نظریه بازیها
Game Theory

ارائه کننده: امیرحسین نیکوفرد
مهندسی برق و کامپیوتر دانشگاه خواجه نصیر

InfiniteDynamic Games

Material

- Dynamic Non-cooperative Game Theory: Second Edition
  - Chapter5: Sections 5:1–5:3.
- An Introductory Course in Non-cooperative Game Theory
  - Chapter 15,16

InfiniteDynamic Games

- Zero sum games
- Non-zero sum games
- Infinite Games
- **Infinite Dynamic Games**
  - Dynamic games in discrete time
  - Information structures
  - Continuous-time differential games
  - **Discrete-time dynamic programming**
  - **Continuous-time dynamic programming**

Dynamic games in discrete time (recap)

One-player discrete-time games:

We start by discussing solution methods for one-player dynamic games, which are simple optimizations. In the context of discrete-time dynamic games, this corresponds to dynamics of the form

$$\underbrace{x_{k+1}}_{\substack{\text{state}\\ \text{at stage } k+1}} = \underbrace{f_k}_{\substack{\text{"dynamics" at}\\ \text{stage } k}} \left(\underbrace{x_k}_{\substack{\text{state}\\ \text{at stage } k}} , \underbrace{u_k}_{\substack{P_1\text{'s action}\\ \text{at stage } k}} \right) \quad , \forall k \in \{1,2,...,K\}, \qquad (1)$$

We assume a finite horizon $(K < \infty)$ stage additive costs of the form

$$J = \sum_{k=1}^{K} g(x_k, u_k) \qquad (2)$$

Open-loop optimization

We can use the following procedure to compute the open-loop policy γ^{OL} that minimizing the cost (2) for the dynamics (1)

1) Compute the cost-to-go using a *backward iteration* starting from $l = K$ and proceeding backward in time until $l = 1$ using

$$V_{K+1}(x) = 0, \qquad V_l(x) = \inf_{u_l \in U_l} (g_l(\mathrm{x}, \mathrm{u}_l) + V_{l+1}(f_l(x, u_l)), \quad \forall \mathrm{x} \in \mathrm{X}$$

- We are assuming that the infimum of $g_l(\mathrm{x}_l^*, \mathrm{u}_l) + V_{l+1}(f_l(\mathrm{x}_l^*, u_l))$ is achieved at some point $u_k \in U_k$. If this is not the case, then this procedure fails. When the infimum is achieved at multiple points, any one can be used in step 2.

Open-loop optimization

2) Compute the sequence of actions

$$u_1^* \in U_1, u_2^* \in U_2, \ldots, u_K^* \in U_K,$$

that minimize $V_i(x_i)$ using a *forward iteration* starting from $k = 1$ and proceeding forward in time until $k = K$:

$$u_k^* = \arg\min_{u_k \in U_k}(g_k(x_k^*, u_k) + V_{k+1}(f_k(x_k^*, u_k))), \quad x_{k+1}^* = f_k(x_k^*, u_k^*), \quad x_1^* = x_1$$

3)Finally, the optimal open-loop policy γ^{OL} is given by

$$\gamma_1^{OL}(x_1) := u_1^*, \quad \gamma_2^{OL}(x_1) := u_2^*, \quad \ldots, \quad \gamma_K^{OL}(x_1) := u_K^*$$

State-feedback optimization

Suppose that one uses the optimal open-loop policy γ^{OL} defined by open loop optimization, which selects the actions

$$u_k = \gamma_k^{OL}(x_1) := u_k^*, \qquad \forall k \in \{1,2,...,\mathrm{K}\}$$

In this case, the pre-computed states x_k^* defined by step 2 in open loop optimization match precisely the states x_k that would be measured during the game. Therefore we would get precisely the same minimum value $\mathrm{V}_1(\mathrm{x}_1)$ for the cost (2), if we were using a state-feedback policy γ^{FB} defined by

$$\gamma_k^{FB}(x_k) = \arg\min_{u_k \in U_k} \underbrace{(g_k(\mathrm{x}_k, \mathrm{u}_k) + V_{k+1}(f_k(x_k, u_k))}_{\text{computed using the measured state } x_k}, \quad \forall k \in \{1,2,...,\mathrm{K}\} \qquad (3)$$

State-feedback optimization

When all the $g_k(x_k, u_k) + V_{k+1}(f_k(x_k, u_k))$ have a minimum for some $u_k \in U_k$, this state-feedback policy γ^{FB} that can do as well as the optimal open-loop policy γ^{OL}. Since it is not possible to obtain a value for (2) lower than $V_1(x_1)$, we conclude that (3) is an optimal state-feedback policy.

