Lecture 2

Foundations of Reinforcement Learning and Optimal Control

Thomas Gallien[1,2]

[1]Institute for Robotics and Flexible Production
JOANNEUM RESEARCH

[2]RL Community
AI Austria

Reinforcement Learning Bootcamp, 25.09.2024

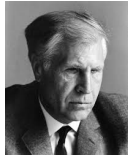
A Brief Look at History

Richard Bellman

Dynamic Programming

1950 Optimal Control

Lev Pontryagin



Dimitri Bertsekas

1980 Reinforcement Learning

Richard Sutton

Andrew Barto

2010

DeepMind

OpenAI

Temporal Difference Methods

How can we estimate the value function?

- Monte Carlo Prediction: $V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha\left[G_t - V(\mathbf{s}_t)\right]$
- Problems:
  - G_t can only be calculated at the end of an episode
  - How can we deal with infinite horizon problems?
- Approach (TD-Prediction):
  - Estimate the value function by means of bootstrapping

$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha \underbrace{\left[R_{t+1} + \gamma V(\mathbf{s}_{t+1}) - V(\mathbf{s}_t)\right]}_{:=\delta_t \text{ TD-error or TD(0)}}$$

On-Policy TD Control

Algorithm SARSA

Require: learning rate $\alpha \in (0, 1]$, discount rate $\gamma \in (0, 1]$, $\epsilon \in (0, 1)$

Initialize $Q(\mathbf{s}, \mathbf{a})$ arbitrarily for all $\mathbf{s} \in S^+, \mathbf{a} \in \mathcal{A}(\mathbf{s})$, $Q(\mathbf{s}, .) = 0$ for all $\mathbf{s} \in S^{\text{term}}$
for each episode **do**
    Initialize $\mathbf{s}$
    Sample $\mathbf{a}$ from $\mathbf{s}$ using policy $\pi(\mathbf{a}|\mathbf{s})$ (e.g., ϵ-greedy)
    for each step of the episode **do**
        Take action $\mathbf{a}$, observe reward r and next state $\mathbf{s}'$
        Sample $\mathbf{a}'$ from $\mathbf{s}'$ using policy $\pi(\mathbf{a}|\mathbf{s})$ (e.g., ϵ-greedy)
        $Q(\mathbf{s}, \mathbf{a}) \leftarrow Q(\mathbf{s}, \mathbf{a}) + \alpha \left[r + \gamma Q(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a}) \right]$
        $\mathbf{s} \leftarrow \mathbf{s}'$, $\mathbf{a} \leftarrow \mathbf{a}'$
        if $\mathbf{s}'$ is terminal **then**
            break
        end if
    end for
end for

Off-Policy TD Control

Algorithm Q-Learning

Require: learning rate $\alpha \in (0, 1]$, discount rate $\gamma \in (0, 1]$, $\epsilon \in (0, 1)$

Initialize $Q(\mathbf{s}, \mathbf{a})$ arbitrarily for all $\mathbf{s} \in S^+, \mathbf{a} \in \mathcal{A}(\mathbf{s})$, $Q(\mathbf{s}, .) = 0$ for all $\mathbf{s} \in S^{\text{term}}$
for each episode **do**
    Initialize $\mathbf{s}$
    for each step of the episode **do**
        Sample $\mathbf{a}$ from $\mathbf{s}$ using policy $\pi(\mathbf{a}|\mathbf{s})$ (e.g., ϵ-greedy)
        Take action $\mathbf{a}$, observe reward r and next state $\mathbf{s}'$
        $Q(\mathbf{s}, \mathbf{a}) \leftarrow Q(\mathbf{s}, \mathbf{a}) + \alpha \left[r + \gamma \max_{\mathbf{a}} Q(\mathbf{s}', \mathbf{a}^*) - Q(\mathbf{s}, \mathbf{a}) \right]$
        $\mathbf{s} \leftarrow \mathbf{s}'$
        if $\mathbf{s}'$ is terminal **then**
            break
        end if
    end for
end for

The Pros and Cons of Q-learning

Pros:

