CBC QuantEcon Workshop

Dynamic Programming

John Stachurski

September 2022

Introduction

Summary of this lecture:

- Overview of dynamic programming
- Introduce RDP framework and provide examples
- Provide RDP optimality results
- Discuss algorithms
- Study their performance for some applications
    - optimal savings, optimal investment...

Dynamic program

an initial state X_0 is given
$t \leftarrow 0$
while $t < T$ **do**
  observe current state X_t
  choose action A_t
  receive reward R_t based on (X_t, A_t)
  state updates to X_{t+1}
  $t \leftarrow t + 1$
end

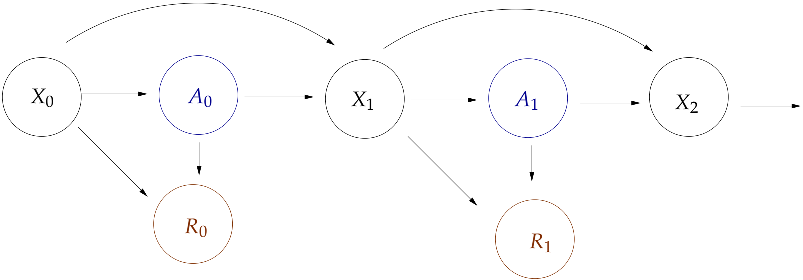


Figure: A dynamic program

Comments:

- Objective: maximize **lifetime rewards**
    - Example. $\mathbb{E}[R_0 + \beta R_1 + \beta^2 R_2 + \cdots]$ for some $\beta \in (0, 1)$
- If $T < \infty$ then the problem is called a **finite horizon** problem
- Otherwise it is called an **infinite horizon** problem

Example: Optimal Inventories

Given a demand process $(D_t)_{t \geqslant 0}$, inventory $(X_t)_{t \geqslant 0}$ obeys

$$X_{t+1} = F(X_t, A_t, D_{t+1})$$

where

- the **action** A_t is stock ordered this period
- $F(X, A, D) := \max\{X - D, 0\} + A$

The firm can store at most K items at one time

- The **state space** is $\mathsf{X} := \{0, \ldots, K\}$

We assume $(D_t) \overset{\text{IID}}{\sim} \varphi \in \mathcal{D}(\mathbb{Z}_+)$

Profits are

$$\pi_t := X_t \wedge D_{t+1} - cA_t - \kappa \mathbb{1}\{A_t > 0\}$$

- sales price $= 1$ and orders $>$ inventory are lost
- c is unit product cost
- κ is a fixed cost of ordering inventory

With $\beta := 1/(1+r)$, the value of the firm is

$$V_0 = \mathbb{E} \sum_{t \geqslant 0} \beta^t \pi_t$$

Objective: maximize (shareholder) value

Expected current profit is

$$r(x,a) := \sum_{d \geqslant 0} (x \wedge d)\varphi(d) - ca - \kappa \mathbb{1}\{a > 0\}$$

The **feasible correspondence** (which gives feasible order sizes) is

$$\Gamma(x) := \{0, \ldots, K - x\}$$

The **Bellman equation** is

$$v(x) = \max_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \sum_{d} v[F(x,a,d)]\varphi(d) \right\}$$

The solution v^* equals the value function

The **standard solution procedure** for this problem is VFI:

1. define the **Bellman operator** T via

$$(Tv)(x) = \max_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \sum_d v[F(x,a,d)] \varphi(d) \right\}$$

2. iterate with T to calculate $v \approx v^*$ and

3. compute a **v-greedy policy** σ^*, which satisfies

$$\sigma^*(x) \in \operatorname*{argmax}_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \sum_d v[F(x,a,d)] \varphi(d) \right\}$$

See notebook `inventory.ipynb`

Optimal Savings

Wealth evolves according to

$$C_t + W_{t+1} \leqslant RW_t + Y_t \qquad (t = 0, 1, \ldots)$$

- (W_t) takes values in finite set $\mathsf{W} \subset \mathbb{R}_+$
- (Y_t) is Q-Markov chain on finite set Y
- $C_t \geqslant 0$

The household maximizes

$$\mathbb{E} \sum_{t \geqslant 0} \beta^t u(C_t)$$

The Bellman equation is

$$v(w, y) = \max_{w' \in \Gamma(w,y)} \left\{ u(Rw + y - w') + \beta \sum_{y' \in \mathsf{Y}} v(w', y') Q(y, y') \right\}$$

The standard solution procedure is VFI

1. Set up Bellman operator T
2. Iterate with T from some initial guess to approximate v^*
3. Compute the v^*-greedy policy

