Introduction to Reinforcement Learning:Part IV

Dr. Mayank Shekhar JHA
Slides contribution: Soha KANSO

Maitre de Conférences (Associate Professor),
CRAN,
UMR 7039, CNRS,
Polytech Nancy,
Office: C 225,
Université de Lorraine
France.

Table of Contents

1 Temporal Difference

2 Function Approximation

3 RL: Nonlinear Discrete Time
   Forward in time Learning
   Neural network based approximation

Table of Contents

1. Temporal Difference
2. Function Approximation
3. RL: Nonlinear Discrete Time
   - Forward in time Learning
   - Neural network based approximation

Introduction

- learn online the solutions to optimal control problems without knowing the full system dynamics.
- leads to true online reinforcement learning
- control actions are improved in real time based on estimating their value functions by observing data measured along the system trajectories.
- based on the Bellman equation and solve Policy Evaluation equation by using data observed along a single trajectory of the system.
- TD learning is applicable for feedback control applications

Learning Along System Trajectories

- Temporal difference reinforcement learning methods are based on the Bellman equation and solve equations of PI, without using systems dynamics knowledge, but using data observed along a single trajectory of the system.
- Temporal difference updates the value at each time step as observations of data are made along a trajectory.
- Periodically, the new value is used to update the policy. Temporal difference methods are related to adaptive control in that they adjust values and actions online in real time along system trajectories.

Recursive relationship for *Value function*, REMINDER

The value of the policy $\pi(x, u)$ can be written as

$$V_k^{\pi}(x) = E_{\pi}\left\{J_k \mid x_k = x\right\} = E_{\pi}\left\{\sum_{i=k}^{k+T} \gamma^{i-k} r_i \mid x_k = x\right\},$$

$$V_k^{\pi}(x) = E_{\pi}\left\{r_k + \gamma \sum_{i=k+1}^{k+T} \gamma^{i-(k+1)} r_i \mid x_k = x\right\},$$

$$V_k^{\pi}(x) = \sum_{u} \pi(x, u) \sum_{x'} P_{xx'}^{u}\left[R_{xx'}^{u} + \gamma V_{k+1}^{\pi}\left(x'\right)\right]$$

$$V^{\pi}\left(x_k\right) = E_{\pi}\left\{r_k \mid x_k\right\} + \gamma E_{\pi}\left\{V^{\pi}\left(x_{k+1}\right) \mid x_k\right\}$$

Temporal Difference Equation

Temporal difference reinforcement learning uses one sample path, namely the current system trajectory, to update the value.
Value update can be replaced as:

$$V^{\pi}(x_k) = r_k + \gamma V^{\pi}(x_{k+1})$$

which holds for each observed data experience set (x_k, x_{k+1}, r_k) at each time stage k. This data set consists of the current state x_k, the observed cost incurred r_k, and the next state x_{k+1}.

TD Equation

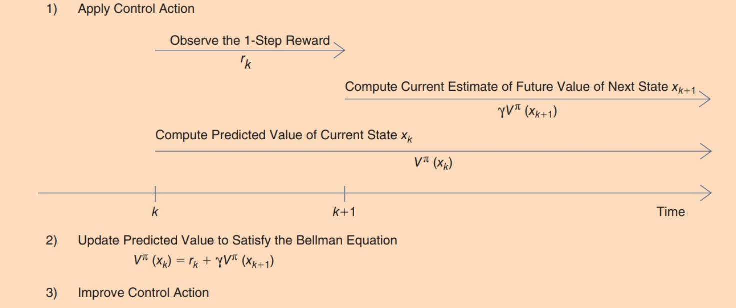
The temporal difference error is defined as

$$e_k = -V^{\pi}(x_k) + r_k + \gamma V^{\pi}(x_{k+1})$$

and the value estimate is updated to make the temporal difference error small.

- $V^{\pi}(x_k)$ may be considered as a predicted performance or value,
- r_k as the observed one-step reward,
- $\gamma V^{\pi}(x_{k+1})$ as a current estimate of future.