Dynamic Programming Example
Affine-Quadratic Problems

- State Equation: $x_{k+1} = A_k x_k + B_k u_k + c_k$
- Cost Function: $\frac{1}{2} \sum_{k=1}^{K} \left(x_{k+1}^\top Q_{k+1} x_{k+1} + u_k^\top R_k u_k \right)$

Proposition

The optimal control policy for the above problem is given by

$$u_k^* = \gamma_k^*(x_k) = -P_k S_{k+1} A_k x_k - P_k (s_{k+1} + S_{k+1} c_k)$$

for all $k \in \mathbf{K}$, *where*

$$P_k = (R_k + B_k^\top S_{k+1} B_k)^{-1} B_k^\top$$
$$S_k = Q_k + A_k^\top S_{k+1} (I - B_k P_k S_{k+1}) A_k \,, \quad S_{K+1} = Q_{K+1}$$
$$s_k = A_k^\top (I - B_k P_k S_{k+1})^\top (s_{k+1} + S_{k+1} c_k) \,, \quad s_{k+1} = 0$$

Computational complexity

- For large state-spaces χ , the computational effort needed to compute the cost-to-go at all stages can be very large. One may then wonder if it is worth to use **dynamic programing**, instead of doing an **exhaustive search**. To decide which option is best one needs to estimate the computation involved in exploring each option. We shall do this assuming finite state-spaces and finite action spaces.
- **Exhaustive search:** Suppose that a game has K stages and that at the stage l the number of actions available to the player is equal to $|U_l|$. An exhaustive search over all possible selections of actions requires comparing the costs associated as many as options as $|U_1| \times |U_2| \times \ldots \times |U_K|$

Computational complexity

- **Dynamic Programing :** At a particular stage l and for a specific value of the state x, computing the cost-to-go $V_l(x)$ requires comparing among all the actions available, which roughly requires making $|U_l|$ comparisons. Since this has to be done for every state x and for every stage $\forall l \in \{1,2,...,K\}$, the total number of comparisons is roughly equal to

$$|U_1|\times|\chi_1|+|U_2|\times|\chi_2|+\ldots+|U_K|\times|\chi_K|$$

where we are denoting by $|\chi_l|$ the total number of possible states at the stage l. By comparing **Exhaustive search** with **Dynamic Programing** we see that dynamic programing can result in significant savings provided that the size of the spate space is small when compared to Exhaustive search .

Computational complexity (Example)

- **Tic-Tac-Toe**: Consider a (silly) version of the Tic-Tac-Toe game in which the same player places all the marks. An **exhaustive search** among all possible ways to play would have to consider 9 possible ways to place the first **x**, 8 possible way to place the subsequent **o**, 7 possible ways to place the second **x**, etc., leading to a total of

$$9! = 9 \times 8 \times 7 \times \ldots \times 1 = 362880$$

distinct options that must be compared.

Computational complexity (Example)

- For **dynamic programing**, the total number of comparisons needed is computed in Table below and turns out to be about 19 times smaller what would be needed for an exhaustive search. In larger games, the difference between the two approaches is even more spectacular. Essentially, this happens because many different sequences of actions collapse to the same state.

number of ×'s	number of ○'s	$\lvert\mathcal{X}_\ell\rvert$	$\lvert\mathcal{U}_\ell\rvert$	$\lvert\mathcal{X}_\ell\rvert \times \lvert\mathcal{U}_\ell\rvert$
0	0	1	9	9
1	0	9	8	72
1	1	$9 \times 8 = 72$	7	504
2	1	$\binom{9}{2} \times 7 = 252$	6	1512
2	2	$\binom{9}{2} \times \binom{7}{2} = 756$	5	3780
3	2	$\binom{9}{3} \times \binom{6}{2} = 1260$	4	5040
3	3	$\binom{9}{3} \times \binom{6}{3} = 1680$	3	5040
4	3	$\binom{9}{4} \times \binom{5}{3} = 1260$	2	2520
4	4	$\binom{9}{4} \times \binom{5}{4} = 630$	1	630
5	4	$\binom{9}{5} = 126$	0	0
		Total number of comparisons needed		19107

