Dynamic Programming

Chapter 6: Markov Decision Processes

Thomas J. Sargent and John Stachurski

2023

Topics

- Introduction to Markov decision processes
- Lifetime value
- Optimality
- Value function iteration
- Howard policy iteration
- Optimistic policy iteration
- Applications
- Refactoring dynamic programs

Markov Decision Processes (MDPs)

MDPs are a class of dynamic programs that are

- broad enough to encompass many economic applications
- includes optimal stopping problems as a special case
- admit a clean, powerful theory

Also a foundation for

- reinforcement learning
- artificial intelligence
- operations research, etc.

MDPs are dynamic programs characterized by two features

1. Aggregation of the reward sequence $(R_t)_{t \geqslant 0}$ is linear
2. The discount rate is constant

In particular,

$$\text{lifetime reward } = \mathbb{E} \sum_{t \geqslant 0} \beta^t R_t \; \text{ for some } \beta \in (0, 1)$$

States and Actions

We take as given

1. a finite set X called the **state space** and
2. a finite set A called the **action space**

A **correspondence** Γ from X to A is a map

$$\mathsf{X} \ni x \mapsto \Gamma(x) \subset \mathsf{A}$$

- called **nonempty** if $\Gamma(x)$ is not empty for all $x \in \mathsf{X}$

Below, $\Gamma(x) =$ "feasible actions" given x

We study a controller who, at each integer $t \geqslant 0$

1. observes the current state X_t
2. responds with an action A_t

Her aim is to maximize

$$\mathbb{E} \sum_{t \geqslant 0} \beta^t r(X_t, A_t) \quad \text{given } X_0 = x_0$$

Restrictions on Actions / Assumptions

Time travel is forbidden

- A_t cannot depend on $(X_{t+j})_{j \geqslant 1}$

The current state is sufficient

- A_t only depends on X_t

Below we write $A_t = \sigma(X_t)$ and call σ a **policy function**

Actions also restricted by a **feasible correspondence** Γ

- from X to A
- $\Gamma(x) =$ actions available to the controller in state x

Given Γ, we set

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

- called the set of **feasible state-action pairs**

Reward $r(x, a)$ is received at feasible state-action pair (x, a)

A **stochastic kernel** from G to X is a map $P \colon \mathsf{G} \times \mathsf{X} \to [0, 1]$ satisfying

$$\sum_{x' \in \mathsf{X}} P(x, a, x') = 1 \quad \text{for all } (x, a) \text{ in } \mathsf{G}$$

Interpretation

- next period state x' is drawn from distribution $P(x, a, \cdot)$

Now let's put it all together:

Given X and A, an **MDP** is a tuple (Γ, β, r, P) where

1. Γ is a nonempty correspondence from X $\to$ A
2. β is a constant in $(0, 1)$
3. r is a function from G to $\mathbb{R}$
4. P is a stochastic kernel from G to X

In what follows,

- β is called the **discount factor**
- r is called the **reward function**

Algorithm 1: MDP dynamics: states, actions, and rewards

$t \leftarrow 0$
input X_0
while $t < \infty$ **do**
    observe X_t
    choose action A_t from $\Gamma(X_t)$
    receive reward $r(X_t, A_t)$
    draw X_{t+1} from $P(X_t, A_t, \cdot)$
    $t \leftarrow t + 1$
end

Bellman's equation is

$$v(x) = \max_{a \in \Gamma(x)} \left\{ r(x, a) + \beta \sum_{x' \in \mathsf{X}} v(x') P(x, a, x') \right\}$$

Below we show that Bellman's equation reduces the infinite horizon problem to a two period problem

— "Bellman's principle of optimality"

In the two period problem, the controller trades off

1. current rewards and
2. expected discounted value from future states

Example. Cake eating (no labor income), where

$$W_{t+1} = R(W_t - C_t) \qquad (t = 0, 1, \ldots)$$

- Investing d dollars today returns Rd next period
- $C_t, W_t \geqslant 0$ are current consumption and wealth
- We restrict wealth to finite set $\mathsf{W} \subset \mathbb{R}_+$

The agent seeks to maximize $\mathbb{E} \sum_{t \geqslant 0} \beta^t u(C_t)$ given $W_0 = w$

Bellman equation:

$$v(w) = \max_{0 \leqslant w' \leqslant Rw} \left\{ u \left(w - \frac{w'}{R} \right) + \beta v(w') \right\}$$

This model can be framed as an MDP with W as the state space

- The action is $A_t = W_{t+1} =$ savings
- Hence the action space is also W
- The feasible correspondence is

$$\Gamma(w) = \{a \in \mathsf{W} : a \leqslant Rw\}$$

- Since $A_t = R(W_t - C_t)$, current reward is

$$r(w, a) = u(w - a/R) \qquad (a \in \Gamma(w))$$

- The stochastic kernel is $P(w, a, w') = \mathbb{1}\{w' = a\}$

Example. All optimal stopping problems can be framed as MDPs

- See the text for details

This is important from a theoretical perspective

- illustrates the generality of MDPs

However, expressing optimal stopping problems as an MDP requires an extra state variable

- status $S_t = \mathbb{1}\{\text{already stopped}\}$

Hence treating optimal stopping problems separately is neater

Policies

As discussed above, actions are governed by policies

- today's action is a function of today's state

In other words, a policy is an element σ of A^{X}

- respond to state X_t with action $A_t := \sigma(X_t)$ at **all** $t \geqslant 0$

A **feasible policy** is a

$$\sigma \in \mathsf{A}^{\mathsf{X}} \text{ such that } \sigma(x) \in \Gamma(x) \text{ for all } x \in \mathsf{X}$$

- Let $\Sigma :=$ the set of all feasible policies

What are the **dynamics** when $A_t := \sigma(X_t)$ at all $t \geqslant 0$?

Now

$$X_{t+1} \sim P(X_t, \sigma(X_t), \cdot) \quad \text{for all } t \geqslant 0$$

Thus, X_t updates according to the stochastic matrix

$$P_\sigma(x, x') := P(x, \sigma(x), x') \qquad (x, x' \in \mathsf{X})$$

The state process becomes P_σ-Markov

- Fixing a policy "closes the loop" in the state dynamics
- Solving an MDP means choosing a Markov chain!

Rewards

Under the policy σ,

- $(X_t)_{t \geqslant 0}$ is P_σ-Markov
- rewards at x are $r(x, \sigma(x))$

Let

- $r_\sigma(x) := r(x, \sigma(x))$
- $\mathbb{E}_x := \mathbb{E}[\, \cdot \mid X_0 = x]$

Now

$$\mathbb{E}_x \, r(X_t, A_t) = \mathbb{E}_x \, r_\sigma(X_t) = \sum_{x'} r_\sigma(x') P_\sigma^t(x, x') = (P_\sigma^t \, r_\sigma)(x)$$

The **lifetime value of σ** starting from x is

$$
\begin{aligned}
v_\sigma(x) &:= \mathbb{E}_x \sum_{t \geqslant 0} \beta^t r_\sigma(X_t) \\
&= \sum_{t \geqslant 0} \mathbb{E}_x \left[\beta^t r_\sigma(X_t) \right] \\
&= \sum_{t \geqslant 0} \beta^t (P_\sigma^t \, r_\sigma)(x)
\end{aligned}
$$

Since $\beta < 1$, the spectral radius of βP is < 1, so

$$
v_\sigma = \sum_{t \geqslant 0} (\beta P_\sigma)^t \, r_\sigma = (I - \beta P_\sigma)^{-1} \, r_\sigma
$$

Policy Operators

How should we compute v_σ given σ?