Recursive Decision Processes

We will study an abstract dynamic program with Bellman equation

$$v(x) = \max_{a \in \Gamma(x)} B(x, a, v)$$

Advantages of "abstract" dynamic programming

- Subsumes standard Markov decision processes
- Can handle state-dependent discounting, recursive prefs, etc.
- Abstraction means clean proofs
- Abstraction allows better analysis of algorithms

Let X and A be finite sets (**state** and **action spaces**)

Actions are constrained by the **feasible correspondence** — a nonempty correspondence Γ from X to A

The feasible correspondence in turn defines

1. the **feasible state-action pairs**

$$\mathsf{G} := \{(x, a) \in \mathsf{X} \times \mathsf{A} : a \in \Gamma(x)\}$$

2. the set of **feasible policies**

$$\Sigma := \{\sigma \in \mathsf{A}^{\mathsf{X}} : \sigma(x) \in \Gamma(x) \text{ for all } x \in \mathsf{X}\}.$$

- "follow" $\sigma \iff$ always respond to state x with action $\sigma(x)$

Given X, A and Γ, a **recursive decision process** (RDP) consists of

1. a subset $\mathcal{V}$ of $\mathbb{R}^{\mathsf{X}}$ called the **candidate value functions** and
2. a **value aggregator**, which is a function

$$B \colon \mathsf{G} \times \mathcal{V} \to \mathbb{R}$$

satisfying $v, w \in \mathcal{V}$ and $v \leqslant w \implies$

$$B(x, a, v) \leqslant B(x, a, w) \ \text{ for all } (x, a) \in \mathsf{G}$$

and

$$\sigma \in \Sigma \ \text{ and } \ v \in \mathcal{V} \implies w \in \mathcal{V} \quad \text{where } w(x) := B(x, \sigma(x), v)$$

Example. For the inventory problem we set

- $\Gamma(x) := \{0, \dots, K - x\}$
- $\mathcal{V} = \mathbb{R}^{\mathsf{X}}$ and

$$B(x, a, v) := r(x, a) + \beta \sum_{d \geqslant 0} v[F(x, a, d)] \varphi(d)$$

The Bellman equation is then

$$v(x) = \max_{a \in \Gamma(x)} B(x, a, v)$$

The function B is a valid aggregator

For example, if $v \leqslant w$, then

$$B(x, a, v) \leqslant B(x, a, w)$$

Example. For the savings problem we set

- $\Gamma(w, y) := \{w' \in \mathsf{W} : w' \leqslant Rw + y\}$
- $\mathcal{V} = \mathbb{R}^{\mathsf{X}}$ and

$$B((w, y), w', v) := u(Rw + y - w') + \beta \sum_{y' \in \mathsf{Y}} v(w', y') Q(y, y')$$

The Bellman equation is then

$$v(w, y) = \max_{w' \in \Gamma(w,y)} B((w, y), w', v)$$

The function B is a valid aggergator

For example, if $f \leqslant g$, then

$$B((w, y), w', f) \leqslant B((w, y), w', g)$$

The RDP framework admits a huge range of generalizations

Example. State-dependent discounting: replace β with

$$\beta_t = \beta(Z_t) \text{ where } Z_t \text{ is a new state variable}$$

Example. Epstein–Zin preferences:

$$B(x, a, v) = \left\{ r(x, a)^\alpha + \beta \left[\sum_{x' \in \mathsf{X}} v(x')^\gamma P(x, a, x') \right]^{\alpha/\gamma} \right\}^{1/\alpha}$$

Example. Risk-sensitive preferences, ambiguity aversion, shortest path problems, etc.

Lifetime Value

Fix $\sigma \in \Sigma$

A $v \in \mathcal{V}$ that satisfies

$$v(x) = B(x, \sigma(x), v) \quad \text{for all } x \in \mathsf{X}$$

is called a **σ-value function**

Key idea: a σ-value function gives the <u>lifetime value</u> of following σ, from each state

Why is this interpretation valid?

Example. In the inventory problem, a σ-value function solves

$$v(x) = r(x, \sigma(x)) + \beta \sum_{d} v[F(x, \sigma(x), d)] \varphi(d)$$

With a change of variable, we can write this as

$$v(x) = r(x, \sigma(x)) + \beta \sum_{x'} v(x') P(x, \sigma(x), x')$$

where

$$P(x, a, x') := \mathbb{P}\{F(x, a, D) = x'\} \quad \text{when} \quad D \sim \varphi$$

In matrix notation,

$$v = r_\sigma + \beta \, P_\sigma \, v$$

Solving this equation gives the σ-value function:

$$v_\sigma = (I - \beta P_\sigma)^{-1} r_\sigma$$

Applying the Neumann series lemma, we can write v_σ as

$$v_\sigma = \sum_{t \geqslant 0} \beta^t P_\sigma^t r_\sigma$$

This is the lifetime value of the profit flow when

- following policy σ
- discounting at rate β

Now let's return to the general case, with RDP $(\Gamma, \mathcal{V}, B)$

Is lifetime value well-defined?