InfiniteDynamic Games

- Zero sum games
- Non-zero sum games
- Infinite Games
- **Infinite Dynamic Games**
  - Dynamic games in discrete time
  - Information structures
  - Continuous-time differential games
  - Discrete-time dynamic programming
  - **Continuous-time dynamic programming**

One-player Continuous-time differential games

Consider now a one-player continuous-time differential game with dynamics of the form

$$\underbrace{\dot{x}(t)}_{\text{state derivative}} = \underbrace{f}_{\text{game dynamics}}\left(\underbrace{t}_{\text{time}}, \underbrace{x(t)}_{\text{current state}}, \underbrace{u(t)}_{P_1\text{'s action at time } t}\right), \qquad \forall t \in [0,T] \qquad (4)$$

with state $x(t) \in R^n$ initialized at a given $x(0) = x_0$. For every time $t \in [0,T]$, the action $u(t)$ is required to belong to a given action space U. We assume a finite horizon ($T < \infty$) integral cost of the form

$$J = \int_0^T \underbrace{g(t, x(t), u(t))dt}_{\text{cost along trajctory}} + \underbrace{q(x(T))}_{\text{final cost}} \qquad (5)$$

One-player Continuous-time differential games

that the (only) player wants to minimize either using an **open-loop** policy

$$u(t) = \gamma^{OL}(t, x(0)) \qquad \forall t \in [0, T]$$

or a **(perfect) state-feedback** policy

$$u(t) = \gamma^{FB}(t, x(t)) \qquad \forall t \in [0, T]$$

❑ Continuous-time cost-to-go

The definition of cost-to-go for differential games follows the same scenario used for discrete-time games: A player finds herself at some state x at time τ and wants to estimate the minimum cost that she should expect, if she were to play so as to minimize the cost incurred from this point forwards until the end of the game.

Continuous-time cost-to-go

Formally, the cost-to-go from state x at time τ by

$$V(\tau, x) := \inf_{u(t) \in U, \forall t \in [\tau, T]} \int_{\tau}^{T} g(t, x(t), u(t)) dt + q(x(T)) \qquad (6)$$

with the state $x(t), \forall t \in [\tau, T]$ initialized at

$$x(\tau) = \mathrm{x}$$

and satisfying the dynamics

$$\dot{x} = f(\mathrm{t}, \mathrm{x}(\mathrm{t}), \mathrm{u}(\mathrm{t})), \qquad \forall t \in \{\tau, T\},$$

Continuous-time cost-to-go

- Computing the cost-to-go $V(0,x_0)$ from the initial state x_0 at time $\tau = 0$ essentially amounts to minimizing the cost (5) for the dynamics (4). This observation leads to two important conclusions

1) Regardless of the information structure considered (open loop, state feedback, or other), it is not possible to obtain a cost (5) lower than $V(0,x_0)$. This is because in the minimization in (6) we place no constraints on what information may or may not be available to compute the optimal $u(t), \forall t \in [\tau, T.]$

Continuous-time cost-to-go

2) If the infimum in (6) is achieved for for some specific signal

$$u^*(t) \in U, \qquad \forall t \in [\tau, T]$$

that can be computed before the game starts just with knowledge of x_0, then this action signal provides an optimal **open-loop** policy γ^{OL}

$$u^*(t) = \gamma^{OL}(t, x_0) \qquad \forall t \in [\tau, T]$$

Continuous-time dynamic programming

- Also in continuous-time it is possible to compute the cost-to-go somewhat recursively. For the final time *T*, the cost-to-go *V(T,x)* is simply given by

$$V(\mathrm{T},x)=q(x(T))=q(x)$$

- because for $\tau=T$ the integral term in (6) disappears and the game starts (and ends) precisely at *x(T)* = *x*.

Consider now some time $\tau<T$ and pick some small positive constant *h* so that $\tau+h$ *is* still smaller than *T*. Then

$$V(\tau,x):=\inf_{u(t)\in U,\forall t\in[\tau,T]}\int_{\tau}^{T}g(t,x(t),u(t))dt+q(x(T))$$