TD Equation


1) Apply Control Action
Observe the 1-Step Reward
r_k
Compute Current Estimate of Future Value of Next State x_{k+1}
$\gamma V^{\pi}(x_{k+1})$
Compute Predicted Value of Current State x_k
$V^{\pi}(x_k)$
k
$k+1$
Time
2) Update Predicted Value to Satisfy the Bellman Equation
$V^{\pi}(x_k) = r_k + \gamma V^{\pi}(x_{k+1})$
3) Improve Control Action

TD Equation: Observations

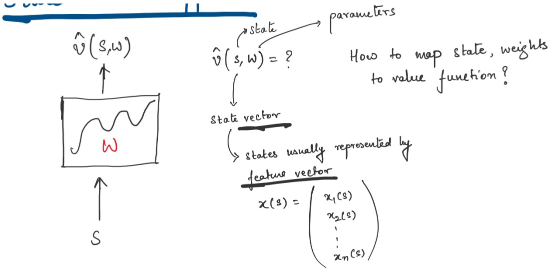
- The Bellman equation can be interpreted as a consistency equation that holds if the current estimate for the predicted value $V^{\pi}(x_k)$ is correct.
- Temporal difference methods update the predicted value estimate $\hat{V}^{\pi}(x_k)$ to make the temporal difference error small.
- Idea: TD application on Bellman's equation repeatedly in policy iteration or value iteration, then on average these algorithms converge toward the solution of the stochastic Bellman equation.

Table of Contents

1. Temporal Difference
2. Function Approximation
3. RL: Nonlinear Discrete Time
   - Forward in time Learning
   - Neural network based approximation

Motivation

- Policy iteration and value iteration can be implemented for a finite MDP by storing and updating lookup tables.
- For dynamical systems with infinite state and action spaces is to approximate the value function by a suitable approximator structure in terms of unknown parameters.



Value Function approximation

For nonlinear systems → the value function contains higher order nonlinearities. Then, according to the Weierstrass higher order approximation theorem, there exists a dense basis set $\{\varphi_i(x)\}$ such that

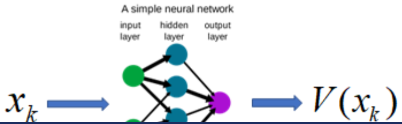
$$
\begin{aligned}
V(x) &= \sum_{i=1}^{\infty} w_i \varphi_i(x) \\
&= \sum_{i=1}^{L} w_i \varphi_i(x) + \sum_{i=L+1}^{\infty} w_i \varphi_i(x) \equiv W^T \phi(x) + \varepsilon_L(x),
\end{aligned}
$$

where basis vector $\phi(x) = [\varphi_1(x)\varphi_2(x)\cdots\varphi_L(x)] : R^n \to R^L$ and $\varepsilon_L(x)$ converges uniformly to zero as the number of terms retained $L \to \infty$.

NN based approximation of Value function: CRITIC neural Network

- Value function is sufficiently smooth over compact space
- Consider dense basis set $\{\phi_i(x)\}$ with basis vector (Weierstrass Theorem): $\phi(x) = [\varphi_1(x)\varphi_2(x)...\varphi_L(x)] : \mathbb{R}^n \to \mathbb{R}^L$

$V_\pi(x) = \sum_{i=1}^{L} w_i \varphi_i(x) = W^T \phi(x)$
Substituting in Bellman TD equation:
$e_k = r(x_k, \pi_{x_k}) + W^T \phi(x_{k+1}) - W^T \phi(x_k)$



NN based approximation of control policy: ACTOR neural Network

Control policy approximation: Actor neural Network

Introducing a second neural network for the control policy, known as the actor neural network. Consider a parametric approximator structure for the control action

$$u_k = \pi(x_k) = U^T \sigma(x_k),$$

with $\sigma(x) : R^n \rightarrow R^M$ being a vector of M activation functions and $U \in R^{M \times m}$ being a matrix of weights or unknown parameters. In the LQR, the optimal state feedback is linear in the states so that the basis set $\sigma(x)$ can be taken as the state vector.

Table of Contents

1. Temporal Difference
2. Function Approximation
3. RL: Nonlinear Discrete Time
    - Forward in time Learning
    - Neural network based approximation

RL: Discrete time optimal control

System

$$x_{k+1} = f(x_k) + g(x_k)u(x_k) \tag{1}$$

- $x_k \in \Omega \subset \mathbb{R}^n$ is the state variable vector
- Ω being a compact set
- $u(x_k) \in U \subset \mathbb{R}^m$ is the control input vector
- $f(x)$ is C^1 and $x = 0$ is an equilibrium state such that $f(0) = 0$ and $g(0) = 0$.

