Stochastic Dynamic Programming

V. Leclère (ENPC)

October 8, 2015

Contents

Contents

Stochastic Controlled Dynamic System

A stochastic controlled dynamic system is defined by its *dynamic*

$$\boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_{t+1})$$

and initial state

$$\boldsymbol{x}_0 = x_0$$

The variables

- $\boldsymbol{x}_t$ is the *state* of the system,
- $\boldsymbol{u}_t$ is the *control* applied to the system at time t,
- $\boldsymbol{w}_t$ is an exogeneous noise.

Examples

- Stock of water in a dam:
  - $\boldsymbol{x}_t$ is the amount of water in the dam at time t,
  - $\boldsymbol{u}_t$ is the amount of water turbined at time t,
  - $\boldsymbol{w}_t$ is the inflow of water at time t.
- Boat in the ocean:
  - $\boldsymbol{x}_t$ is the position of the boat at time t,
  - $\boldsymbol{u}_t$ is the direction and speed chosen at time t,
  - $\boldsymbol{w}_t$ is the wind and current at time t.
- Subway network:
  - $\boldsymbol{x}_t$ is the position and speed of each train at time t,
  - $\boldsymbol{u}_t$ is the acceleration chosen at time t,
  - $\boldsymbol{w}_t$ is the delay due to passengers and incident on the network at time t.

Optimization Problem

We want to solve the following optimization problem

$$\min \quad \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_{t+1}\big) + K\big(\boldsymbol{x}_T\big)\Big] \tag{1a}$$

$$s.t. \quad \boldsymbol{x}_{t+1} = f_t\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_{t+1}\big), \qquad \boldsymbol{x}_0 = x_0 \tag{1b}$$

$$\boldsymbol{u}_t \in U_t(\boldsymbol{x}_t) \tag{1c}$$

$$\sigma(\boldsymbol{u}_t) \subset \sigma\big(\boldsymbol{w}_0, \cdots, \boldsymbol{w}_t\big) \tag{1d}$$

Dynamic Programming Principle

Assume that the noises $\boldsymbol{w}_t$ are independent and exogeneous.

Then, there exists an optimal solution, called a strategy, of the form $\boldsymbol{u}_t = \pi_t(\boldsymbol{x}_t)$, given by

$$\pi_t(x) = \arg\min_{u \in U_t(x)} \mathbb{E}\Big[\underbrace{L_t(x, u, \boldsymbol{w}_{t+1})}_{\text{current cost}} + \underbrace{V_{t+1} \circ f_t(x, u, \boldsymbol{w}_{t+1})}_{\text{future costs}}\Big] ,$$

where (Dynamic Programming Equation)

$$\begin{cases} V_T(x) & = K(x) \\ V_t(x) & = \min_{u \in U_t(x)} \mathbb{E}\Big[L_t(x, u, \boldsymbol{w}_{t+1}) + V_{t+1} \circ \underbrace{f_t(x, u, \boldsymbol{w}_{t+1})}_{\text{"}\boldsymbol{X}_{t+1}\text{"}}\Big] \end{cases}$$

Dynamic Programming Principle

Assume that the noises $\boldsymbol{w}_t$ are independent and exogeneous.

Then, there exists an optimal solution, called a strategy, of the form $\boldsymbol{u}_t = \pi_t\big(\boldsymbol{x}_t\big)$, given by

$$\pi_t(x) = \arg\min_{u \in U_t(x)} \mathbb{E}\Big[\underbrace{L_t(x, u, \boldsymbol{w}_{t+1})}_{\text{current cost}} + \underbrace{V_{t+1} \circ f_t\big(x, u, \boldsymbol{w}_{t+1}\big)}_{\text{future costs}}\Big] ,$$

where (Dynamic Programming Equation)

$$\begin{cases} V_T(x) & = K(x) \\ V_t(x) & = \min_{u \in U_t(x)} \mathbb{E}\Big[L_t(x, u, \boldsymbol{w}_{t+1}) + V_{t+1} \circ \underbrace{f_t\big(x, u, \boldsymbol{w}_{t+1}\big)}_{\text{"}\boldsymbol{X}_{t+1}\text{"}}\Big] \end{cases}$$

Interpretation of Bellman Value

The Bellman's value function $V_{t_0}(x)$ can be interpreted as the value of the problem starting at time t_0 from the state x. More precisely we have

$$V_{t_0}(x) = \min \quad \mathbb{E}\Big[\sum_{t=t_0}^{T-1} L_t\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_{t+1}\big) + K\big(\boldsymbol{x}_T\big)\Big] \tag{2a}$$

$$s.t. \quad \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_{t+1}), \qquad \boldsymbol{x}_{t_0} = x \tag{2b}$$

$$\boldsymbol{u}_t \in U_t(\boldsymbol{x}_t) \tag{2c}$$

$$\sigma(\boldsymbol{u}_t) \subset \sigma\big(\boldsymbol{w}_0, \cdots, \boldsymbol{w}_t\big) \tag{2d}$$

Information structures in the multistage setting

Open-Loop Every decision $(u_t)_{t\in[\![0,T-1]\!]}$ is taken before any noises $(\boldsymbol{\xi}_t)_{t\in[\![0,T-1]\!]}$ is known. We decide a planning, and stick to it.