Continuous-time dynamic programming

$$V(\tau,x) := \inf_{u(t)\in U,\forall t\in[\tau,T]} \int_{\tau}^{T} g(t,x(t),u(t))dt + q(x(T))$$

$$= \inf_{u(t)\in U,\forall t\in[\tau,T]} \int_{\tau}^{\tau+h} \underbrace{g(t,x(t),u(t))dt}_{\text{independent of } u(t),\forall t\in[\tau+h,T]} + \int_{\tau+h}^{T} \underbrace{g(t,x(t),u(t))dt + q(x(T))}_{\text{depends on all } u(t),\quad \forall t\in[\tau,T]}$$

$$= \inf_{u(t)\in U,\forall t\in[\tau,\tau+h]} \Big(\int_{\tau}^{\tau+h} g(t,x(t),u(t))dt +$$

$$\inf_{u(t)\in U,\forall t\in[\tau+h,T]} \int_{\tau+h}^{T} g(t,x(t),u(t))dt + q(x(T)) \Big)$$

Continuous-time dynamic programming

- Recognizing that the "inner" infimum is precisely the cost-to-go from the state $x(\tau+h)$ at time $\tau+h$, we can re-write this equations compactly as

$$V(\tau,x) := \inf_{u(t)\in U,\forall t\in[\tau,\tau+h)} \left(\int_{\tau}^{\tau+h} g(t,x(t),u(t))dt + V(\tau+h,x(\tau+h)) \right)$$

- Subtracting $V(\tau,x) = V(\tau,x(\tau))$ from both sides and dividing both sides by $h > 0$, we can further re-write the above equation

$$0 = \inf_{u(t)\in U,\forall t\in[\tau,\tau+h)} \left(\frac{1}{h} \int_{\tau}^{\tau+h} g(t,x(t),u(t))dt + \frac{V(\tau+h,x(\tau+h)) - V(\tau,x)}{h} \right) \quad (7)$$

Continuous-time dynamic programming

- Since the left hand side must be equal to zero for every $h \in (0, \mathrm{T}-\tau)$, the limit of the right hand side as $h \to 0$ must also be equal to zero. If we optimistically assume that the limit of the infimum is the same as the infimum of the limit and also that all limits exist, we could use the following equalities

$$\lim_{h\to 0}\frac{1}{h}\int_{\tau}^{\tau+h} g(t, x(t), u(t))dt = g(\tau, x(\tau), u(\tau))$$

$$\lim_{h\to 0}\frac{V(\tau+h, x(\tau+h)) - V(\tau, x)}{h} = \frac{dV(\tau, x(\tau))}{d\tau}$$

$$= \frac{\partial V(\tau, x(\tau))}{\partial \tau} + \frac{\partial V(\tau, x(\tau))}{\partial x} f(\tau, x(\tau), u(\tau))$$

Continuous-time dynamic programming

- to transform (7) into the so-called **Hamilton-Jacobi-Bellman (HJB)** equation:

$$0 = \inf_{u \in U} \left(g(\tau, x, u) + \frac{\partial V(\tau, x)}{\partial \tau} + \frac{\partial V(\tau, x)}{\partial x} f(\tau, x, u) \right) \quad \forall \tau \in [0, \mathrm{T}], \mathrm{x} \in \mathrm{R}^n$$

It turns out that this equation is indeed quite useful to compute the cost-to-go:

Theorem (Hamilton-Jacobi-Bellman). Any continuously differentiable function $V(\tau, x)$ that satisfies the Hamilton-Jacobi-Bellman equation with

$$V(\mathrm{T}, x) = q(x), \qquad \forall \mathrm{x} \in \mathrm{R}^n$$

Continuous-time dynamic programming

is equal to the cost-to-go $V(\tau, x)$. In addition, if the infimum in the Hamilton-Jacobi-Bellman equation is always achieved at some point in U, we have that:

1. For any given x_0, an optimal open-loop policy γ^{OL} is given by

$$\gamma^{OL}(t, x_0) := u^*(t), \qquad \forall t \in [0, T]$$

with $u^*(t)$ obtained from solving

$$u^*(t) = \arg\min_{u \in U} g(\tau, x^*(t), u) + \frac{\partial V(\tau, x^*(t))}{\partial x} f(\tau, x^*(t), u)$$

$$\dot{x}^*(t) = f(\tau, x^*(t), u^*(t)), \qquad \forall t \in [0, T], \qquad x^*(0) = x_0$$

Continuous-time dynamic programming

2. An optimal (time consistent) state-feedback policy γ^{FB} is given by

$$\gamma^{FB}(t,x(t)) = \arg\min_{u\in U} g(\tau,x(\mathrm{t}),u) + \frac{\partial V(\tau,x(\mathrm{t}))}{\partial x} f(\tau,x(\mathrm{t}),u) \qquad \forall \mathrm{t} \in [0,\mathrm{T}]$$

Either of the above optimal policies leads to an optimal cost equal to $V(0, x_0)$.