We saw above that

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

- Computationally helpful when X is small
- Problematic for large problems

Example. If $|\mathsf{X}| = 10^6$, then $I - \beta \, P_\sigma$ is $10^6 \times 10^6$

Matrices of this size are difficult invert—or even store in memory

Another way to compute v_σ: use the **policy operator**

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

- Defined at all $v \in \mathbb{R}^{\mathsf{X}}$
- Analogous to T_σ for the optimal stopping problem

In vector notation, we can write

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

- T_σ is order-preserving on $\mathbb{R}^{\mathsf{X}}$ — why?

Ex. Show that T_σ is a contraction of modulus β on $\mathbb{R}^{\mathsf{X}}$

For any v, w in $\mathbb{R}^{\mathsf{X}}$ we have

$$\begin{aligned} |T_\sigma v - T_\sigma w| &= \beta \; |P_\sigma v - P_\sigma w| \\ &= \beta \; |P_\sigma (v - w)| \\ &\leqslant \beta \, P_\sigma \, |v - w| \\ &\leqslant \beta \, P_\sigma \, \|v - w\|_\infty \mathbb{1} \\ &= \beta \, \|v - w\|_\infty \mathbb{1} \end{aligned}$$

Now use $|a| \leqslant |b|$ implies $\|a\|_\infty \leqslant \|b\|_\infty$

Ex. Show that T_σ is a contraction of modulus β on $\mathbb{R}^{\mathsf{X}}$

For any v, w in $\mathbb{R}^{\mathsf{X}}$ we have

$$\begin{aligned}
|T_\sigma v - T_\sigma w| &= \beta \; |P_\sigma v - P_\sigma w| \\
&= \beta \; |P_\sigma (v - w)| \\
&\leqslant \beta \, P_\sigma \, |v - w| \\
&\leqslant \beta \, P_\sigma \, \|v - w\|_\infty \mathbb{1} \\
&= \beta \, \|v - w\|_\infty \mathbb{1}
\end{aligned}$$

Now use $|a| \leqslant |b|$ implies $\|a\|_\infty \leqslant \|b\|_\infty$

Ex. Show that v_σ is the unique fixed point of T_σ in $\mathbb{R}^{\mathsf{X}}$

Proof: Since $\beta < 1$, we have

$$v = T_\sigma v \iff v = r_\sigma + \beta P_\sigma v$$

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

$$\iff v = v_\sigma$$

Hence

$$v \text{ is a fixed point of } T_\sigma \iff v = v_\sigma$$

Ex. Show that v_σ is the unique fixed point of T_σ in $\mathbb{R}^{\mathsf{X}}$

<u>Proof</u>: Since $\beta < 1$, we have

$$\begin{aligned} v = T_\sigma \, v \iff & v = r_\sigma + \beta P_\sigma v \\ \iff & v = (I - \beta P_\sigma)^{-1} \, r_\sigma \\ \iff & v = v_\sigma \end{aligned}$$

Hence

$$v \text{ is a fixed point of } T_\sigma \iff v = v_\sigma$$

Greedy Policies

Fix $v \in \mathbb{R}^{\mathsf{X}}$

A policy σ is called **v-greedy** if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(x)} \left\{ r(x, a) + \beta \sum_{x'} v(x') P(x, a, x') \right\}$$

for all $x \in \mathsf{X}$

Ex. Prove: at least one v-greedy policy exists in Σ

Proof: Immediate because $\Gamma(x)$ is finite and nonempty at all x

Greedy Policies

Fix $v \in \mathbb{R}^{\mathsf{X}}$

A policy σ is called **v-greedy** if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \sum_{x'} v(x') P(x,a,x') \right\}$$

for all $x \in \mathsf{X}$

Ex. Prove: at least one v-greedy policy exists in Σ

<u>Proof</u>: Immediate because $\Gamma(x)$ is finite and nonempty at all x

The Bellman Operator

Recall: Bellman's equation is

$$v(x) = \max_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \sum_{x' \in \mathsf{X}} v(x') P(x,a,x') \right\}$$

The **Bellman operator** for MDP (Γ, β, r, P) is the self-map on $\mathbb{R}^{\mathsf{X}}$ defined by

$$(Tv)(x) = \max_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \sum_{x' \in \mathsf{X}} v(x') P(x,a,x') \right\}$$

$$Tv = v \quad \iff \quad v \text{ satisfies Bellman's equation}$$

Ex. Prove: σ is v-greedy if and only if

$$T_\sigma \, v = T v$$

Proof: σ is v-greedy if and only if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(x)} \left\{ r(x, a) + \beta \sum_{x'} v(x') P(x, a, x') \right\} \qquad \forall \, x \in \mathsf{X}$$

This is equivalent to

$$r(x, \sigma(x)) + \beta \sum_{x'} v(x') P(x, \sigma(x), x') = (Tv)(x) \qquad \forall \, x \in \mathsf{X}$$

Ex. Prove: σ is v-greedy if and only if

$$T_\sigma v = Tv$$

<u>Proof</u>: σ is v-greedy if and only if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \sum_{x'} v(x') P(x,a,x') \right\} \qquad \forall\, x \in \mathsf{X}$$

This is equivalent to

$$r(x, \sigma(x)) + \beta \sum_{x'} v(x') P(x, \sigma(x), x') = (Tv)(x) \qquad \forall\, x \in \mathsf{X}$$

Ex. Prove that, for all $v \in \mathbb{R}^{\mathsf{X}}$,

$$Tv = \max_{\sigma \in \Sigma} T_\sigma \, v := \bigvee_{\sigma \in \Sigma} T_\sigma \, v$$

Proof:

Fixing $v \in \mathbb{R}^{\mathsf{X}}$, $\sigma \in \Sigma$ and $x \in \mathsf{X}$, we have

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

Conversely, for any v-greedy $\sigma \in \Sigma$, we have $T_\sigma \, v = Tv$

$$\therefore \quad Tv = \bigvee_{\sigma \in \Sigma} T_\sigma \, v$$

Ex. Prove that, for all $v \in \mathbb{R}^{\mathsf{X}}$,

$$Tv = \max_{\sigma \in \Sigma} T_\sigma \, v := \bigvee_{\sigma \in \Sigma} T_\sigma \, v$$

<u>Proof</u>:

Fixing $v \in \mathbb{R}^{\mathsf{X}}$, $\sigma \in \Sigma$ and $x \in \mathsf{X}$, we have

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

Conversely, for any v-greedy $\sigma \in \Sigma$, we have $T_\sigma \, v = Tv$

$$\therefore \quad Tv = \bigvee_{\sigma \in \Sigma} T_\sigma \, v$$

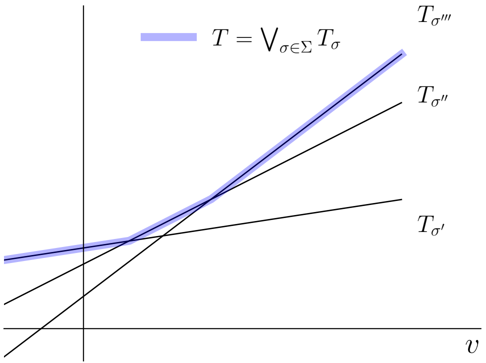


Figure: Visualization in one dimension

Ex. Prove: T is a contraction of modulus β on $\mathbb{R}^{\mathsf{X}}$

Proof: Recall that, for finite S and any $f, g \in \mathbb{R}^S$,

$$|\max_{s \in S} f(s) - \max_{s \in S} g(s)| \leqslant \max_{s \in S} |f(s) - g(s)|$$