- Model-free
- Off-policy learning
- Reduced noise compared MC-methods
- Significant higher sample efficiency compared to MC-methods
- Easy to implement

Cons:

- Requires a discrete finite action space
- **Maximation Bias**
  - Overestimation of action values due to max operator
  - Coupling of action selection and evaluation due to $\max_{\mathbf{a}} Q(\mathbf{s}', \mathbf{a}^*)$
  - Workaround: Double Q-learning

$$Q_1(\mathbf{s}, \mathbf{a}) \leftarrow Q_1(\mathbf{s}, \mathbf{a}) + \alpha \left[r + \gamma \max_{\mathbf{a}} Q_2(\mathbf{s}', \mathbf{a}^*) - Q_1(\mathbf{s}, \mathbf{a}) \right]$$

$$Q_2(\mathbf{s}, \mathbf{a}) \leftarrow Q_2(\mathbf{s}, \mathbf{a}) + \alpha \left[r + \gamma \max_{\mathbf{a}} Q_1(\mathbf{s}', \mathbf{a}^*) - Q_2(\mathbf{s}, \mathbf{a}) \right]$$

$$a \sim \text{Cat}\left(\left\{ \frac{1-\epsilon}{2} : \arg\max_{a'} Q_1(s, a'),\ \frac{1-\epsilon}{2} : \arg\max_{a'} Q_2(s, a'),\ \epsilon : \text{U}(A) \right\} \right)$$

Grid World Example

Goal: Find the way to heaven

Reward: Every step costs 0.1, except tax, heaven and hell fields.

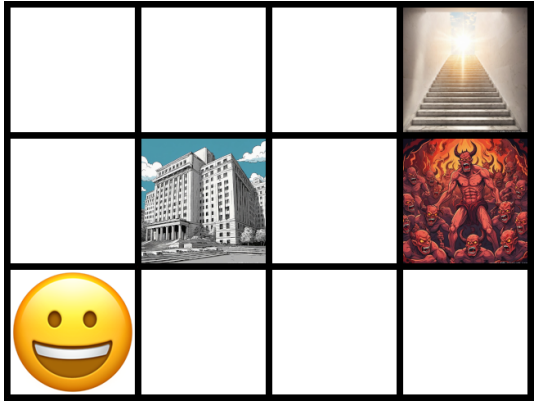


Figure: Start

Grid World Example

Goal: Find the way to heaven

Reward: Every step costs 0.1, except tax, heaven and hell fields.

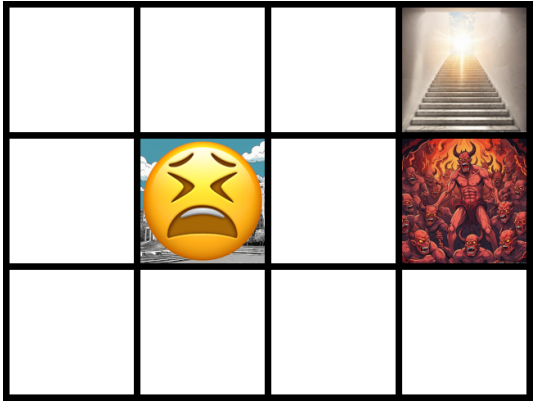


Figure: Pay your taxes: reward = -0.5.

Grid World Example

Goal: Find the way to heaven

Reward: Every step costs 0.1, except tax, heaven and hell fields.

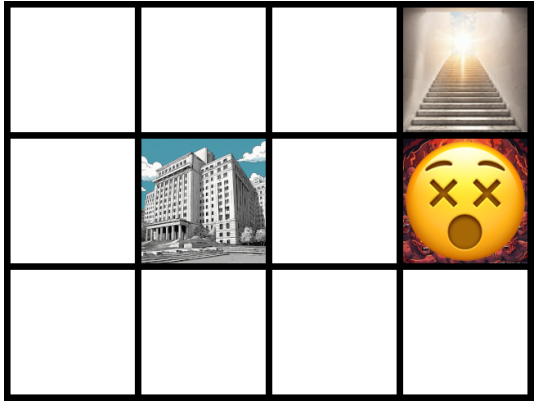


Figure: Terminal state for sinners: reward = -1.