Note: $u(x_k)$ will be denoted as u_k.

Dr. Mayank S JHA, mayank-shekhar.jha@univ-lorraine.fr
Polytech Nancy, CRAN, University of Lorraine, France

RL: Discrete time optimal control

Control law/ Policy

A control policy is a function from state space to control space $\pi(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^m$, that defines for every state x_k , a control action:

$$u_k = \pi(x_k) \tag{2}$$

- Such mappings $\rightarrow$ feedback controllers.
- Example: linear state-variable feedback $u_k = \pi(x_k) = -Kx_k$

RL: Discrete time optimal control

Goal directed performance

Cost-to-go is a sum of (discounted) future costs from the current time k into the infinite horizon future under a prescribed control law $u_k = \pi(x_k)$:

$$J(x_k, u_k) = \sum_{n=k}^{\infty} r(x_n, u_n) \tag{3}$$

where $r(x_n, u_n)$ is the utility function defined as:
$r(x_n, u_n) = x_n^T Q x_n + u_n^T R \, u_n$

- Q symmetric positive semi-definite matrix $Q = Q^T \geq 0$
- R is a symmetric positive definite matrix $R = R^T > 0$.

$\gamma = 1$

Special Case: $\gamma = 1$

For simplicity, $\gamma = 1$ in what follows for non-linear discrete time case

RL: Discrete time optimal control

Assumption: Stabilizable system

System (1) is *stabilizable* on the prescribed set $\Omega \in \mathbb{R}^n$.

$\Rightarrow$ There is a control policy $u_k^1 = \pi(x)$ such that closed loop system $x_{k+1} = f(x_k) + g(x_k)u_k^1$ is asymptotically stable over Ω i.e. $u_k^1 = (u^1(x_k), u^1(x_{k+1}), u^1(x_{k+2}), ...u^1(x_\infty))$ exists that

- that stabilises the system (1)
- associated cost $J(x_k, u_k^1)$ is finite.

U denotes the set of all admissible control inputs.

RL: Discrete time optimal control

For a given admissible prescribed policy $\pi(x)$,
the cost associated is called as it *value* denoted as
$V_\pi(x_k) = J(x_k, \pi(x))$

RL: Discrete time optimal control

Objective: Optimal Cost

To find a control policy $\pi^*(x_k)$ that minimizes the infinite horizon cost function,

$$V^*(x_k) = \min_{u_k \in U} \sum_{n=k}^{\infty} r(x_n, u_n), \forall x_k \tag{4}$$

or, $V^*(x_k) = \min_{\pi(\cdot)} \sum_{n=k}^{\infty} r(x_n, \pi(x_n)), \forall x_k$

Optimal policy

Optimal control policy: $\pi^*(x_k) = \arg\min_{\pi(\cdot)} \sum_{n=k}^{\infty} r(x_n, \pi(x_n)), \forall x_k$

RL: Discrete time optimal control

Cost (given a prescribed policy $u_k = \pi(x_k)$)

$$V_\pi(x_k) = \sum_{n=k}^{\infty} r(x_n, u_n), \forall x_k$$

$$V_\pi(x_k) = r(x_k, u_k) + V_\pi(x_{k+1})$$

Bellman Eq/ Nonlinear Lyapunov Eq (Recursive): Hamiltonian:

$$H(x_k, u_k, V_\pi) = r(x_k, u_k) + V_\pi(x_{k+1}) - V_\pi(x_k)$$

Optimal Cost:

$$V^*(x_k) = \min_{u_k \in U} \left(r(x_k, u_k) + V_\pi(x_{k+1}) \right)$$

RL: Discrete time optimal control

Bellman Principle Bellman, 1957

"An optimal policy has the property that no matter what the previous decisions (i.e. controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions

Dr. Mayank S JHA, mayank-shekhar.jha@univ-lorraine.fr Polytech Nancy, CRAN, University of Lorraine, France

RL: Discrete time optimal control

Cost (given a prescribed policy $u_k = \pi(x_k)$)

$$V_\pi(x_k) = \sum_{n=k}^{\infty} r(x_n, u_n), \forall x_k$$

$$V_\pi(x_k) = r(x_k, u_k) + V_\pi(x_{k+1})$$

Bellman Eq/ Nonlinear Lyapunov Eq (Recursive): Hamiltonian:

$$H(x_k, u_k, V_\pi) = r(x_k, u_k) + V_\pi(x_{k+1}) - V_\pi(x_k)$$

Optimal Cost:

$$V^*(x_k) = \min_{u_k \in U} \left(r(x_k, u_k) + V_\pi(x_{k+1})\right)$$

Bellman principle:

$$V^*(x_k) = \min_{u_k \in U} \left(r(x_k, u_k) + V^*(x_{k+1})\right)$$

Backwards in Time!!