To answer this we introduce the **policy operator** T_σ via

$$(T_\sigma v)(x) = B(x, \sigma(x), v) \qquad (x \in \mathsf{X},\ v \in \mathcal{V})$$

Note: $v \in \mathcal{V}$ is a σ-valued function iff v is a fixed point of T_σ

Below we impose conditions under which T_σ always has a unique fixed point, **denoted by** v_σ

Hence lifetime value is always uniquely defined

With lifetime value uniquely defined for each σ, we can discuss optimality

A policy $\sigma^* \in \Sigma$ is called **optimal** if

$$v_{\sigma^*}(x) = \max_{\sigma \in \Sigma} v_\sigma(x) \quad \text{for all } x \in \mathsf{X}$$

Also, the **value function** is defined as

$$v^*(x) = \max_{\sigma \in \Sigma} v_\sigma(x) \quad (x \in \mathsf{X})$$

Hence σ^* is optimal iff $v_{\sigma^*} = v^*$

But how do we find optimal policies??

Operators

Given v in $\mathcal{V}$, we call $\sigma \in \Sigma$ **v-greedy** if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(x)} B(x, a, v) \qquad \text{for all } x \in \mathsf{X}$$

The **Bellman operator** is defined by

$$(Tv)(x) = \max_{a \in \Gamma(x)} B(x, a, v) \qquad (x \in \mathsf{X},\ v \in \mathcal{V})$$

Notes:

- v solves the Bellman equation iff v is a fixed point of T
- $(Tv)(x) = \max_{\sigma \in \Sigma} (T_\sigma\, v)(x)$

Stability

Let $\mathcal{R} := (\Gamma, \mathcal{V}, B)$ be an RDP with

- Bellman operator T and
- policy operators $\{T_\sigma\}_{\sigma \in \Sigma}$

We call $\mathcal{R}$ **globally stable** if

1. T is globally stable on $\mathcal{V}$ and
2. T_σ is globally stable on $\mathcal{V}$ for all $\sigma \in \Sigma$

Example. In the inventory problem, the operator T_σ is defined by

$$(T_\sigma v)(x) = r(x, \sigma(x)) + \beta \sum_d v[F(x, \sigma(x), d)] \varphi(d)$$

Hence, fixing $x \in \mathsf{X}$ and $v, w \in \mathcal{V}$,

$$\begin{aligned} &|(T_\sigma v)(x) - (T_\sigma w)(x)| \\ &\qquad = \beta \left| \sum_d v[F(x, \sigma(x), d)] \varphi(d) - \sum_d w[F(x, \sigma(x), d)] \varphi(d) \right| \end{aligned}$$

This is bounded above by

$$\beta \sum_d |v[F(x, \sigma(x), d)] - w[F(x, \sigma(x), d)]| \, \varphi(d) \leqslant \beta \|v - w\|_\infty$$

In summary,

$$|(T_\sigma v)(x) - (T_\sigma w)(x)| \leqslant \beta \|v - w\|_\infty \quad \text{for all } x \in \mathsf{X}$$

Taking the max over x gives

$$\|T_\sigma v - T_\sigma w\|_\infty \leqslant \beta \|v - w\|_\infty \quad \text{for all } x \in \mathsf{X}$$

Hence T_σ is a contraction on $\mathcal{V} = \mathbb{R}^{\mathsf{X}}$

In particular, T_σ is globally stable on $\mathcal{V}$

A similar argument works for T

Which other DP problems are globally stable?

- the optimal savings problem
- all standard Markov decision problems
- models with time-varying discount rates, under certain conditions
- models with risk-sensitive preferences, under some conditions
- models with Epstein–Zin preferences, under some conditions
- etc.