Hence, for any v, w in $\mathbb{R}^{\mathsf{X}}$, we have

$$\begin{aligned} |(Tv)(x) - (Tw)(x)| &= \left| \max_{\sigma \in \Sigma} (T_\sigma \, v)(x) - \max_{\sigma \in \Sigma} (T_\sigma \, w)(x) \right| \\ &\leqslant \max_{\sigma \in \Sigma} |(T_\sigma \, v)(x) - (T_\sigma \, w)(x)| \end{aligned}$$

$$\therefore \quad |Tv - Tw| \leqslant \max_{\sigma \in \Sigma} |T_\sigma \, v - T_\sigma \, w| \leqslant \beta \|v - w\|_\infty$$

Ex. Prove: T is a contraction of modulus β on $\mathbb{R}^{\mathsf{X}}$

Proof: Recall that, for finite S and any $f, g \in \mathbb{R}^S$,

$$|\max_{s \in S} f(s) - \max_{s \in S} g(s)| \leqslant \max_{s \in S} |f(s) - g(s)|$$

Hence, for any v, w in $\mathbb{R}^{\mathsf{X}}$, we have

$$\begin{aligned} |(Tv)(x) - (Tw)(x)| &= \left| \max_{\sigma \in \Sigma} (T_\sigma\, v)(x) - \max_{\sigma \in \Sigma} (T_\sigma\, w)(x) \right| \\ &\leqslant \max_{\sigma \in \Sigma} |(T_\sigma\, v)(x) - (T_\sigma\, w)(x)| \end{aligned}$$

$$\therefore \quad |Tv - Tw| \leqslant \max_{\sigma \in \Sigma} |T_\sigma\, v - T_\sigma\, w| \leqslant \beta \|v - w\|_\infty$$

Optimality

The **value function** is defined by $v^* := \bigvee_{\sigma \in \Sigma} v_\sigma$

More explicitly,

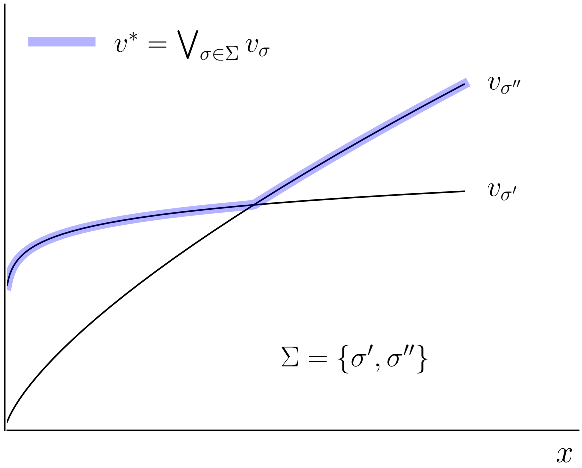
$$v^*(x) := \max_{\sigma \in \Sigma} v_\sigma(x) \qquad (x \in \mathsf{X})$$

Thus, $v^*(x) =$ **maximal lifetime value** from state x

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

$$v_\sigma = v^*$$

Thus, σ is optimal $\iff$ lifetime value is maximal at each state



$v^* = \bigvee_{\sigma \in \Sigma} v_\sigma$
$v_{\sigma''}$
$v_{\sigma'}$
$\Sigma = \{\sigma', \sigma''\}$
x

The last figure shows v^* when $\Sigma = \{\sigma', \sigma''\}$

Note that, as drawn, there is no optimal policy

Indeed, v^* differs from both $v_{\sigma'}$ and $v_{\sigma''}$

Hence no σ satisfies $v^* = v_\sigma$

Below we show that such an outcome is **not possible** for MDPs

In other words, an optimal policy always exists

This leads to our next slide...

Theorem. For any MDP (Γ, β, r, P) with Bellman operator T and value function v^*,

1. v^* is the unique fixed point of T in $\mathbb{R}^{\mathsf{X}}$
2. A feasible policy is optimal if and only it is v^*-greedy
3. At least one optimal policy exists

Remark: Point (2) is called **Bellman's principle of optimality**

We prove the proposition below through some exercises

Ex. Show that (2) implies (3)

Proof: For each $v \in \mathbb{R}^{\mathsf{X}}$, a v-greedy policy exists

Hence a v^*-greedy policy exists

Hence an optimal policy exists whenever (2) is valid

Ex. Show that (2) implies (3)

Proof: For each $v \in \mathbb{R}^{\mathsf{X}}$, a v-greedy policy exists

Hence a v^*-greedy policy exists

Hence an optimal policy exists whenever (2) is valid

We know that T is a contraction mapping on $\mathbb{R}^{\mathsf{X}}$

Hence T is globally stable on $\mathbb{R}^{\mathsf{X}}$ with unique fixed point $\bar{v} \in \mathbb{R}^{\mathsf{X}}$

To prove (1) of the above proposition, we have to show $\bar{v} = v^*$

Ex. Show that $\bar{v} \leqslant v^*$

Proof: Let $\sigma \in \Sigma$ be $\bar{v}$-greedy

Then $T_\sigma \bar{v} = T\bar{v} = \bar{v}$

Hence $\bar{v}$ is a fixed point of T_σ

But the only fixed point of T_σ in $\mathbb{R}^{\mathsf{X}}$ is v_σ

Hence $\bar{v} = v_\sigma$

But then $\bar{v} \leqslant v^*$, since $v^* = \bigvee_{\sigma \in \Sigma} v_\sigma$

Ex. Show that $v^* \leqslant \bar v$

Proof: Fix $\sigma \in \Sigma$ and note that $T_\sigma \, \bar v \leqslant T \bar v = \bar v$

Since T_σ is order-preserving and $T_\sigma \, \bar v \leqslant \bar v$, we have

$$T_\sigma^2 \, \bar v \leqslant T_\sigma \, \bar v \leqslant \bar v$$

Continuing in this way gives $T_\sigma^k \, \bar v \leqslant \bar v$ for all k

Taking the limit yields $v_\sigma \leqslant \bar v$

Hence $\bar v$ is an upper bound of $\{v_\sigma\}_{\sigma \in \Sigma}$

Therefore $v^* \leqslant \bar v$

Ex. Show that $v^* \leqslant \bar{v}$

Proof: Fix $\sigma \in \Sigma$ and note that $T_\sigma \, \bar{v} \leqslant T\bar{v} = \bar{v}$

Since T_σ is order-preserving and $T_\sigma \, \bar{v} \leqslant \bar{v}$, we have

$$T_\sigma^2 \, \bar{v} \leqslant T_\sigma \, \bar{v} \leqslant \bar{v}$$

Continuing in this way gives $T_\sigma^k \, \bar{v} \leqslant \bar{v}$ for all k

Taking the limit yields $v_\sigma \leqslant \bar{v}$

Hence $\bar{v}$ is an upper bound of $\{v_\sigma\}_{\sigma \in \Sigma}$

Therefore $v^* \leqslant \bar{v}$

We have now shown that $v^* = \bar{v}$

Thus, v^* is a fixed point of T in $\mathbb{R}^{\mathsf{X}}$

But T is globally stable on $\mathbb{R}^{\mathsf{X}}$

Hence v^* is the only fixed point of T in $\mathbb{R}^{\mathsf{X}}$

In other words,

$$v^* = \text{the unique solution to Bellman's equation in } \mathbb{R}^{\mathsf{X}}$$

To prove Bellman's principle of optimality, we use $Tv^* = v^*$

We have

$$\sigma \text{ is } v^*\text{-greedy} \quad \iff \quad T_\sigma \, v^* = Tv^* \quad \iff \quad T_\sigma \, v^* = v^*$$

The last statement is equivalent to $v_\sigma = v^*$ (why?)