Optimal control (policy):

$$\pi^*(x_k) = \arg\min_{u_k \in U} \left(r(x_k, u_k) + V^*(x_{k+1})\right)$$

RL: Discrete time optimal control

Cost (given a prescribed policy $u_k = \pi(x_k)$)

$$V_\pi(x_k) = \sum_{n=k}^{\infty} r(x_n, u_n), \forall x_k$$

$$V_\pi(x_k) = r(x_k, u_k) + V_\pi(x_{k+1})$$

Bellman Eq/ Nonlinear Lyapunov Eq (Recursive):

Hamiltonian:

$$H(x_k, u_k, V_\pi) = r(x_k, u_k) + V_\pi(x_{k+1}) - V_\pi(x_k)$$

Optimal Cost:

$$V^*(x_k) = \min_{u_k \in U} \left(r(x_k, u_k) + V_\pi(x_{k+1})\right)$$

Bellman principle:

$$V^*(x_k) = \min_{u_k \in U} \left(r(x_k, u_k) + V^*(x_{k+1})\right)$$

Optimal control (policy):

$$\pi^*(x_k) = \arg\min_{u_k \in U} \left(r(x_k, u_k) + V^*(x_{k+1})\right)$$

Only data required!!

RL: Discrete time optimal control

Bellman principle: *(DT Hamilton-Jacobi-Bellman Equation)*

$$V^*(x_k) = \min_{u_k \in U} \left(r(x_k, u_k) + V^*(x_{k+1}) \right)$$
$$= \min_{u_k \in U} \left(x_k^T Q x_k + u_k^T R\, u_k + V^*(x_{k+1}) \right)$$
$$= \min_{u_k \in U} \left(x_k^T Q x_k + u_k^T R\, u_k + V^*(f(x_k) + g(x_k) u_k) \right)$$

Optimal control (policy):

$$\pi^*(x_k) = \arg\min_{u_k \in U} \left(r(x_k, u_k) + V^*(x_{k+1}) \right)$$
$$\pi^*(x_k) = u_k^* = (-1/2) R^{-1} g^T(x_k) \frac{\partial V^*(x_{k+1})}{\partial x_{k+1}}$$

Dr. Mayank S JHA, mayank-shekhar.jha@univ-lorraine.fr Polytech Nancy, CRAN, University of Lorraine, France

RL: Discrete time optimal control

Cost (given a prescribed policy $u_k = \pi(x_k)$)

$$V_\pi(x_k) = \sum_{n=k}^{\infty} r(x_n, u_n), \forall x_k$$

Bellman Eq/ Nonlinear Lyapunov Eq (Recursive):

$$V_\pi(x_k) = r(x_k, u_k) + V_\pi(x_{k+1})$$

Optimal Cost:

$$V^*(x_k) = \min_{u_k \in U} \left(r(x_k, u_k) + V_\pi(x_{k+1})\right)$$

Bellman principle:

$$V^*(x_k) = \min_{u_k \in U} \left(r(x_k, u_k) + V^*(x_{k+1})\right)$$

Optimal control (policy):

$$\pi^*(x_k) = \arg\min_{u_k \in U} \left(r(x_k, u_k) + V^*(x_{k+1})\right)$$

DT Policy Iteration

Initialization

Select any *stabilizing* /admissible control policy: $\pi_j(x_k)$

Policy Evaluation

Determine the *Value* under the current policy using Bellman Equation/Nonlinear Lyapunov Eq.

$$V_{j+1}(x_k) = r(x_k, \pi_j(x_k)) + V_{j+1}(x_{k+1}) \ ; \ V_{j+1}(0) = 0$$

Policy Improvement

Determine an improved policy

$$\pi_{j+1}(x_k) = \arg\min_{u_k \in U} \left(r(x_k, u_k) + V_{j+1}(x_{k+1}) \right)$$

DT Policy Iteration

Initialization

Select any *stabilizing* /admissible control policy: $\pi_j(x_k)$

Policy Evaluation

Determine the *Value* under the current policy using Bellman Equation/Nonlinear Lyapunov Eq.

