Reinforcement Learning and Optimal Control

IFT6760C, Fall 2021

Pierre-Luc Bacon

November 17, 2021

Linear Quadratic Regulation

Assumption: cost function is quadratic and the dynamics are linear.

$$\begin{aligned}
&\text{minimize} \quad x_T Q_T x_T + \sum_{t=1}^{T} \left(x_t^\top Q x_t + u_t^\top R u_t \right) \\
&\text{subject to} \quad x_{t+1} = A x_t + B u_t, \quad t = 1, \ldots T-1 \\
&\text{given} \quad x_1 \quad .
\end{aligned}$$

This is a finite-horizon **deterministic** MDP over continuous sets of states and actions.

Backward Induction

The backward induction algorithm (value iteration in finite-horizon mdps) starts at the end, setting $v_T^\star(s_t) = r(s_T)$. Here, we have:

$$J_T(x_T) = x_T Q_T x_T \quad .$$

By induction, we also have that:

$$J_t(x_t) = \min_u \left\{ x_t^\top Q x_t + u^\top R u + J_{t+1}(Ax_t + Bu) \right\} \quad ,$$

which must be quadratic. We solve the minimization problem in closed-form: take the gradient, set to zero.

$$\begin{aligned} f_t(u) &\triangleq \left\{ x_t^\top Q x_t + u^\top R u + (Ax_t + Bu)^\top P_{t+1}(Ax_t + Bu) \right\} \\ Df_t(u) &= 2u^\top R + 2(Ax_t + Bu)^\top P_{t+1} B = 0 \\ \Rightarrow u_t^\star &= -(R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A x_t \end{aligned}$$

Exact optimal cost-to-go function

If we substitute $u_t^\star$ into $J_t(x_t)$, we get:

$$\begin{aligned} J_t(x_t) &= \min_u \left\{ x_t^\top Q x_t + u^\top R u + J_{t+1}(Ax_t + Bu) \right\} \\ &= x_t^\top Q x_t + (u_t^\star)^\top R u_t^\star + J_{t+1}(Ax_t + Bu_t^\star) \quad . \end{aligned}$$

After some simplification, we get:

$$\begin{aligned} J_t(x_t) &= x_t P_t x_t \\ P_t &= Q + A^\top P_{t+1} A - A^\top P_{t+1} B (R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A \end{aligned}$$

Backward Induction for LQR: Ricatti Equation

We get the algorithm:

- Set $P_T = Q_T$
- From $t = T - 1, \ldots, 1$:
  - Set $P_t = Q + A^\top P_{t+1} A - A^\top P_{t+1} B (R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A$
  - Set $K_t = -(R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A$

You can then compute the optimal control at time t in state x_t with $u_t^\star = K_t x_t$ (the optimal controls are linear in the states).

Continuous-Time Control

Types of Problems

Mayer:

$$\begin{aligned} &\text{minimize} && c(x(t_f)) \\ &\text{subject to} && \dot{x}(t) = f(x(t), u(t)) \\ &\text{given} && x(t_0) = x_0 \quad . \end{aligned}$$

Lagrange:

$$\begin{aligned} &\text{minimize} && \int_{t_0}^{t_f} c(x(t), u(t))dt \\ &\text{subject to} && \dot{x}(t) = f(x(t), u(t)) \\ &\text{given} && x(t_0) = x_0 \quad . \end{aligned}$$

Bolza

$$\text{minimize} \quad c(x(t_f)) + \int_{t_0}^{t_f} c(x(t), u(t))dt$$

$$\text{subject to} \quad \dot{x}(t) = f(x(t), u(t))$$

$$\text{given } x(t_0) = x_0 \quad .$$

We now want to find a control function $u(t)$ (not just a sequence) that depends on the time t (which is a real).

The control function $u(t)$ depends only on the time, and not the state, hence we are still in the world of open-loop control (not closed-loop).

Ordinary Differential Equation

In the disrete-time case, we had a *difference equation*:

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

Now instead of explicitely specifying the *next state*, we specify how the state changes through time, aka:

$$\dot{x}(t) \triangleq Dx(t) = f(x(t), u(t)) \quad .$$

Initial Value Problem (IVP)

$$
\begin{aligned}
&\text{find} \quad x(t_f) \\
&\text{subject to} \quad \dot{x}(t) = f(x(t), t) \\
&\text{given} \quad x(t_0) = x_0 \quad .
\end{aligned}
$$

Given an ODE and initial state at time t_0, find the value of the state at time t_f.