Hence

$$\sigma \text{ is } v^*\text{-greedy} \iff v^* = v_\sigma \iff \sigma \text{ is optimal}$$

In other words, Bellman's principle of optimality holds

Algorithms

Previously we used value function iteration (VFI) to solve optimal stopping problems

Here we

1. present a generalization suitable for aribtrary MDPs
2. introduce two other important methods

The two other methods are called

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

Algorithm 2: VFI for MDPs

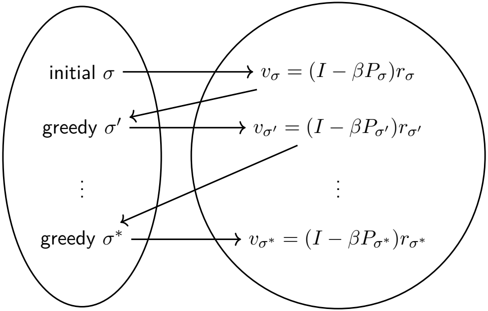
input $v_0 \in \mathbb{R}^{\mathsf{X}}$, an initial guess of v^*
input τ, a tolerance level for error
$\varepsilon \leftarrow \tau + 1$
$k \leftarrow 0$
while $\varepsilon > \tau$ **do**
    for $x \in \mathsf{X}$ **do**
        $v_{k+1}(x) \leftarrow (Tv_k)(x)$
    end
    $\varepsilon \leftarrow \|v_k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

VFI is

- globally convergent
- relatively robust
- easy to implement
- very popular

However, we can often find faster methods with a bit of effort

Howard Policy Iteration



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

Algorithm 3: Howard policy iteration for MDPs

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 $(I - \beta P_{\sigma_k})^{-1} r_{\sigma_k}$

  $\sigma_{k+1} \leftarrow$ a v_k-greedy policy

  $\varepsilon \leftarrow \|\sigma_k - \sigma_{k+1}\|_\infty$

  $k \leftarrow k + 1$

end

return σ_k

Advantages:

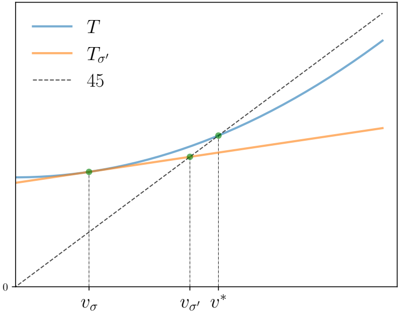
1. in this setting, always converges to the exact optimal policy in a finite number of steps
2. rate of convergence is faster that VFI

In fact HPI is analogous to gradient-based Newton iteration on T

- Details are in the text

In general, for a given fixed point problem,

1. Newton iteration yields a quadratic rate of convergence
2. Successive approximation yields a linear rate of convergence



- σ' is v_σ-greedy if $T_{\sigma'} v_\sigma = T v_\sigma$
- $v_{\sigma'}$ is the fixed point of $T_{\sigma'}$

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 4: Optimistic policy iteration for MDPs

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

Regarding m,

- If $m = \infty$, OPI is identical to HPI ($\lim_{m \to \infty} T_{\sigma_k}^m v_k = v_{\sigma_k}$)
- If $m = 1$, OPI is identical to VFI ($T_{\sigma_k} v_k = T v_k$)

Usually, an intermediate value of m is better than both

- We investigate this in the applications below

The sequence $(\sigma_k)_{k \geqslant 1}$ always converges to an optimal policy

Application: Optimal Inventories

Previously we analyzed S-s inventory dynamics

- our aim was to understand Markov chains

But are such dynamics realistic?

We now investigate whether S-s behavior arises naturally in optimizing model

- firm chooses its inventory path to maximize firm value

We assume for now that the firm only sells one product

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

$$X_{t+1} = m(X_t - D_{t+1}) + A_t$$

where

- $m(y) := y \vee 0$
- A_t is units of stock ordered this period
- 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 given by

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

- Orders in excess of inventory are lost
- c is unit product cost (and unit sales prices $= 1$)
- κ is a fixed cost of ordering inventory

With $\beta := 1/(1+r)$ and $r > 0$, the value of the firm is

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

Managers of the firm try to 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 set of feasible order sizes at x is

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

Bellman's equation is

$$v(x) = \max_{a \in \Gamma(x)} \left\{ r(x, a) + \beta \sum_{d \geqslant 0} v(m(x - d) + a)\varphi(d) \right\}$$

An MDP with state space X and action space A := X

- Γ, r and β are as given above
- The stochastic kernel is

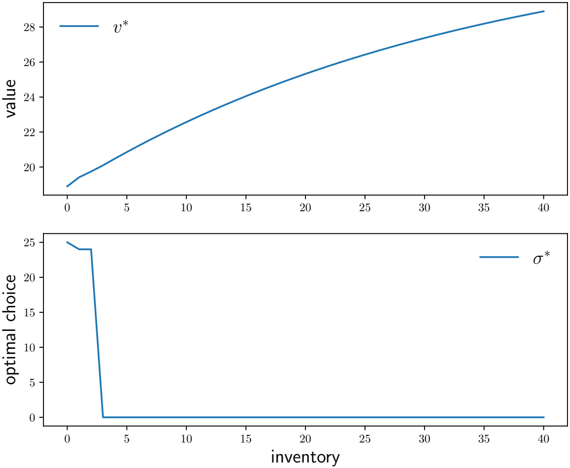
$$P(x, a, x') := \mathbb{P}\{m(x - D) + a = x'\} \quad \text{when} \quad D \sim \varphi$$