Decision Hazard Decision $\boldsymbol{u}_t$ is taken knowing all past noises $\xi_0,\dots,\xi_t$, but not knowing $\boldsymbol{\xi}_{t+1},\dots,\boldsymbol{\xi}_T$.

Hazard Decision Decision $\boldsymbol{u}_t$ is taken knowing all past noises $\xi_0,\dots,\xi_t$, and the next noise $\boldsymbol{\xi}_{t+1}$ but not knowing $\boldsymbol{\xi}_{t+2},\dots,\boldsymbol{\xi}_T$.

Anticipative Every decision $(\boldsymbol{u}_t)_{t\in[\![0,T-1]\!]}$ is taken knowing the whole scenario $(\boldsymbol{\xi}_t)_{t\in[\![0,T-1]\!]}$. There is one deterministic optimization problem by scenario.

With the same objective function this gives better and better value as the solution use more and more information.

Information structures: comments

Open-Loop This case can happen in practice (e.g. fixed planning). There are specific methods to solve this type of optimization problem (e.g. stochastic gradient methods).

Decision Hazard The decision $\boldsymbol{u}_t$ is taken at the beginning of period $[t, t+1[$. The decision is always implementable, and might be conservative as it doesnot leverage any prediction over the noise in $[t, t+1[$.

Hazard Decision The decision $\boldsymbol{u}_t$ is taken at the end of period $[t, t+1[$. The modelization is optimistic as it assumes perfect knowledge that might not be available in practice.

Anticipative This problem is never realistic. However it is a lower bound of the real problem that can be estimated through Monte-Carlo and deterministic optimization.

Independence of noise

- The Dynamic Programming equation requires only the time-independence of noises.
- This can be relaxed if we consider an extended state.
- Consider a dynamic system driven by an equation

$$\boldsymbol{y}_{t+1} = f_t(\boldsymbol{X}_t, \boldsymbol{u}_t, \varepsilon_{t+1})$$

where the random noise ε_t is an AR1 process :

$$\varepsilon_t = \alpha_t \varepsilon_{t-1} + \beta_t + \boldsymbol{w}_t,$$

$\{\boldsymbol{w}_t\}_{t \in \mathbb{Z}}$ being independent.

- Then $\boldsymbol{y}_t$ is called the physical state of the system and DP can be used with the information state $\boldsymbol{X}_t = (\boldsymbol{y}_t, \varepsilon_{t-1})$.
- Generically speaking, if the noise $\boldsymbol{w}_t$ is exogeneous (not affected by decisions $\boldsymbol{u}_t$), then we can always apply Dynamic Programming with the state

$$(\boldsymbol{x}_t, \boldsymbol{w}_1, \dots, \boldsymbol{w}_t)$$

Contents

Dynamic Programming Algorithm

Data: Problem parameters
Result: optimal control and value;

$V_T \equiv K$;

for $t : T \to 0$ **do**

  for $x \in \mathbb{X}_t$ **do**

    $V_t(x) = \infty$;

    for $u \in U_t(x)$ **do**

      $v_u = \mathbb{E}\Big[L_t(x, u, \boldsymbol{w}_{t+1}) + V_{t+1} \circ f_t\big(x, u, \boldsymbol{w}_{t+1}\big)\Big]$;

      if $v_u < \underline{v}$ **then**

        $V_t(x) = v_u$;

        $\pi_t(x) = u$;

Algorithm 1: Dynamic Programming Algorithm (discrete case)

Number of flops: $O(T \times |\mathbb{X}_t| \times |\mathbb{U}_t| \times |\mathbb{W}_t|)$.

3 curses of dimensionality

1. State. If we consider 3 independent states each taking 10 values, then $|\mathbb{X}_t| = 10^3 = 1000$. In practice DP is not applicable for states of dimension more than 5.
2. Decision. The decision are often vector decisions, that is a number of independent decision, hence leading to huge $|U_t(x)|$.
3. Expectation. In practice random information came from large data set. Without a proper statistical treatment computing an expectation is costly. Monte-Carlo approach are costly too, and unprecise.