$V_{j+1}(x_k) = r(x_k, \pi_j(x_k)) + V_{j+1}(x_{k+1})$; $V_{j+1}(0) = 0$

Policy Improvement

Determine an improved policy

$\pi_{j+1}(x_k) = \arg\min_{u_k \in U} \left(r(x_k, u_k) + V_{j+1}(x_{k+1}) \right)$

DT Policy Iteration

Initialization

Select any *stabilizing* /admissible control policy: $\pi_j(x_k)$

Policy Evaluation

Determine the *Value* under the current policy using Bellman Equation/Nonlinear Lyapunov Eq.

$$V_{j+1}(x_k) = r(x_k, \pi_j(x_k)) + V_{j+1}(x_{k+1}) \; ; \; V_{j+1}(0) = 0$$

Policy Improvement

Determine an improved policy

$$\pi_{j+1}(x_k) = \arg\min_{u_k \in U} \left(r(x_k, u_k) + V_{j+1}(x_{k+1}) \right)$$

DT Policy Iteration

Initialization

$\pi_j(x_k)$

Policy Evaluation

$V_{j+1}(x_k) = r(x_k, \pi_j(x_k)) + V_{j+1}(x_{k+1})$

Policy Improvement

$$\pi_{j+1}(x_k) = \arg\min_{u_k \in U} \left(r(x_k, u_k) + V_{j+1}(x_{k+1})\right)$$

When $r(x_k, u_k) = x_k^T Q x_k + u_k^T R\, u_k$,

$$\pi_{j+1}(x_k) = (-1/2) R^{-1} g^T(x_k) \frac{\partial V_{j+1}(x_{k+1})}{\partial x_{k+1}}$$

DT Policy Iteration: Observations

- Initial policy must be stabilizing.
- Policy Iteration (Howard, 1960; Leake and Liu, 1967) $\Rightarrow$
  - $V_{j+2}(x_k) \leq V_{j+1}(x_k)$
- As $j \to \infty$:
  - $V_j(x_k) \to V^*(x_k)$
  - $\pi_j \to \pi^*$
- Convergence to optimal cost and thus, optimal control policy.

DT Policy Iteration: Observations

- $V_{j+1}(x_k) = r(x_k, \pi_j(x_k)) + V_{j+1}(x_{k+1}); \forall x_k \in \Omega$
  - value of using a given policy starting in all current states possible.
  - Several states $\Rightarrow$ Significant computations!
- Called *full backup* (Sutton and Barto, 2018)$\Rightarrow$ Massive computational load
- Bellman Eq $\rightarrow$ *fixed point equation*
  - Given admissible policy π_j, $V^{i+1}(x_k) = r(x_k, \pi_j(x_k)) + V^i(x_{k+1})$ is a *contraction map*
  - Upon iterated starting from $V^0(x_k)$, $V^i(x_k) \rightarrow V_{j+1}(x_k)$ as $i \rightarrow \infty$.

DT Policy Iteration: Observations

- $V_{j+1}(x_k) = r(x_k, \pi_j(x_k)) + V_{j+1}(x_{k+1}); \forall x_k \in \Omega$
  - value of using a given policy starting in all current states possible.
  - Several states $\Rightarrow$ Significant computations!
- Called *full backup* (Sutton and Barto, 2018)$\Rightarrow$ Massive computational load
- Bellman Eq $\rightarrow$ *fixed point equation*
  - Given admissible policy π_j, $V^{i+1}(x_k) = r(x_k, \pi_j(x_k)) + V^i(x_{k+1})$ is a *contraction map*
  - Upon iterated starting from $V^0(x_k)$, $V^i(x_k) \rightarrow V_{j+1}(x_k)$ as $i \rightarrow \infty$.