Since the inventory model is an MDP, all optimality results apply

- the unique fixed point of the Bellman operator is v^*
- a policy σ^* is optimal if and only if it is v^*-greedy
- etc.

```julia
using Distributions, OffsetArrays
m(x) = max(x, 0)  # Convenience function

function create_inventory_model(; β=0.98,     # discount factor
                                  K=40,       # maximum inventory
                                  c=0.2, κ=2, # cost paramters
                                  p=0.6)      # demand parameter
    ϕ(d) = (1 - p)^d * p  # demand pdf
    return (; β, K, c, κ, p, ϕ)
end

"The function B(x, a, v) = r(x, a) + β Σ_x′ v(x′) P(x, a, x′)."
function B(x, a, v, model; d_max=100)
    (; β, K, c, κ, p, ϕ) = model
    reward = sum(min(x, d)*ϕ(d) for d in 0:d_max) - c * a - κ * (a > 0)
    continuation_value = β * sum(v[m(x - d) + a] * ϕ(d) for d in 0:d_max)
    return reward + continuation_value
end
```

```julia
"The Bellman operator."
function T(v, model)
    (; β, K, c, κ, p, ϕ) = model
    new_v = similar(v)
    for x in 0:K
        Γx = 0:(K - x)
        new_v[x], _ = findmax(B(x, a, v, model) for a in Γx)
    end
    return new_v
end

"Get a v-greedy policy.  Returns a zero-based array."
function get_greedy(v, model)
    (; β, K, c, κ, p, ϕ) = model
    σ_star = OffsetArray(zeros(Int32, K+1), 0:K)
    for x in 0:K
        Γx = 0:(K - x)
        _, a_idx = findmax(B(x, a, v, model) for a in Γx)
        σ_star[x] = Γx[a_idx]
    end
    return σ_star
end
```



value
28
26
24
22
20
v^*
0
5
10
15
20
25
30
35
40
optimal choice
25
20
15
10
5
0
σ^*
0
5
10
15
20
25
30
35
40
inventory

Ex. Try to replicate these plots

- Use the code given above (or at least the same parameters)
- Use value function iteration

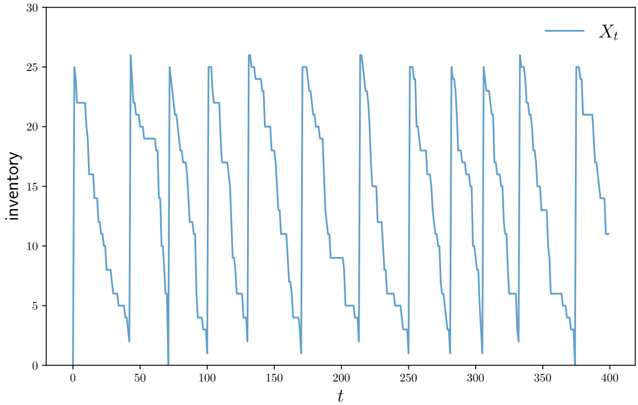


Figure: Optimal inventory dynamics

Optimal Savings with Labor Income

Wealth evolves according to

$$W_{t+1} = R(W_t + Y_t - C_t) \qquad (t = 0, 1, \ldots)$$

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

The household maximizes

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

The model is an MDP with state space $\mathsf{X} := \mathsf{W} \times \mathsf{Y}$

- The feasible correspondence is

$$\Gamma(w, y) = \{s \in \mathsf{W} : s \leqslant R(w + y)\}$$

- The current reward is

$$r(w, y, s) = u(w + y - s/R) \qquad \text{(utility of consumption)}$$

- The stochastic kernel is

$$P((w, y), s, (w', y')) = \mathbb{1}\{w' = s\} Q(y, y')$$

Hence all MDP optimality results apply

Bellman operator:

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

The policy operator for given $\sigma \in \Sigma$ is

$$(T_\sigma v)(w, y) = u(w + y - \sigma(w, y)/R) + \beta \sum_{y' \in \mathsf{Y}} v(\sigma(w, y), y') Q(y, y')$$

How to solve $v_\sigma = (I - \beta\, P_\sigma)^{-1} r_\sigma$?

We set

$$P_\sigma((w, y), (w', y')) := \mathbb{1}\{\sigma(w, y) = w'\} Q(y, y')$$

and

$$r_\sigma(w, y) := u(w + y - \sigma(w, y)/R)$$

How to use matrix algebra routines?

Set up a bijection $(i, j) \leftrightarrow m$ where

$$x_m = (w_i, y_j)$$

```julia
using QuantEcon, LinearAlgebra, IterTools

function create_savings_model(; R=1.01, β=0.98, γ=2.5,
                                w_min=0.01, w_max=5.0, w_size=200,
                                ρ=0.9, ν=0.1, y_size=5)
    w_grid = LinRange(w_min, w_max, w_size)
    mc = tauchen(y_size, ρ, ν)
    y_grid, Q = exp.(mc.state_values), mc.p
    return (; β, R, γ, w_grid, y_grid, Q)
end

"B(w, y, w′) = u(R*w + y - w′) + β Σ_y′ v(w′, y′) Q(y, y′)."
function B(i, j, k, v, model)
    (; β, R, γ, w_grid, y_grid, Q) = model
    w, y, w′ = w_grid[i], y_grid[j], w_grid[k]
    u(c) = c^(1-γ) / (1-γ)
    c = w + y - (w′ / R)
    @views value = c > 0 ? u(c) + β * dot(v[k, :], Q[j, :]) : -Inf
    return value
end
```

```julia
"The Bellman operator."
function T(v, model)
    w_idx, y_idx = (eachindex(g) for g in (model.w_grid, model.y_grid))
    v_new = similar(v)
    for (i, j) in product(w_idx, y_idx)
        v_new[i, j] = maximum(B(i, j, k, v, model) for k in w_idx)
    end
    return v_new
end

"The policy operator."
function T_σ(v, σ, model)
    w_idx, y_idx = (eachindex(g) for g in (model.w_grid, model.y_grid))
    v_new = similar(v)
    for (i, j) in product(w_idx, y_idx)
        v_new[i, j] = B(i, j, σ[i, j], v, model)
    end
    return v_new
end
```

```julia
include("finite_opt_saving_0.jl")

"Compute a v-greedy policy."
function get_greedy(v, model)
    w_idx, y_idx = (eachindex(g) for g in (model.w_grid, model.y_grid))
    σ = Matrix{Int32}(undef, length(w_idx), length(y_idx))
    for (i, j) in product(w_idx, y_idx)
        _, σ[i, j] = findmax(B(i, j, k, v, model) for k in w_idx)
    end
    return σ
end
```

```julia
"Get the value v_σ of policy σ."
function get_value(σ, model)
    # Unpack and set up
    (; β, R, γ, w_grid, y_grid, Q) = model
    wn, yn = length(w_grid), length(y_grid)
    n = wn * yn
    u(c) = c^(1-γ) / (1-γ)
    # Function to extract (i, j) from m = i + (j-1)*wn"
    single_to_multi(m) = (m-1)%wn + 1, div(m-1, wn) + 1
    # Allocate and create single index versions of P_σ and r_σ
    P_σ = zeros(n, n)
    r_σ = zeros(n)
    for m in 1:n
        i, j = single_to_multi(m)
        w, y, w′ = w_grid[i], y_grid[j], w_grid[σ[i, j]]
        r_σ[m] = u(w + y - w′/R)
        for m′ in 1:n
            i′, j′ = single_to_multi(m′)
            if i′ == σ[i, j]
                P_σ[m, m′] = Q[j, j′]
            end
        end
    end
    # Solve for the value of σ
    v_σ = (I - β * P_σ) \ r_σ
    # Return as multi-index array
    return reshape(v_σ, wn, yn)
```

```julia
"Value function iteration routine."
function value_iteration(model, tol=1e-5)
    vz = zeros(length(model.w_grid), length(model.y_grid))
    v_star = successive_approx(v -> T(v, model), vz, tolerance=tol)
    return get_greedy(v_star, model)
end

"Howard policy iteration routine."
function policy_iteration(model)
    wn, yn = length(model.w_grid), length(model.y_grid)
    σ = ones(Int32, wn, yn)
    i, error = 0, 1.0
    while error > 0
        v_σ = get_value(σ, model)
        σ_new = get_greedy(v_σ, model)
        error = maximum(abs.(σ_new - σ))
        σ = σ_new
        i = i + 1
        println("Concluded loop $i with error $error.")
    end
    return σ
end
```

```julia
"Optimistic policy iteration routine."
function optimistic_policy_iteration(model; tolerance=1e-5, m=100)
    v = zeros(length(model.w_grid), length(model.y_grid))
    error = tolerance + 1
    while error > tolerance
        last_v = v
        σ = get_greedy(v, model)
        for i in 1:m
            v = T_σ(v, σ, model)
        end
        error = maximum(abs.(v - last_v))
    end
    return get_greedy(v, model)
end
```