Numerical considerations

- The DP equation holds in (almost) any case.
- The algorithm shown before compute a look-up table of control for every possible state *offline*. It is impossible to do if the state is (partly) continuous.
- Alternatively, we can focus on computing *offline* an approximation of the value function V_t and derive the optimal control *online* by solving a one-step problem, solved only at the current state :

$$\pi_t(x) \in \arg\min_{u \in U_t(x)} \mathbb{E}\Big[L_t(x, u, \boldsymbol{w}_{t+1}) + V_{t+1} \circ f_t\big(x, u, \boldsymbol{w}_{t+1}\big)\Big]$$

- The field of Approximate DP gives methods for computing those approximate value function (decomposed on a base of functions).
- The simpler one consisting in discretizing the state, and then interpolating the value function.

DP on a Markov Chain

- Sometimes it is easier to represent a problem as a controlled Markov Chain
- Dynamic Programming equation can be computed directly, without expliciting the control.
- Let's work out an example...

Contents

The Linear-Quadratic setting

We assume a linear dynamic

$$\boldsymbol{x}_{t+1} = A_t \boldsymbol{x}_t + B_t \boldsymbol{u}_t + \boldsymbol{W}_{t+1}$$

associated with a quadratic cost

$$\mathbb{E}\Big[\sum_{t=0}^{T-1} \Big(\boldsymbol{X}_t' Q_t \boldsymbol{x}_t + \boldsymbol{u}_t' R_t \boldsymbol{u}_t\Big)\Big] + \boldsymbol{X}_T' Q_T \boldsymbol{x}_T.$$

A few more assumptions

- $\boldsymbol{x}_t$ is of dimension n, $\boldsymbol{u}_t$ of dimension m.
- Q_t is a symmetric semidefinite positive matrix, and R_t symmetric definite positive.
- $\boldsymbol{w}_t$ is a centered (i.e. of mean 0) independent, exogeneous noise (i.e their law does not depend of the state or control), with finite second order moment.
- The controls are non-anticipative.

Solving the LQ case I

The DP equation read

$$\begin{cases} V_T(x) &= x' Q_T x \\ V_t(x) &= \min_u \mathbb{E}\Big[x' Q_t x + u' R_t u + V_{t+1}\big(A_t x + B_t u + \boldsymbol{W}_{t+1}\big)\Big] \end{cases}$$

Leading to

$$V_t(x_t) = x_t' K_t x_t + \sum_{\tau=t}^{T-1} \mathbb{E}\left[\boldsymbol{w}'_{t+1} K_{t+1} \boldsymbol{w}_{t+1}\right]$$

and

$$\boldsymbol{u}_t^\sharp = \pi_t^\sharp(\boldsymbol{x}_t) = L_t \boldsymbol{x}_t \ .$$

Solving the LQ case I

The DP equation read

$$\begin{cases} V_T(x) & = x' Q_T x \\ V_t(x) & = \min_u \mathbb{E}\Big[x' Q_t x + u' R_t u + V_{t+1}\big(A_t x + B_t u + \boldsymbol{W}_{t+1} \big) \Big] \end{cases}$$

Leading to

$$V_t(x_t) = x_t' K_t x_t + \sum_{\tau=t}^{T-1} \mathbb{E}\big[\boldsymbol{w}'_{t+1} K_{t+1} \boldsymbol{w}_{t+1} \big]$$

and

$$\boldsymbol{u}_t^\sharp = \pi_t^\sharp(\boldsymbol{x}_t) = L_t \boldsymbol{x}_t \ .$$

Solving the LQ case II

We have

$$V_t(x_t) = x_t' K_t x_t + \sum_{\tau=t}^{T-1} \mathbb{E}\left[\boldsymbol{w}'_{t+1} K_{t+1} \boldsymbol{w}_{t+1}\right]$$

and

$$\boldsymbol{u}_t^\sharp = \pi_t^\sharp(\boldsymbol{x}_t) = L_t \boldsymbol{x}_t \ .$$

Where

$$L_t = -\left(B_t' K_{t+1} B_t + R_t\right)^{-1} B_t' K_{t+1} A_t \ ,$$

and

$$\begin{cases} K_T & = Q_T \\ K_t & = A_t' \Big(K_{t+1} - K_{t+1} B_t \big(B_t' K_{t+1} B_t + R_t \big)^{-1} B_t' K_{t+1} \Big) A_t + Q_t \end{cases}$$