Issues

- This strategy $\Rightarrow$ *backward in time procedure*
- Good for:
  - Off-line planning, Offline optimization, Offline control synthesis.
  - NOT online leanring (optimla control synthesis using real time data measured along system trajectories.
- Exact solutions: very difficult
  - Large state space
  - Highly nonlinear dynamics

Issues

- This strategy $\Rightarrow$ *backward in time procedure*
- Good for:
  - Off-line planning, Offline optimization, Offline control synthesis.
  - NOT online leanring (optimla control synthesis using real time data measured along system trajectories.
- Exact solutions: very difficult
  - Large state space
  - Highly nonlinear dynamics

Issues

- This strategy $\Rightarrow$ *backward in time procedure*
- Good for:
  - Off-line planning, Offline optimization, Offline control synthesis.
  - NOT online learning (optimal control synthesis using real time data measured along system trajectories.
    Temporal Difference (TD) or *forward in time learning*
- Exact solutions: very difficult
  - Large state space
  - Highly nonlinear dynamics
    Value Function approximation (VFA): Neural Networks

Forward-in-time Learning

Temporal Difference Error (TD error):
$e_k = r(x_k, \pi_{x_k}) + V_\pi(x_{k+1}) - V_\pi(x_k)$

- RHS is DT Hamiltonian
- If Bellman Eq holds, e_k is zero.
- Linear in x.
- Thus, given a policy $\pi(x)$, Least Square based solution at each time k for $e_k = 0$.

NN based approximation

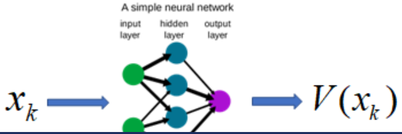
Value Function approximation (VFA): Neural Networks

- Value function is sufficiently smooth over compact space
- Consider dense basis set $\{\phi_i(x)\}$ with basis vector (Weierstrass Theorem): $\phi(x) = [\varphi_1(x)\varphi_2(x)...\varphi_L(x)] : \mathbb{R}^n \to \mathbb{R}^L$

$V_\pi(x) = \sum_{i=1}^{L} w_i \varphi_i(x) = W^T \phi(x)$

Substituting in Bellman TD equation:

$e_k = r(x_k, \pi_{x_k}) + W^T \phi(x_{k+1}) - W^T \phi(x_k)$



Online DT Policy Iteration

Initialization Choose an initial stabilizing policy (admissible): $\pi_0(x_k)$

Policy Evaluation

$$V_{j+1}(x_k) = r(x_k, \pi_j(x_k)) + V_{j+1}(x_{k+1})$$

$$\mathrm{r}(\mathrm{x}_k, \pi_{x_k}) = W_{j+1}^T(\phi(x_k) - \phi(x_{k+1}))$$

Policy Improvement

$$\pi_{j+1}(x_k) = \arg\min_{u_k \in U} \left(r(x_k, u_k) + W_{j+1}^T(\phi(x_{k+1})) \right)$$

With $r(x_k, u_k) = x_k^T Q x_k + u_k^T R\, u_k$,

$$\pi_{j+1}(x_k) = (-1/2) R^{-1} g^T(x_k) \nabla\phi^T(x_{k+1}) W_{j+1}$$

Online Policy Iteration

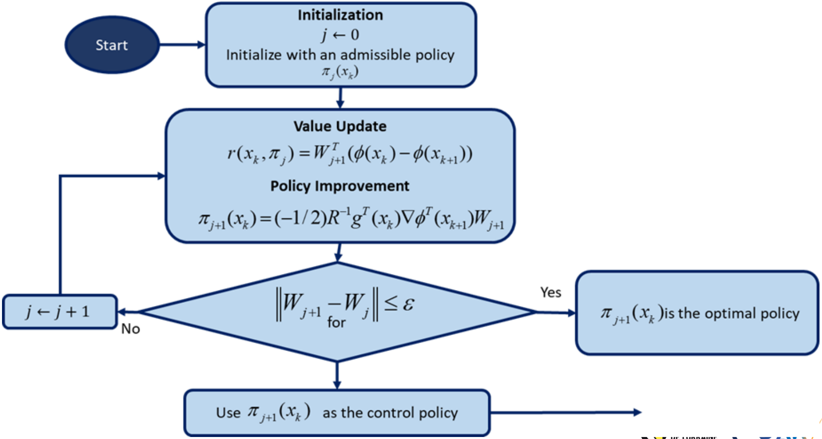


Figure: Online PI

Online DT Policy Iteration: Observations

- At $k+1$: observe $x_k, u_k = \pi_j(x_k), x_{k+1}$
- Calculate $r(x_k, u_k)$ One scalar Equation in $r(x_k, \pi_{x_k}) = W_{j+1}^T(\phi(x_k) - \phi(x_{k+1}))$
- Use same policy $u_k = \pi_j(x_k)$, collect L data $\Rightarrow$ L equations $(!!\phi(x) = \mathbb{R}^n \rightarrow \mathbb{R}^L)$.
- Determine LS based solution $\hat{W}_{j+1}$
- Repeat till $\hat{W}_{j+1} \equiv \hat{W}_{j+2} \rightarrow W^*$ Apply Improved control