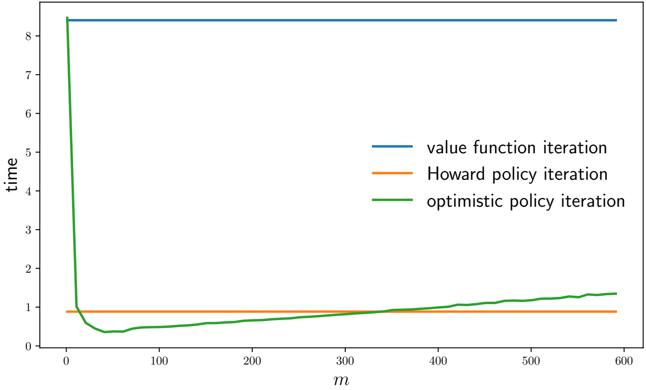


Figure: Timings for alternative algorithms

Investment with Adjustment Costs

A monopolist faces an inverse demand function of the form

$$P_t = a_0 - a_1 Y_t + Z_t,$$

where

- a_0, a_1 are positive parameters
- Y_t is output
- P_t is price and
- the demand shock Z_t follows

$$Z_{t+1} = \rho Z_t + \sigma \eta_{t+1}, \qquad \{\eta_t\} \overset{\text{IID}}{\sim} N(0,1).$$

Current profits are given by

$$\pi_t := P_t Y_t - c Y_t - \gamma (Y_{t+1} - Y_t)^2$$

- $\gamma (Y_{t+1} - Y_t)^2$ represents adjustment costs
- rapid changes to capacity are expensive when $\gamma > 0$

Objective: maximize value

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

- $\beta = 1/(1+r)$, where $r > 0$ is a fixed interest rate

Building intuition: What would happen if $\gamma = 0$?

- No intertemporal trade-off
- maximize current profit each period

Solve

$$\max_{Y_t} \{P_t Y_t - c Y_t\} = \max_{Y_t} \{(a_0 - a_1 Y_t + Z_t) Y_t - c Y_t\}$$

Ex. Show that the maximizer is

$$\bar{Y}_t := \frac{a_0 - c + Z_t}{2 a_1}$$

On the other hand, if γ is very large then $(Y_t)_{t\geqslant 0}$ should be almost constant

Thus, we expect the following:

- If $\gamma \approx 0$, then Y_t will track $\bar{Y}_t$ closely
- If γ is large, then $(Y_t)_{t\geqslant 0}$ will be smoother than $(\bar{Y}_t)_{t\geqslant 0}$

In short,

$$\text{more adjustment costs} \implies \text{smoother time path for } (Y_t)_{t\geqslant 0}$$

Implementation as an MDP

Let $\mathsf{Y} \subset \mathbb{R}_+$ be a grid containing output values

We discretize (Z_t) using Tauchen's method

- Now (Z_t) is Q-Markov on finite set $\mathsf{Z} \subset \mathbb{R}$

The state space $\mathsf{X} := \mathsf{Y} \times \mathsf{Z}$

The action space is Y

The feasible correspondence is $\Gamma(x) = \mathsf{Y}$ for all x

- choice of output is not restricted by the state

The set Σ is all $\sigma \colon \mathsf{Y} \times \mathsf{Z} \to \mathsf{Y}$

The current reward function is current profits:

$$r(y, z, \hat{y}) = (a_0 - a_1 y + z - c)y - \gamma(\hat{y} - y)^2$$

($\hat{y} \in \mathsf{Y}$ is our action: the choice of next-period output

The stochastic kernel is

$$P((y, z), \hat{y}, (y', z')) = \mathbb{1}\{y' = \hat{y}\} Q(z, z')$$

Now the problem defines an MDP

- all of the optimality theory for MDPs applies

Optimal Investment

The Bellman and policy operators for this problem are

$$(Tv)(y,z) = \max_{y' \in \mathbb{R}} \left\{ r(y,z,y') + \beta \sum_{z' \in \mathsf{Z}} v(y',z') Q(z,z') \right\}$$

$$(T_\sigma \, v)(y,z) = r(y,z,\sigma(y,z)) + \beta \sum_{z' \in \mathsf{Z}} v(\sigma(y,z),z') Q(z,z')$$

On $\mathbb{R}^{\mathsf{X}}$, both are

- order-preserving
- contractions of modulus β

A v-greedy policy is a $\sigma \in \Sigma$ that obeys

$$\sigma(y, z) = \operatorname*{argmax}_{y' \in \mathsf{Y}} \left\{ r(y, z, y') + \beta \sum_{z' \in \mathsf{Z}} v(y', z') Q(z, z') \right\}$$

By our results for MDPs

- v^*-greedy policies = optimal policies
- optimistic policy iteration and VFI converge

Implications for output can be studied by

1. generating a Q-Markov chain $(Z_t)_{t=1}^T$
2. simulating optimal output via $Y_{t+1} = \sigma^*(Y_t, Z_t)$

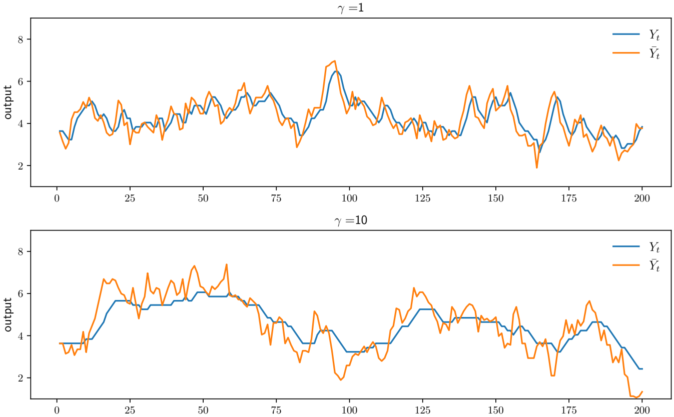
```julia
using QuantEcon, LinearAlgebra, IterTools
include("s_approx.jl")

function create_investment_model(;
        r=0.04,                              # Interest rate
        a_0=10.0, a_1=1.0,                   # Demand parameters
        γ=25.0, c=1.0,                       # Adjustment and unit cost
        y_min=0.0, y_max=20.0, y_size=100,   # Grid for output
        ρ=0.9, ν=1.0,                        # AR(1) parameters
        z_size=25)                           # Grid size for shock
    β = 1/(1+r)
    y_grid = LinRange(y_min, y_max, y_size)
    mc = tauchen(y_size, ρ, ν)
    z_grid, Q = mc.state_values, mc.p
    return (; β, a_0, a_1, γ, c, y_grid, z_grid, Q)
```