Contents

Introducing the Bellman operators

We define the Bellman operator associated to our optimisation problem

$$T_t(J) : x \mapsto \min_{u \in U_t(x)} \mathbb{E}\Big[L_t(x, u, \boldsymbol{w}_{t+1}) + J \circ f_t\big(x, u, \boldsymbol{w}_{t+1}\big)\Big] .$$

The Dynamic Programming equation can then be written

$$\begin{cases} V_T &= K \\ V_t &= T_t\Big(V_{t+1}\Big) \end{cases}$$

We also construct the *policy-dependent Bellman operator*

$$T_t^{\pi}(J) : x \mapsto \mathbb{E}\Big[L_t(x, \pi(x), \boldsymbol{w}_{t+1}) + J \circ f_t\big(x, \pi(x), \boldsymbol{w}_{t+1}\big)\Big] .$$

Discounted fixed cost case

We now consider the following specific case problem, where $\big(\boldsymbol{w}_t\big)_{t\in\mathbb{N}}$ is i.i.d.

$$\begin{aligned} \min \quad & \mathbb{E}\Big[\sum_{t=0}^{T} \alpha^t L\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_{t+1}\big)\Big] && (3)\\ s.t. \quad & \boldsymbol{x}_{t+1} = f\big(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_{t+1}\big), \qquad \boldsymbol{x}_0 = x_0 && (4)\\ & \boldsymbol{u}_t \in U(\boldsymbol{x}_t) && (5)\\ & \sigma\big(\boldsymbol{u}_t\big) \subset \sigma\big(\boldsymbol{w}_0, \cdots, \boldsymbol{w}_t\big) && (6) \end{aligned}$$

where $\alpha \in]0,1]$. Note that the constraint and cost structure doesnot depend on t.

The Bellman operator is given by

$$T(J) : x \mapsto \min_{u\in U(x)} \mathbb{E}\Big[L(x, u, \boldsymbol{w}_{t+1}) + \alpha J \circ f\big(x, u, \boldsymbol{w}_{t+1}\big)\Big]$$

Infinite horizon problems

There is different ways of considering the above problem in an "infinite horizon" setting.

1. Discounted case. This is the case where $\alpha < 1$. It is especially easy to treat if the cost L is bounded.
2. Stochastic shortest path. In this case $\alpha = 1$ but there is a "cemetary state" such that once reached the system remains there with null cost. Moreover, we assume that the system always reach the cemetary state in a finite time.
3. Average cost per stage problems. This approach is mainly taken if the infinite time cost isn't finite (for example $\alpha = 1$ and $L > 0$). We consider

$$\lim_{T\to\infty} \frac{1}{T} \quad \mathbb{E}\Big[\sum_{t=0}^{T-1} L(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{w}_{t+1})\Big] .$$

An overview of typical infinite horizon results

Here are the main results that can be shown in infinite horizon problems (under the right set of assumptions)

1. the sequence of value function $V_{n+1} = T\big(V_n\big)$, converges toward the value function of the infinite horizon problem: $\lim_{n\to\infty} V_n = V^\sharp$.
2. The optimal value of the infinite horizon problem is a fixed point of the Bellman operator: $V^\sharp = T\big(V^\sharp\big)$.
3. If π is such that $V^\sharp = T^\pi V^\sharp$ then the stationnary policy π is optimal.

Value iteration algorithm

Data: Initial value $V^{(0)}$
Result: optimal policy and value;
repeat
  for $x \in \mathbb{X}$ **do**

$$V^{(k+1)}(x) = T\big(V^{(k)}\big)(x)$$

until $\|V^{(k+1)} - V^{(k)}\|_\infty < \varepsilon$;

Algorithm 2: Value iteration algorithm

- Each step takes $O(|\mathbb{X}| \times |\mathbb{U}| \times |\Omega|)$ flops.
- The error $|V_n(x) - V^\sharp(x)|$ is bounded by $C\alpha^n$.

Policy iteration algorithm

Data: Initial policy $\pi^{(0)}$
Result: optimal policy and value;
repeat
policy evaluation step: solve $V = T^{\pi^{(k)}}\big(V\big)$ which gives $V^{(k)}$;
policy improvement step :
for $x \in \mathbb{X}$ **do**

$$\pi^{(k+1)}(x) = \operatorname*{arg\,min}_{u \in U(x)} \mathbb{E}\Big[L(x, u, \boldsymbol{w}_{t+1}) + \alpha V^{(k)} \circ f\big(x, u, \boldsymbol{w}_{t+1}\big)\Big]$$

until $V^{(k)} = T\big(V^{(k)}\big)$;

Algorithm 3: Policy iteration algorithm

The policy iteration algorithm terminate in a finite number of step.