Grid World Example

Goal: Find the way to heaven

Reward: Every step costs 0.1, except tax, heaven and hell fields.

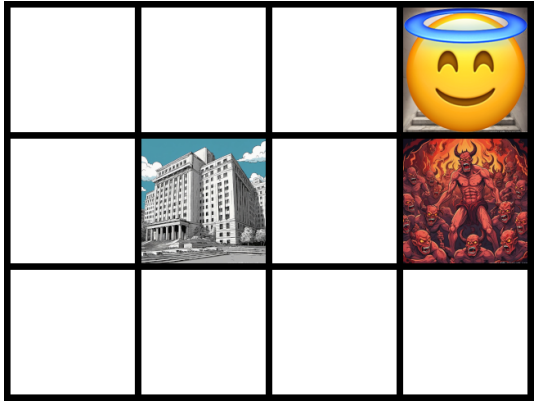


Figure: Terminal state for saints: reward = 1.

Grid World Results

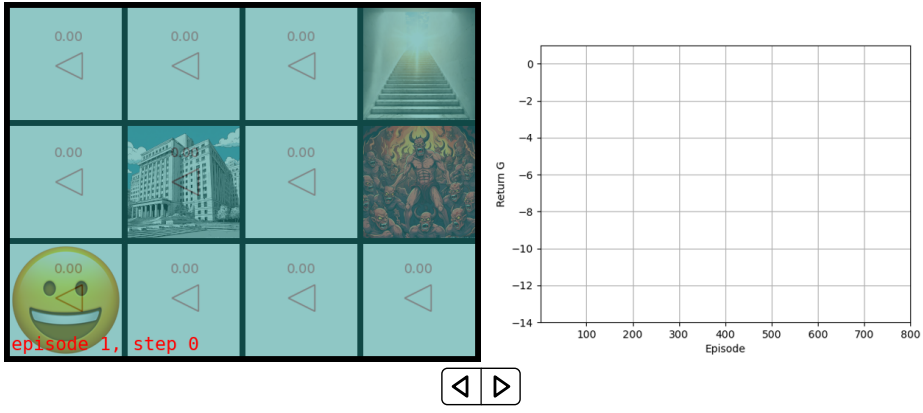
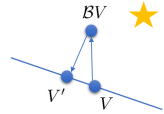


Figure: Q-learning agent in action.

From Tabular Methods to Function Approximators

- Tabular value estimation seems to be tedious
- What do with very large state spaces (or even continuous ones)?
- We need a method that scales ...
  $\Rightarrow$ function approximators
- Regression Operator Π: $\|\Pi V - \Pi \bar{V}\|_2 \leq \|V - \bar{V}\|_2$
- Bellman Operator $\mathcal{B}$: $\|\mathcal{B}V - \mathcal{B}\bar{V}\|_\infty \leq \gamma \|V - \bar{V}\|_\infty$
- Fitted Value iteration: $V \leftarrow \Pi \mathcal{B} V$
  - $\Pi\mathcal{B}$ is not a contraction
  - **does not converge in general**

(a) Tabular case: Updates guarantee better estimate.



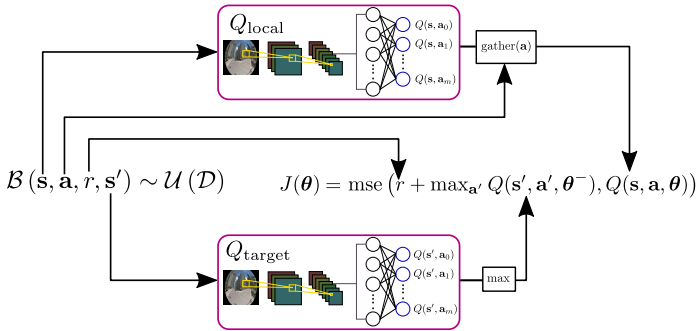
(b) Fitted case: Update does not guarantee better estimate.

Figure: Comparison Tabular vs. fitted value iteration[1].

[1] Picture shamelessly taken from Sergey Levine's course slides (CS285).