- Open-loop and state-feedback information structures are "optimal," in the sense that it is not possible to achieve a cost lower than $V(0, x_0)$, regardless of the information structure.

Continuous-time dynamic programming

Proof of Theorem: Let $u^*(t)$ and $x^*(t)$ $\forall t \in [0,T]$ be a trajectory arising from either the open-loop or the state-feedback policies and let $\bar{u}(t)$ and $\bar{x}(t)$, $\forall t \in [0,T]$ be another arbitrary trajectory. To prove optimality, we need to show that the latter trajectory cannot lead to a cost lower than the former.

Since $V(\tau, x)$ satisfies the Hamilton-Jacobi-Bellman equation and $u^*(t)$ achieves the infimum in the Hamilton-Jacobi-Bellman equation, for every $\forall t \in [0,T]$, we have that

$$0 = \inf_{u \in U} \left(g(t, x^*(t), u) + \frac{\partial V(\tau, x^*(t))}{\partial \tau} + \frac{\partial V(t, x^*(t))}{\partial x} f(t, x^*(t), u) \right)$$

$$= g(t, x^*(t), u^*(t)) + \frac{\partial V(t, x^*(t))}{\partial t} + \frac{\partial V(t, x^*(t))}{\partial x} f(t, x^*(t), u^*(t)) \quad (8)$$

Continuous-time dynamic programming

However, since $\bar{u}(t)$ does not necessarily achieve the infimum, we have that

$$0 = \inf_{u \in U} \left(g(t, \bar{x}(t), u) + \frac{\partial V(t, \bar{x}(t))}{\partial t} + \frac{\partial V(t, \bar{x}(t))}{\partial x} f(t, \bar{x}(t), u) \right)$$

$$\leq g(t, \bar{x}(t), \bar{u}(t)) + \frac{\partial V(t, \bar{x}(t))}{\partial t} + \frac{\partial V(t, \bar{x}(t))}{\partial x} f(t, \bar{x}(t), \bar{u}(t)) \quad (9)$$

Integrating both side of (8) and (9) over the interval $[0, \mathrm{T}]$, we conclude that

$$0 = \int_0^T \left(g(t, x^*(t), u^*(t)) + \frac{\partial V(t, x^*(t))}{\partial t} + \frac{\partial V(t, x^*(t))}{\partial x} f(t, x^*(t), u^*(t)) \right) dt$$

$$\leq \int_0^T \left(g(t, \bar{x}(t), \bar{u}(t)) + \frac{\partial V(t, \bar{x}(t))}{\partial t} + \frac{\partial V(\tau, \bar{x}(t))}{\partial x} f(t, \bar{x}(t), \bar{u}(t)) \right) \mathrm{dt}$$

Continuous-time dynamic programming

from which we obtain

$$0 = \int_0^T g(t, x^*(t), u^*(t))\,\mathrm{dt} + V(T, x^*(T)) - V(0, x_0)$$

$$\leq \int_0^T g(t, \bar{x}(t), \bar{u}(t))\,\mathrm{dt} + V(T, \bar{x}(T)) - V(0, x_0)$$

Using boundary condition , two conclusions can be drawn from here: First, the signal $\bar{u}(t)$ does not lead to a cost smaller than that of $u^*(t)$, because

$$\int_0^T g(t, x^*(t), u^*(t))\,\mathrm{dt} + q(x^*(T)) \leq \int_0^T g(t, \bar{x}(t), \bar{u}(t))\,\mathrm{dt} + q(\bar{x}(T))$$

Continuous-time dynamic programming

Second, $V(0,x_0)$ is equal to the optimal cost obtained with $u^*(t)$, because

$$V(0,x_0)=\int_0^T g(t,x^*(t),u^*(t))\,\mathrm{dt}+q(x^*(T))$$

If we had carry out the above proof on an interval $[\tau,\mathrm{T}]$ with initial state $x(\tau)=\mathrm{x}$, we would have concluded that $V(\tau,x)$ is the (optimal) value of the cost-to-go from state x at time τ.

- In a open-loop setting both $u^*(t)$ and $x^*(t),\forall\,\mathrm{t}\in[0,\mathrm{T}]$ are pre-computed before the game starts.