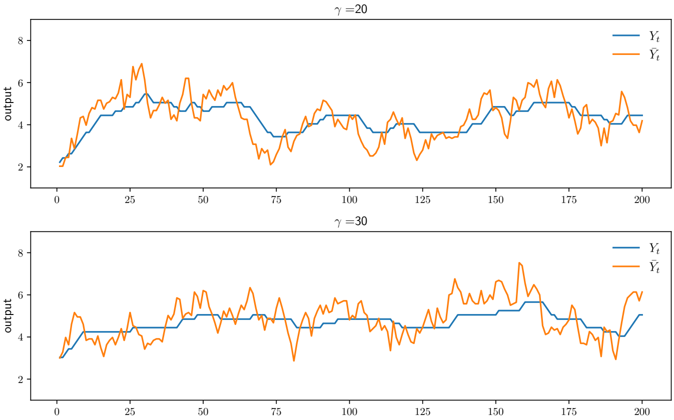
```julia
"""
The aggregator B is given by

    B(y, z, y′) = r(y, z, y′) + β Σ_z′ v(y′, z′) Q(z, z′)."

where

    r(y, z, y′) := (a_0 - a_1 * y + z - c) y - γ * (y′ - y)^2

"""
function B(i, j, k, v, model)
    (; β, a_0, a_1, γ, c, y_grid, z_grid, Q) = model
    y, z, y′ = y_grid[i], z_grid[j], y_grid[k]
    r = (a_0 - a_1 * y + z - c) * y - γ * (y′ - y)^2
    return @views r + β * dot(v[k, :], Q[j, :])
end
```

```julia
"The policy operator."
function T_σ(v, σ, model)
    y_idx, z_idx = (eachindex(g) for g in (model.y_grid, model.z_grid))
    v_new = similar(v)
    for (i, j) in product(y_idx, z_idx)
        v_new[i, j] = B(i, j, σ[i, j], v, model)
    end
    return v_new
end

"The Bellman operator."
function T(v, model)
    y_idx, z_idx = (eachindex(g) for g in (model.y_grid, model.z_grid))
    v_new = similar(v)
    for (i, j) in product(y_idx, z_idx)
        v_new[i, j] = maximum(B(i, j, k, v, model) for k in y_idx)
    end
    return v_new
end
```

```julia
"Compute a v-greedy policy."
function get_greedy(v, model)
    y_idx, z_idx = (eachindex(g) for g in (model.y_grid, model.z_grid))
    σ = Matrix{Int32}(undef, length(y_idx), length(z_idx))
    for (i, j) in product(y_idx, z_idx)
        _, σ[i, j] = findmax(B(i, j, k, v, model) for k in y_idx)
    end
    return σ
end

"Value function iteration routine."
function value_iteration(model; tol=1e-5)
    vz = zeros(length(model.y_grid), length(model.z_grid))
    v_star = successive_approx(v -> T(v, model), vz, tolerance=tol)
    return get_greedy(v_star, model)
end
```

```julia
"Optimistic policy iteration routine."
function optimistic_policy_iteration(model; tol=1e-5, m=100)
    v = zeros(length(model.y_grid), length(model.z_grid))
    error = tol + 1
    while error > tol
        last_v = v
        σ = get_greedy(v, model)
        for i in 1:m
            v = T_σ(v, σ, model)
        end
        error = maximum(abs.(v - last_v))
    end
    return get_greedy(v, model)
end
```


γ =1
output
Y_t
$\bar{Y}_t$
8
6
4
2
0
25
50
75
100
125
150
175
200
γ =10
output
Y_t
$\bar{Y}_t$
8
6
4
2
0
25
50
75
100
125
150
175
200


γ =20
output
Y_t
$\bar{Y}_t$
γ =30
output
Y_t
$\bar{Y}_t$

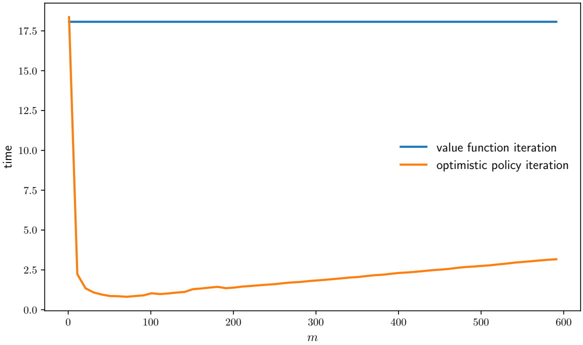


Figure: Timings for alternative algorithms, investment model

Notes on timing

- horizonal axis shows $m =$ step parameter in OPI
- vertical axis shows time in seconds
- Result for HPI not shown because time is 12x larger than VFI

Why is VFI faster than HPI here?

HPI tends to be strong when $\beta \approx 1$

- VFI convergence is linear in β, HPI convergence is quadratic

Here β is relatively small, so VFI beats HPI

Main messages

- OPI dominates both VFI and HPI for almost all values of m
- At $m = 60$, OPI is
    - 20 times faster than VFI
    - 240 times faster than HPI

Also note that OPI is easier to implement than HPI

- no need to map to single indices

Fixed Costs in Hiring and Firing

Consider a firm that maximizes expected present value

Future profits are discounted at rate

$$\beta = \frac{1}{1+r} \qquad r > 0$$

The only production input is labor

Hiring and firing involves fixed costs

Letting ℓ_t be employment, current profits are

$$\pi_t = pZ_t\ell_t^{\alpha} - w\ell_t - \kappa\mathbb{1}\{\ell_{t+1} \neq \ell_t\}$$

- p is the output price
- w is the wage rate
- α is a production parameter
- productivity $(Z_t)_{t\geqslant 0}$ is Q-Markov on Z and
- κ is a fixed cost of hiring and firing

Let $\mathsf{L} \subset \mathbb{R}_+$ be a finite grid for labor stock

The model is an MDP with state space $\mathsf{L} \times \mathsf{Z}$ and action space L

The feasible correspondence is

$$\Gamma(\ell, z) = \mathsf{L}$$

The reward function is

$$r(\ell, z, \ell') := pz\ell^{\alpha} - w\ell_t - \kappa \mathbb{1}\{\ell' \neq \ell\}$$

The stochastic kernel is

$$P((\ell, z), \ell', (\ell', z')) = \mathbb{1}\{\ell = \ell'\} Q(z, z')$$

Bellman operator:

$$(Tv)(\ell, z) = \max_{\ell' \in \Gamma(\ell, z)} \left\{ r(\ell, z, \ell') + \beta \sum_{z' \in \mathsf{Y}} v(\ell', z') Q(z, z') \right\}$$

The policy operator for given $\sigma \in \Sigma$ is

$$(T_\sigma v)(\ell, z) = r(\ell, z, \sigma(\ell, z)) + \beta \sum_{z' \in \mathsf{Y}} v(\sigma(\ell, z), z') Q(z, y')$$