Deep Q-Learning

- Sample ϵ-greedy actions according local Q network
- Store experiences in experience buffer $\mathcal{D}$
- Update local network by random sampling batch from $\mathcal{D}$

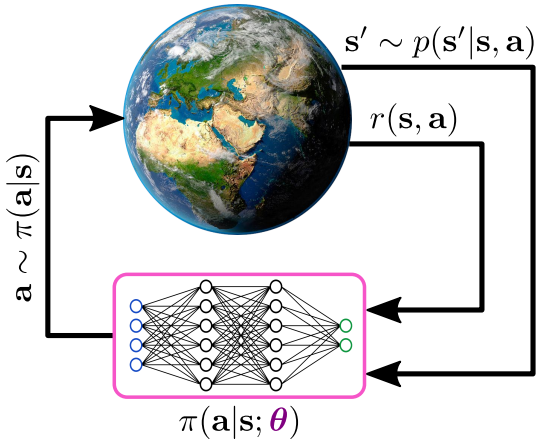


- $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$ every M^{th} iteration or slowly parameter mixing

Policy-Gradient Methods

How can we deal with continuous action spaces?

- Suppose we have a policy $\pi(\mathbf{a}|\mathbf{s})$, parameterised by learnable vector $\boldsymbol{\theta}$.
- We can observe at least a chunk of a trajectory τ up to the horizon T being composed as state-action tuples $\tau = \{(\mathbf{s}_0, \mathbf{a}_0), (\mathbf{s}_1, \mathbf{a}_1) \ldots, (\mathbf{s}_{T-1}, \mathbf{a}_{T-1}), \mathbf{s}_T\}$
- Derive an update rule for the reinforcement learning objective $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right]$



How To Derive the Policy Gradient

First we use the identify $f(x)' = f(x) \ln f(x)'$ to express the gradient in terms of the log-density

$$\nabla_\theta J = \nabla_\theta \int_{\mathcal{S}_\tau} p(\tau) \left[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] d\tau = \int_{\mathcal{S}_\tau} p(\tau) \nabla_\theta \ln p(\tau) \left[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] d\tau$$

We can simplify the gradient of the (log) joint-density of the state-action pairs by marginalisation

$$\begin{aligned}
\nabla_\theta \ln p(\tau) &= \nabla_\theta \ln \left(p\left(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, \mathbf{s}_T\right) \right) \\
&= \nabla_\theta \ln \left(p(\mathbf{s}_T | \mathbf{a}_{T-1}, \mathbf{s}_{T-2}, \ldots) p(\mathbf{a}_{T-1} | \mathbf{s}_{T-1}, \ldots) \ldots p(\mathbf{a}_0 | \mathbf{s}_0) p(\mathbf{s}_0) \right) \\
&= \nabla_\theta \ln \left(p(\mathbf{s}_0) \prod_{t=0}^{T-1} \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{a}_t, \mathbf{s}_t) \right) = \sum_{t=0}^{T-1} \nabla_\theta \ln \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)
\end{aligned}$$

Plugging this back the gradient of the objective reveals to

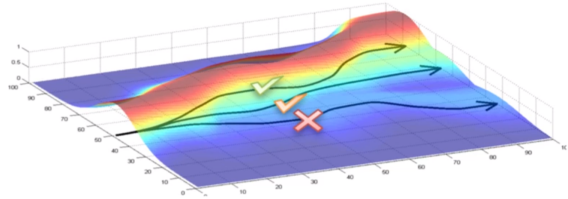
$$\nabla_\theta J = \int_{\mathcal{S}_\tau} p(\tau) \sum_{t=0}^{T-1} \nabla_\theta \ln \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \left[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] d\tau.$$

How to Derive the Policy Gradient

Monte Carlo integration reveals the estimate of the policy gradient

$$\nabla_{\boldsymbol{\theta}} J = \int_{\mathcal{S}_\tau} p(\tau) \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) \left[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] d\tau$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T-1} \nabla \pi_{\boldsymbol{\theta}}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$



[2]

This result can be understood as return-weighted maximum-likelihood approach.