Batch Least Square Approach

- Can be implemented online by standard system identification techniques.
- Note that weight update is a scalar equation, whereas the unknown parameter vector $W_{j+1} \in R^L$ has L elements. Therefore, data from multiple time steps are needed for its solution.
- At time $k+1$ we measure the previous state x_k, the control $u_k = \pi_j(x_k)$, the next state x_{k+1}, and compute the resulting utility $r(x_k, \pi_j(x_k))$.
- These data result in one scalar equation.
- This procedure is repeated for subsequent times using the same policy $\pi_j(\cdot)$ until at least L equations are obtained, at which point the least-squares solution W_{j+1} can be determined. Batch leastsquares can be used for this procedure.

Recursive Least square approach

- Express Critic Weight update as:
  $$W_{j+1}^T \Phi(k) \equiv W_{j+1}^T \left(\phi\left(x_k\right) - \gamma\phi\left(x_{k+1}\right)\right) = r\left(x_k, \pi_j\left(x_k\right)\right)$$

with $\Phi(k) \equiv \left(\phi\left(x_k\right) - \gamma\phi\left(x_{k+1}\right)\right)$ being a regression vector.

- At step j of the policy iteration algorithm, the control policy is fixed at $u = \pi_j(x)$.
- Then, at each time k the data set $\left(x_k, x_{k+1}, r\left(x_k, \pi_j\left(x_k\right)\right)\right)$ is measured.
- One step of RLS is then performed.
- This procedure is repeated for subsequent times until convergence to the parameters corresponding to the value $V_{j+1}(x) = W_{j+1}^T \phi(x)$.
- For RLS to converge, the regression vector $\Phi(k) \equiv \left(\phi\left(x_k\right) - \gamma\phi\left(x_{k+1}\right)\right)$ must be persistently exciting.

Online + ONPOLICY Policy Iteration

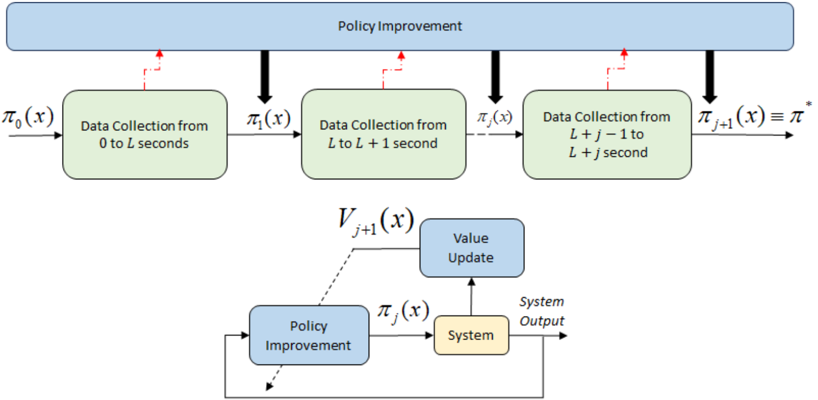


Figure: Online ON-POLICY

Execution: Adaptive Critic Structures

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$$

Control law update

$$u_{j+1}(x_{k+1}) = -\frac{1}{2} R^{-1} g(x_k)^T \frac{dV_j(x_{k+1})}{dx_{k+1}}.$$

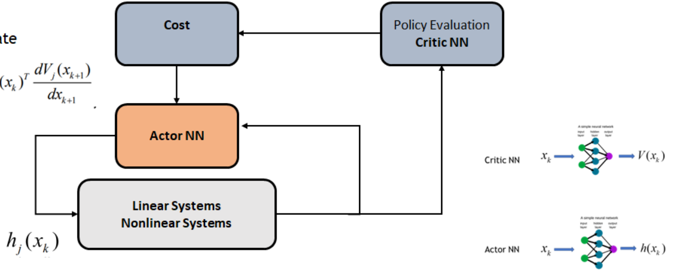


Figure: Actor Critic Structure

Execution: Adaptive Critic Structures: Two time Scale!