Quadrature

If we want to know what would be the value of the state variable at $t_{i+1} \triangleq t_i + h_i$, with h_i being the *integration step size* for the i-th step, then we have to compute the integral. By the fundamental theorem of calculus:

$$x(t_{i+1}) = x(t_i) + \int_{t_i}^{t_{i+1}} f(x(t), t)dt \quad .$$

Using a quadrature rule, we can approximate this integral by dividing the step h_i into K subintervals:

$$\int_{t_i}^{t_{i+1}} f(x(t), t)dt \approx h_i \sum_{j=1}^{N} \beta_j f(x(\tau_{ij}), \tau_{ij})$$

Quadrature: A way to take a weighted sum of points in order to compute the *area*

Quadrature

$$\int_{t_i}^{t_{i+1}} f(x(t), t)dt \approx h_i \sum_{j=1}^{N} \beta_j f(x(\tau_{ij}), \tau_{ij})$$

The coefficients β_j are called *weights* while we refer to ρ_j as *nodes* and must obey the ordering $0 \leq \rho_1 \leq \ldots \leq \rho_k \leq 1$ (for example j/K)

Quadrature within a Quadrature

While the value of $x(t_i)$ is given to us, we still have the problem of knowing what the state would be for each of the N points within that subinterval. If we knew these points, we would compute:

$$\tilde{x}_{t_{i+1}} = \tilde{x}_{t_i} + h_i \sum_{j=1}^{N} \beta_j f(x(\tau_{ij}), \tau_{ij}) \quad .$$

The idea behind the so-called *Runge-Kutta* methods is to set up an auxiliary quadrature problem to obtain the value of those points.

“Bootstrapping”

In *explicit* Runge-Kutta methods, we “bootstrap” our inner quadrature formula from the values computed before:

$$k_{ij} = f\left(\left(\tilde{x}_{t_i} + h_i \sum_{l=1}^{j-1} \alpha_{il} k_{il}\right),\ t_i + \rho_i h_i\right) \quad .$$

This recursive expression is then used within the “outer” quadrature formula and yields:

$$\tilde{x}_{t_{i+1}} = \tilde{x}_{t_i} + h_i \sum_{j=1}^{K} \beta_j k_{ij} \quad .$$

Special cases

If we choose $K = 1$ and set $\beta_j = 1$, we get the *Euler method*:

$$\tilde{x}_{t_{i+1}} = \tilde{x}_{t_i} + h_i k_i$$

The classical Runge-Kutta method with $K = 4$ is given by the updates:

$$\begin{aligned}
k_{i,1} &= h_i f(\tilde{x}_i,\ t_i) \\
k_{i,2} &= h_i f(\tilde{x}_i + \frac{1}{2} k_{i,1},\ t_i + \frac{h_i}{2}) \\
k_{i,3} &= h_i f(\tilde{x}_i + \frac{1}{2} k_{i,2},\ t_i + \frac{h_i}{2}) \\
k_{i,4} &= h_i f(\tilde{x}_i + k_{i,3},\ t_i + h_i) \\
\tilde{x}_{i+1} &= \tilde{x}_i + \frac{1}{6}\left(k_{i,1} + 2k_{i,2} + 2k_{i,3} + k_{i,4}\right) \quad .
\end{aligned}$$

Collocation

Rather than viewing Runge-Kutta methods as a nested quadrature rule, it is also possible to obtain the same results by considering the problem of approximating the continuous function $x(t)$ by a polynomial of order K. This idea will be studied in the context of the so-called “collocation” methods.

Boundary Value Problem

Let's consider a generalization of IVPs, en route to solving "full" optimal control problems. A specific kind of BVP is a Two-point boundary value problem (TPBVP):

$$\dot{x}(t) = f(x(t), t), \quad a < t < b$$
$$\psi(x(a), x(b)) = 0 \quad .$$

Example

Consider the BVP where the dynamics given by

$$\dot{x} = x$$

Goal: find the initial state $x(a)$ such that terminal state is s equal to β.

The boundary condition is: $\psi(x(a), x(b)) \triangleq x(b) - \beta$.

Example: Closed-Form IVP

$$x(t) = x(a)\exp(t-a) \quad .$$

Because we are trying to find the initial value which leads to a trajectory satisfying the boundary condition, we view the initial value as an optimization variable x_a and define a constraint function of the form:

$$h(x_a) \triangleq x_a \exp(b-a) - \beta \quad ,$$

This leads us to the idea of single-shooting methods (or sequential methods): simulate (integrate), observe terminal state, evaluate cost, "backprop" (if doing gradient-based optimization).

Figure

Forward Sensitivity Equation

Let's consider a parameterized ODE of the form:

$$\dot{x} = f(x(t), t, \theta), \quad x(a) = x_a(\theta) \quad ,$$

Taking the total derivative:

$$\frac{\partial^2 x(t)}{\partial \theta \partial t} = \frac{\partial f(x(t), t, \theta)}{\partial x} \frac{\partial x(t)}{\partial \theta} + \frac{\partial f(x(t), t, \theta)}{\partial \theta} \quad .$$

Using the symmetry of second derivatives [1], we can write instead:

$$\frac{\partial^2 x(t)}{\partial t \partial \theta} = \frac{\partial f(x(t), t, \theta)}{\partial x} \frac{\partial x(t)}{\partial \theta} + \frac{\partial f(x(t), t, \theta)}{\partial \theta} \quad .$$

Forward Sensitivity Equation

If we define $s \triangleq \frac{\partial x(t)}{\partial \theta}$ ("s" as in *sensitivity*), we get the *forward sensitivity equations*:

$$\dot{s} = \frac{\partial f(x(t), t, \theta)}{\partial x} s(t) + \frac{\partial f(x(t), t, \theta)}{\partial \theta}, \quad s(a) = \frac{\partial x(a)}{\partial \theta}.$$

We can easily evaluate the Jacobian of f by augmenting the dynamics system f with the corresponding sensitivity equation. We can then solve for the original IVP simultaneously with its sensitivities.