[2] Picture shamelessly taken from Sergey Levine's course slides (CS285).

Concluding the Vanilla Policy Gradient Method (REINFORCE)

Monte Carlo Policy Gradient Estimate

$$\nabla_{\theta} \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T-1} \nabla \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \underbrace{\left(\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)}_{\text{Monte Carlo target}}$$

Pros:

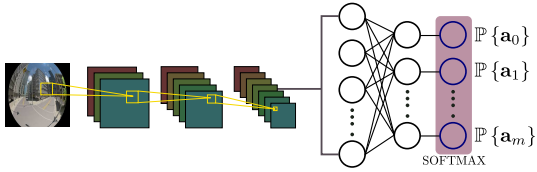
- unbiased
- easy to implement
- only one neural network required
- works with discrete and continuous action spaces

Cons:

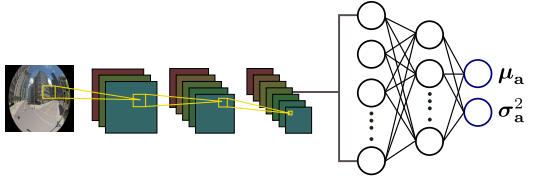
- extremely low sample efficiency $\Rightarrow$ motivates TRPO and PPO
- slow convergence
- noisy (Monte Carlo method)

Policy Networks

- Discrete case: neural net outputs probabilities



- Continuous case: neural net outputs first and second order statistics of model distribution (e.g. Gaussian)



A Brief Teaser on Actor Critic Methods

Can we use other targets to reduce the noise?

$$\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \Rightarrow \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{s}_t, \mathbf{a}_t \right] = Q_\pi(\mathbf{s}_t, \mathbf{a}_t)$$

Can we do even better?

$$Q_\pi(\mathbf{s}_t, \mathbf{a}_t) \Rightarrow Q_\pi(\mathbf{s}_t, \mathbf{a}_t) - \underbrace{\mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{s}_t \right]}_{= V_\pi(\mathbf{s}_t)} = A_\pi(\mathbf{s}_t, \mathbf{a}_t)$$

Implementation trick: Use TD-error: $A_\pi(\mathbf{s}_t, \mathbf{a}_t) \approx \delta_\pi = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V_\pi(\mathbf{s}_{t+1})$ as $Q_\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_\pi [\delta_\pi]$

$$\nabla_{\boldsymbol{\theta}} J_{\mathrm{AC},t} \approx \nabla \ln \pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{s}_t) \left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V_{\pi,\boldsymbol{\phi}}(\mathbf{s}_{t+1}) - V_{\pi,\boldsymbol{\phi}}(\mathbf{s}_t) \right]$$

$\Longrightarrow$ Advantage-Actor-Critic with policy network parameters $\boldsymbol{\theta}$ and value network parameters $\boldsymbol{\phi}$.

A Brief Excursion Into Optimal Control

Some common conventions ...

- Discrete-time states: $\mathbf{s}_t$
  $\Rightarrow$ continuous-time states: $\mathbf{x}(t)$
- Discrete-time actions: $\mathbf{a}_t$
  $\Rightarrow$ continuous-time inputs: $\mathbf{u}(t)$
- Stochastic policy: $p(\mathbf{a}_t|\mathbf{s}_t)$
  $\Rightarrow$ deterministic controller: $\mathbf{u}(t) = \mathbf{g}(\mathbf{x}(t), t)$
- Stochastic state transition: $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
  $\Rightarrow$ non-linear ode: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$
- Stochastic reward: $r(\mathbf{s}_t, \mathbf{a}_t)$
  $\Rightarrow$ deterministic cost: $\underbrace{C(\mathbf{x}(t), \mathbf{u}(t), t)}_{\text{running cost}}, \underbrace{D(\mathbf{x}(T))}_{\text{terminal cost}}$

Derivation of the Hamilton-Jacobi-Bellman Equation

Objective: Find a optimal controller $u(t)$ such the cost functional $\int_0^T C(\mathbf{x}(t), \mathbf{u}(t), t)dt + D(\mathbf{x}(T))$ is minimised $\Rightarrow$ we need do define a value function $V(\mathbf{x}(t), t)$.

