x_k$

Let's apply the policy evaluation and policy improvement steps!

1. Policy Evaulation: substitute into Bellman equation

$$x_k^T P^{j+1} x_k = x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P^j (Ax_k + Bu_k) \qquad (V = x^T P x) \tag{48}$$

$$x_k^T P^{j+1} x_k = x_k^T \left[Q + K^T R K + (A + BK)^T P^j (A + BK)\right] x_k \quad \forall\; x_k \qquad (u = Kx) \tag{49}$$

$$\Rightarrow \qquad P^{j+1} = Q + K^T R K + (A + BK)^T P^j (A + BK) \qquad \text{for some } K \tag{50}$$

which provides an iterative solution to the Lyapunov equation.

2. Policy Improvement: Recall from before:

$$\min_w \left\{ x_k^T Q x_k + w^T R w + (Ax_k + Bw)^T P^{\mathrm{OLD}} (Ax_k + Bw) \right\} \tag{51}$$

differentiating w.r.t. w, setting to zero, and re-arranging gives

$$w^\star = \underbrace{-\left(R + B^T P^{\mathrm{OLD}} B\right)^{-1} B^T P^{\mathrm{OLD}} A}_{=K^{\mathrm{NEW}}} \cdot x_k \tag{52}$$

We now have everything to state an iterative method to solve the infinite-time LQR optimal control problem, using policy evaluation and policy improvement.

3. Summary

$$\boxed{\begin{aligned} &\text{Summary of Iterative Method to solve Infinite-time LQR} && (53)\\ u_k &= K^{j+1} \cdot x_k && (54)\\ K^{j+1} &= -\left(R + B^T P^{j+1} B\right)^{-1} B^T P^{j+1} A && (55)\\ P^{j+1} &= Q + (K^j)^T R K^j + (A + BK^j)^T P^j (A + BK^j) && (56) \end{aligned}}$$

In the control systems community, this is called Hewer's method [Hewer 1971].

4 Approximate Dynamic Programming (ADP)

In this section, we finally arrive at methods to perform online adaptive optimal control (i.e. reinforcement learning) using data measured along the system trajectories. You will see that these methods incorporate supervised learning (regression, in particular) and can be model-based or model-free. These methods are broadly called "approximate dynamic programming" [Werbos 1991, 1992] or "neruo-dynamic programming" [Bertsekas 1996].

First, we require two concepts:

1. the temporal difference error, and
2. value function approximation

4.1 Temporal Difference (TD) Error

The Bellman equation used for policy evaluation (shown below) can be thought of as a consistency equation for the value function of a given policy π.

$$V_\pi(x_k) = c(x_k, \pi(x_k)) + \gamma \cdot V_\pi(x_{k+1}) \tag{57}$$

To turn this into an online adaptive method, consider time-varying residual:

$$e_k = c(x_k, \pi(x_k)) + \gamma \cdot V_\pi(x_{k+1}) - V_\pi(x_k) \tag{58}$$

The symbol e_k is known as the "temporal different (TD) error," which is simply RL jargon for the residual in Bellman's equation.

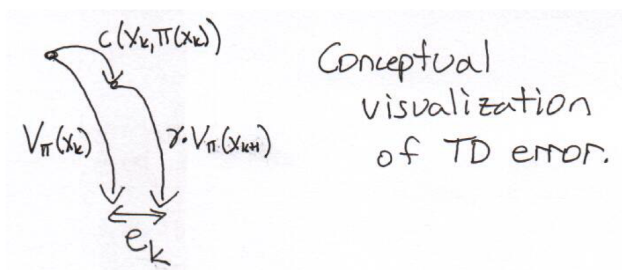

Figure 2: Conceptual visualization of TD error

Suppose that at each time step k we collect data $(x_k, x_{k+1}, c(x_k, \pi(x_k))$. We can use this data for supervised learning. For example, we can fit a regression model for V_π such that it minimizes the sum of squared residuals, e_k, i.e. perform least squares on the TD errors. Next we describe the regression models to approximate the value function.

4.2 Value Function Approximation

To perform supervised learning on V_π using the TD errors (i.e. Bellman equation residuals), we must parameterize $V_\pi(\cdot)$ is some way.

Consider the Weierstrass higher order approximation theorem: There exists a (dense) basis set $\{\phi_i(x)\}$ such that

$$V_\pi(x) = \sum_{i=1}^{\infty} w_i \phi_i(x) = \sum_{i=1}^{L} w_i \phi_i(x) + \underbrace{\sum_{i=L+1}^{\infty} w_i \phi_i(x)}_{\varepsilon_L} \tag{59}$$

$$= W^T \phi(x) + \varepsilon_L \tag{60}$$

$$\text{where} \quad \phi(x) = [\phi_1(x), \phi_2(x), \cdots, \phi_L(x)]^T \tag{61}$$

$$W = [w_1, w_2, \cdots, w_L]^T \tag{62}$$

and $\varepsilon_L \to 0$ uniformly in x as $L \to \infty$.

One of the main contributions of Werbos and Bertsekas was to use neural networks for the regressor vector $\phi(x)$.

4.3 Example: LQR

In LQR problems, we established:

- $V(x_k) = x_k^T P x_k$ is quardratic
- $u_k = K x_k$ is linear

Then the TD error for LQR is:

$$e_k = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} - x_k^T P x_k \tag{63}$$

which is linear in the parameter matrix P. Let's re-write $V(x_k) = x_k^T P x_k$ to enable linear (least squares) regression:

$$V(x_k) = x_k^T P x_k = W^T \phi(x) \tag{64}$$

where $W = \mathrm{vec}(P)$ stacks the columns of P matrix into vector W, and $\phi(x) = x_k \otimes x_k$ is a vector of monomials of x_k. This is best understood by example. Consider:

$$x_k = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad P = \begin{bmatrix} p_{11} & p_{12} \\ * & p_{22} \end{bmatrix} \tag{65}$$

then

$$\begin{aligned}
x_k^T P x &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ * & p_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} p_{11}x_1 + p_{12}x_2 \\ p_{12}x_1 + p_{22}x_2 \end{bmatrix} && (66) \\
&= p_{11}x_1^2 + p_{12}x_2x_1 + p_{12}x_1x_2 + p_{22}x_2^2 && (67) \\
&= \underbrace{\begin{bmatrix} p_{11} & 2p_{12} & p_{22} \end{bmatrix}}_{=W^T} \underbrace{\begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}}_{=\phi(x)} && (68) \\
&= W^T \phi(x) && (69)
\end{aligned}$$

Note that because P is symmetric we have $\phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n(n+1)/2}$. Using this parameterization, we can re-write the LQR TD error as:

$$\begin{aligned}
e_k &= \underbrace{x_k^T Q x_k + u_k^T R u_k}_{=c(x_k,u_k)} + W^T \phi(x_{k+1}) - W^T \phi(x_k) && (70) \\
&= c(x_k, u_k) + W^T \left[\phi(x_{k+1}) - \phi(x_k)\right] && (71)
\end{aligned}$$

The TD error can be computed for supervised learning by, at each time step k, collecting data $(x_k, x_{k+1}, c(x_k, u_k))$.

Remark 3. Previously, DP algorithms required evaluation of Bellman's equation at all $x_k \in \mathcal{X}$. To achieve this computationally, we considered a discrete-valued state space $\mathcal{X}$. This results in an exponential increase in calculations as the state vector size increases, known as the "curse of dimensionality". Value function approximation (and policy approximation) bypasses this challenge!