Theorem. For every globally stable RDP, the following statements are true:

1. The value function v^* satisfies the Bellman equation
2. v^* is the only fixed point of T in $\mathcal{V}$ and

$$\lim_{k \to \infty} T^k v = v^* \quad \text{for all } v \in \mathcal{V}$$

3. A policy $\sigma \in \Sigma$ is optimal if and only if it is v^*-greedy
4. At least one optimal policy exists

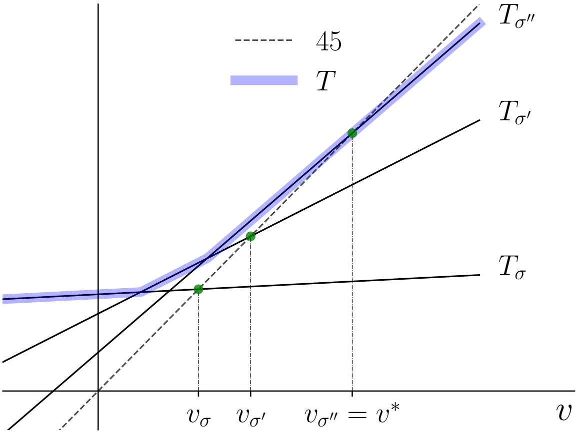


Figure: 1D case when $T_\sigma v = r_\sigma + \beta P_\sigma v$ and $\Sigma = \{\sigma, \sigma', \sigma''\}$

Algorithms

We used VFI to solve some simple problems

Next we

1. present a generalization of VFI suitable for aribtrary RDPs
2. introduce two other important methods

The two other methods are called

1. Howard policy iteration (HPI) and
2. Optimistic policy iteration (OPI)

Algorithm 1: VFI for RDPs

input $v_0 \in \mathbb{R}^{\mathsf{X}}$, an initial guess of v^*
input τ, a tolerance level for error
$\varepsilon \leftarrow \tau + 1$
$k \leftarrow 0$
while $\varepsilon > \tau$ **do**
    for $x \in \mathsf{X}$ **do**
        $v_{k+1}(x) \leftarrow (Tv_k)(x)$
    end
    $\varepsilon \leftarrow \|v_k - v_{k+1}\|_\infty$
    $k \leftarrow k + 1$
end
Compute a v_k-greedy policy σ
return σ

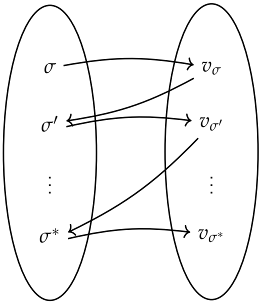
VFI is

- robust
- easy to implement
- very popular in economics (almost universal)

However,

- we can often find faster methods
- VFI is relatively serial — can be hard to parallelize efficiently

Howard Policy Iteration



Iterates between computing the value of a given policy and computing the greedy policy associated with that value

Algorithm 2: Howard policy iteration (HPI) for RDPs

input $\sigma_0 \in \Sigma$, an initial guess of σ^*
$k \leftarrow 0$
$\varepsilon \leftarrow 1$
while $\varepsilon > 0$ **do**
    $v_k \leftarrow$ the σ_k-value function
    $\sigma_{k+1} \leftarrow$ a v_k greedy policy
    $\varepsilon \leftarrow \|\sigma_k - \sigma_{k+1}\|_\infty$
    $k \leftarrow k + 1$
end
return σ_k

- In fact this is Newton's algorithm applied to T!

Advantages:

1. in a finite state setting, HPI always converges to the exact optimal policy in a finite number of steps
2. the rate of convergence is faster that VFI

But

- exact computation of the value of each policy can be problematic
- faster rate but the constant can be larger than VFI...

Optimistic Policy Iteration

OPI borrows from both value function iteration and Howard policy iteration

The same as Howard policy iteration (HPI) except that

- HPI takes σ and obtains v_σ
- OPI takes σ and iterates m times with T_σ

Recall that $T_\sigma^m \to v_\sigma$ as $m \to \infty$

Hence OPI replaces v_σ with an approximation

Algorithm 3: Optimistic policy iteration for RDPs

input $v_0 \in \mathbb{R}^{\mathsf{X}}$, an initial guess of v^*
input τ, a tolerance level for error
input $m \in \mathbb{N}$, a step size
$k \leftarrow 0$
$\varepsilon \leftarrow \tau + 1$
while $\varepsilon > \tau$ **do**
  $\sigma_k \leftarrow$ a v_k-greedy policy
  $v_{k+1} \leftarrow T_{\sigma_k}^m v_k$
  $\varepsilon \leftarrow \|v_k - v_{k+1}\|_\infty$
  $k \leftarrow k + 1$
end
return σ_k

Under mild conditions, $(\sigma_k)_{k \geqslant 1}$ converges to an optimal policy

Regarding m,

- $m = \infty \implies \text{OPI} = \text{HPI}$
- $m = 1 \implies \text{OPI} = \text{VFI}$

Often an intermediate value of m is better than both

We investigate efficiency of VFI–HPI–OPI in two applications

- `investment.ipynb`
- `opt_savings.ipynb`