$$V(\mathbf{x}(t), t) = \min_{\mathbf{u}(.)} \left[\int_t^T C\left(\mathbf{x}(s), \mathbf{u}(s), s\right) ds + D\left(\mathbf{x}(T)\right) \right],$$

according to Bellman's principle of optimality, the optimal cost starting from $\mathbf{x}$ at t and a applying the optimal controller $\mathbf{u}^*(s)$ for $s \in [t, t + \Delta t]$ can be written as:

$$V(\mathbf{x}(t), t) = \min_{\mathbf{u}(.)} \left[\int_t^{t+\Delta t} C\left(\mathbf{x}(s), \mathbf{u}(s), s\right) ds + D\left(\mathbf{x}(T)\right) + V(\mathbf{x}(t + \Delta t), t + \Delta t) \right]$$

Derivation of the Hamilton-Jacobi-Bellman Equation

To get rid of the term $V(\mathbf{x}(t+\Delta t), t+\Delta t)$ we use Taylor expansion around t:

$$V(\mathbf{x}(t+\Delta t), t+\Delta t) = V(\mathbf{x}(t), t) + \frac{\partial V}{\partial \mathbf{x}}^T \dot{\mathbf{x}}\Big|_{\mathbf{x}(t),t} \Delta t + \frac{\partial V}{\partial t}\Big|_{\mathbf{x}(t),t} \Delta t + o(\Delta t)$$

We plug this back into the equation before, subtract $V(\mathbf{x}(t), t)$ from both sides and ignore the higher order terms $o(\Delta t)$. Dividing by Δt reveals the **Hamilton–Jacobi–Bellman equation** in general form.

$$0 = \min_{\mathbf{u}(.)} \left[\frac{1}{\Delta t} \int_t^{t+\Delta t} C\left(\mathbf{x}(s), \mathbf{u}(s), s\right) ds + D\left(\mathbf{x}(T)\right) + \frac{\partial V}{\partial \mathbf{x}}^T \dot{\mathbf{x}} + \frac{\partial V}{\partial t} \right]$$

Simplification: Δt is usually pretty small (sample rates for digital controllers are usually between 10^{-6}s and 10^{-3}s. $\Rightarrow \int_t^{t+\Delta t} C\left(\mathbf{x}(s), \mathbf{u}(s), s\right) ds \approx C\left(\mathbf{x}(s), \mathbf{u}(s), s\right) \Delta t$. Hence, the HJB simplifies to

$$0 = \min_{\mathbf{u}(.)} \left[C\left(\mathbf{x}(t), \mathbf{u}(t), t\right) + D\left(\mathbf{x}(T)\right) + \frac{\partial V}{\partial \mathbf{x}}^T \dot{\mathbf{x}} + \frac{\partial V}{\partial t} \right],$$

which is still a very hard nut to crack.

Infinite Horizon Linear Quadratic Regulators (LQR)

Definitions:

- Linear time-invariant dynamics: $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$
- Linear controller: $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$
- Quadratic running cost: $C(\mathbf{x}(t), \mathbf{u}(t), t) = \mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)$, $\mathbf{Q} \succ 0$, $\mathbf{R} \succ 0$
- Terminal cost: $D(\mathbf{x}(T)) = 0$

For the infinite horizon case it follows that $V(\mathbf{x}) = \mathbf{x}^T\mathbf{P}\mathbf{x}$, where $\mathbf{P} \succ 0$.[3] Given the definitions above the HJB for the infinite horizon LQR is given by

$$0 = \min_{\mathbf{u}(.)} \left[\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u} + \underbrace{\frac{\partial V}{\partial \mathbf{x}}^T \dot{\mathbf{x}}}_{=2\mathbf{x}^T\mathbf{P}(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{u})} + \underbrace{\frac{\partial V}{\partial t}}_{=0} \right] = \min_{\mathbf{u}(.)} \left[\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u} + 2\mathbf{x}^T\mathbf{P}\mathbf{A}\mathbf{x} + 2\mathbf{x}^T\mathbf{P}\mathbf{B}\mathbf{u} \right].$$