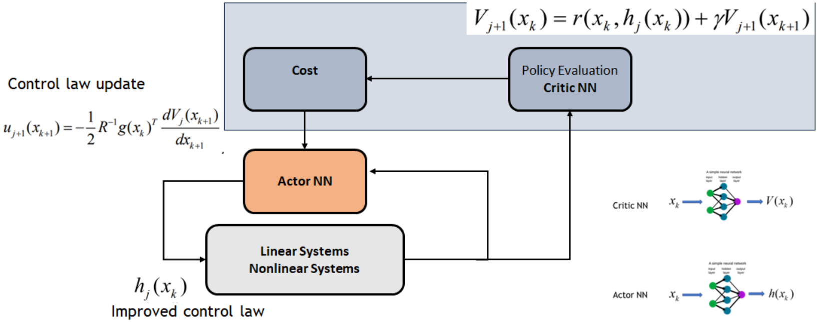


Figure: Actor Critic Structure

Execution: Adaptive Critic Structures

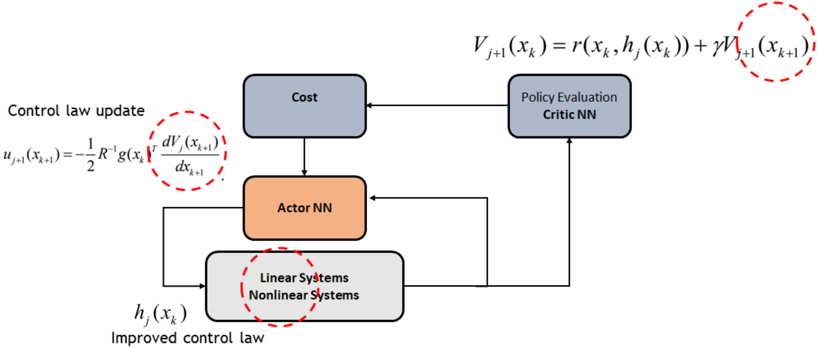


Figure: Actor Critic Structure

Execution: Adaptive Critic Structures

Data-driven control learning

NOT

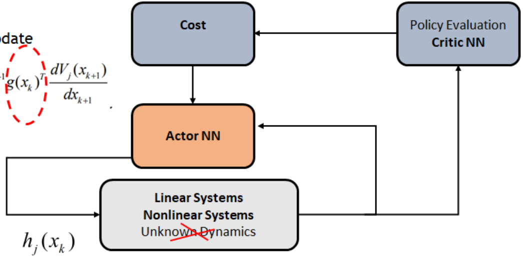
Model Free!

Value update

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$$

Control law update

$$u_{j+1}(x_{k+1}) = -\frac{1}{2} R^{-1} g(x_k)^T \frac{dV_j(x_{k+1})}{dx_{k+1}}$$



$h_j(x_k)$

Improved control law

Figure: Actor Critic Structure

Execution: Adaptive Critic Structures

RL: Model free approach ➔ Q-function

System $x_{k+1} = f(x_k) + g(x_k)u_k$

$$V_h(0) = 0$$

$$Q_h(x_k, u_k) = r(x_k, u_k) + \gamma V_h(x_{k+1})$$

$$Q_h(x_k, u_k) = r(x_k, u_k) + \gamma Q_h(x_{k+1}, h(x_{k+1}))$$

$$V^*(x_k) = \min_{u_k}(Q^*(x_k, u_k))$$

$$h^*(x_k) = \arg\min_{u_k}(Q^*(x_k, u_k))$$

$$u^*(x_k) = -\frac{1}{2}R^{-1}g(x_k)^T \cancel{\frac{dV^*(x_{k+1})}{dx_{k+1}}}$$

References I

Lewis, F. L., Vrabie, D., & Vamvoudakis, K. G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, *32*(6), 76-105.

Bellman, R. (1957).A markovian decision process. *Journal of mathematics and mechanics*, 679–684.

Howard, R. A. (1960).Dynamic programming and markov processes..

Leake, R., & Liu, R.-W. (1967).Construction of suboptimal control sequences. *SIAM Journal on Control*, *5*(1), 54–63.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.