$$\frac{\partial}{\partial \mathbf{u}} \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u} + 2\mathbf{x}^T\mathbf{P}\mathbf{A}\mathbf{x} + 2\mathbf{u}^T\mathbf{P}\mathbf{B}\mathbf{x} = 2\mathbf{R}\mathbf{u} + 2\mathbf{B}\mathbf{P}\mathbf{x} \stackrel{!}{=} \mathbf{0} \Rightarrow \mathbf{u} = -\underbrace{\mathbf{R}^{-1}\mathbf{B}\mathbf{P}}_{=\mathbf{K}}\mathbf{x}$$

[3] Proof: Plug $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$ in and perform a Lyapunov analysis on the closed loop system.

Infinite Horizon Linear Quadratic Regulators (LQR)

Plugging the expression for the optimal controller back in the HJB equation yields the necessary condition given the optimal controller.

$$0 = \mathbf{x}^T \left(\mathbf{Q} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + 2\mathbf{P}\mathbf{A} \right) \mathbf{x}$$

Since this condition must hold for all $\mathbf{x}$ it is sufficient to consider only the matrix equation

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

where we split $2\mathbf{P}\mathbf{A}$ into $\mathbf{P}\mathbf{A} + \mathbf{A}^T\mathbf{P}$ for symmetry reasons.

$\Rightarrow$ Solving this equation (**algebraic Riccati equation**) for $\mathbf{P}$ reveals the optimal controller matrix $\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}\mathbf{P}$.

Finite Horizon Linear Quadratic Regulators (LQR)

Some minor modifications:

- Terminal cost: $D(\mathbf{x}(T)) = \mathbf{x}(T)^T \mathbf{Q}_T \mathbf{x}(T)$
- Explicit time dependence: $V(\mathbf{x}, t) = \mathbf{x}^T \mathbf{P}(t) \mathbf{x}$, where $\mathbf{P}(t) \prec 0 \, \forall \, t$

We derive as similar expression for the HJB with a minor modification[4] .

$$
\begin{aligned}
0 &= \min_{\mathbf{u}(.)} \left[\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} + \underbrace{\frac{\partial V}{\partial \mathbf{x}}^T \dot{\mathbf{x}}}_{=2\mathbf{x}^T \mathbf{P}(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{u})} + \underbrace{\frac{\partial V}{\partial t}}_{=\mathbf{x}^T \dot{\mathbf{P}} \mathbf{x}} \right] \\
&= \min_{\mathbf{u}(.)} \left[\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} + 2\mathbf{x}^T \mathbf{P} \mathbf{A} \mathbf{x} + 2\mathbf{x}^T \mathbf{P} \mathbf{B} \mathbf{u} + \mathbf{x}^T \dot{\mathbf{P}} \mathbf{x} \right].
\end{aligned}
$$

[4] $D(\mathbf{x}(T))$ covered by $V(\mathbf{x}(t + \Delta t), t + \Delta t)$

Finite Horizon Linear Quadratic Regulators (LQR)

Since we do not alter the HJB w.r.t. control inputs the expression of the optimal controller stays the same $\mathbf{u}(t) = \mathbf{R}^{-1}\mathbf{B}\mathbf{P}(t)\mathbf{x} = \mathbf{K}(t)\mathbf{x}$, apart from the fact the controller matrix is now a function of time. Plugging the expression for the optimal controller back in the HJB equation yields the necessary condition given the optimal controller.

$$0 = \mathbf{x}^T \left(\mathbf{Q} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{A}^T\mathbf{P} + \dot{\mathbf{P}} \right) \mathbf{x}$$

Since this condition must hold for all $\mathbf{x}$ it is sufficient to consider only the matrix differential equation

$$\begin{aligned} \dot{\mathbf{P}} &= -\mathbf{Q} + \mathbf{P}(t)\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}(t) - \mathbf{P}(t)\mathbf{A} - \mathbf{A}^T\mathbf{P}(t) \\ \mathbf{P}(T) &= \mathbf{Q}_f, \end{aligned}$$

which corresponds to a **differential Riccati equation**.