A policy σ is v-greedy if

$$\sigma(\ell, z) \in \operatorname*{argmax}_{\ell' \in \Gamma(\ell, z)} \left\{ r(\ell, z, \ell') + \beta \sum_{z' \in \mathsf{Y}} v(\ell', z') Q(z, z') \right\}$$

```julia
using QuantEcon, LinearAlgebra, IterTools

function create_hiring_model(;
        r=0.04,                                 # Interest rate
        κ=1.0,                                  # Adjustment cost
        α=0.4,                                  # Production parameter
        p=1.0, w=1.0,                           # Price and wage
        l_min=0.0, l_max=30.0, l_size=100,      # Grid for labor
        ρ=0.9, ν=0.4, b=1.0,                    # AR(1) parameters
        z_size=100)                             # Grid size for shock
    β = 1/(1+r)
    l_grid = LinRange(l_min, l_max, l_size)
    mc = tauchen(z_size, ρ, ν, b, 6)
    z_grid, Q = mc.state_values, mc.p
    return (; β, κ, α, p, w, l_grid, z_grid, Q)
end
```

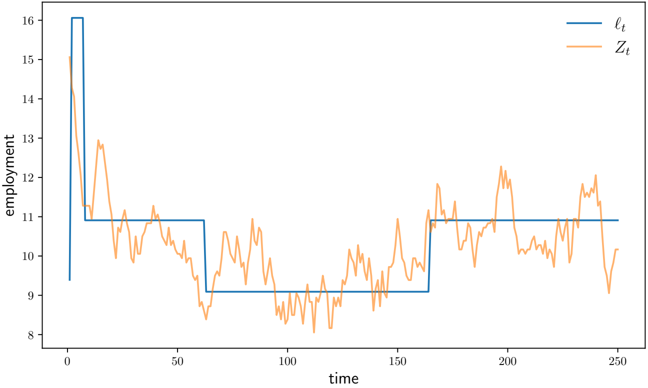


Figure: Fixed costs lead to jumps

Refactoring MDPs

Sometimes direct application of MDP theory is suboptimal

Example. We simplified job search by "refactoring" Bellman's equation

$$v^*(w) = \max \left\{ \frac{w}{1-\beta}, \, c + \beta \sum_{w'} v^*(w') \varphi(w') \right\} \qquad (w \in \mathsf{W})$$

into a recursion on the continuation value

$$h^* = c + \beta \sum_{w'} \max \left\{ \frac{w'}{1-\beta}, \, h^* \right\} \varphi(w')$$

- reduces dimensionality of iterative solution methods

Let's now examine this refactoring idea more systematically

Steps:

1. provide more examples
2. construct a theoretical foundation
3. connect with Bellman's principle of optimality
4. connect with VFI, OPI and HPI

Example: A Discrete Choice Problem

A structural estimation type Bellman's equation:

$$v(y, \varepsilon) = \max_{a \in \Gamma(y)} \left\{ r(y, \varepsilon, a) + \beta \sum_{y'} \int v(y', \varepsilon') P(y, a, y') \varphi(\varepsilon') \, \mathrm{d}\varepsilon' \right\}$$

for all $y \in \mathsf{Y}$ and $\varepsilon \in \mathsf{E}$

Here

- Y is a finite set — the **endogenous state** space
- ε is the **preference shock** — often vector-valued
- E, the outcome space for ε, is allowed to be continuous

Fix $v \in \mathbb{R}^{\mathsf{X}}$

We define the **expected value function** corresponding to v as

$$g(y, a) := \sum_{y'} \int v(y', \varepsilon') P(y, a, y') \varphi(\varepsilon') \, \mathrm{d}\varepsilon'$$

Key idea: work with expected value functions rather than value functions

Potential advantages:

- $|\mathsf{A}|$ can be much smaller than $|\mathsf{E}|$ (e.g., binary choice)
- integration provides smoothing to g

Using

$$g(y, a) = \sum_{y'} \int v(y', \varepsilon') P(y, a, y') \varphi(\varepsilon') \, \mathrm{d}\varepsilon'$$

We rewrite Bellman's equation as

$$v(y, \varepsilon) = \max_{a \in \Gamma(y)} \{r(y, \varepsilon, a) + \beta g(y, a)\}$$

Taking expectations of both sides gives

$$g(y, a) = \sum_{y'} \int \max_{a' \in \Gamma(y')} \left\{ r(y', \varepsilon', a') + \beta g(y', a') \right\} P(y, a, y') \varphi(\varepsilon') \, \mathrm{d}\varepsilon'$$

Next step: solve this equation for g

To solve for g we introduce the **expected value Bellman operator** R via

$$(Rg)(y,a) :=$$

$$\sum_{y'} \int \max_{a' \in \Gamma(y')} \left\{ r(y', \varepsilon', a') + \beta g(y', a') \right\} P(y, a, y') \varphi(\varepsilon') \, \mathrm{d}\varepsilon'$$

Ex. Show that R is an order-preserving self-map on $\mathbb{R}^{\mathsf{G}}$

Ex. Prove that R is a contraction of modulus β on $\mathbb{R}^{\mathsf{G}}$

- let g^* be the fixed point of R in $\mathbb{R}^{\mathsf{G}}$
- note g^* can be computed by successive approximation

Knowing this fixed point is enough to solve the dynamic program:

Proposition. A policy $\sigma \in \Sigma$ is optimal if and only if

$$\sigma(y, \varepsilon) \in \operatorname*{argmax}_{a \in \Gamma(y)} \{r(y, \varepsilon, a) + \beta g^*(y, a)\} \quad \text{for all } (y, \varepsilon) \in \mathsf{Y} \times \mathsf{E}$$

Rather than prove this here, we show something more general below

Q-Learning

Q-learning is a branch of reinforcement learning

- a sub-field of control and artificial intelligence
- dynamic optimization when some aspects of the system are unknown to the controller

Example. Solve MDPs where the full specification of state dynamics and rewards is not known

Q-learning relies on the essential equivalence of value functions and the "Q-factor"

We omit discussion of learning and focus on this equivalence

Fix

- an MDP (Γ, β, r, P) with state space X and action space A
- $v \in \mathbb{R}^{\mathsf{X}}$

The Q-**factor** corresponding to v is the function

$$q(x,a) = r(x,a) + \beta \sum_{x'} v(x') P(x,a,x') \qquad ((x,a) \in \mathsf{G})$$

- some economists call q the "post-action value function"

Given such a q, Bellman's equation can be written as

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

Taking expectations and discounting on both sides of

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

gives

$$\beta \sum_{x'} v(x') P(x, a, x') = \beta \sum_{x'} \max_{a' \in \Gamma(x')} q(x', a') P(x, a, x')$$

Adding $r(x, a)$ and using the definition of q again gives

$$q(x, a) = r(x, a) + \beta \sum_{x'} \max_{a' \in \Gamma(x')} q(x', a') P(x, a, x')$$

This is Bellman's equation expressed in terms of Q-factors

To solve for q we introduce the **Q-factor Bellman operator**

$$(Sq)(x, a) = r(x, a) + \beta \sum_{x'} \max_{a' \in \Gamma(x')} q(x', a') P(x, a, x')$$

Ex. Prove: S is an order-preserving contraction map on $\mathbb{R}^{\mathsf{G}}$

Let q^* be the unique fixed point of S in $\mathbb{R}^{\mathsf{G}}$

Proposition. A policy $\sigma \in \Sigma$ is optimal if and only if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(x)} q^*(x, a) \qquad \text{for all } (x, a) \in \mathsf{G}$$

We prove a more general result below

Operator Factorizations

Fix an MDP (Γ, β, r, P) with state space X and action space A

We seek general framework for "refactoring" this MDP

Questions: If we refactor the dynamic program,

- is Bellman's principle of optimality still valid?
- do the resulting "Bellman" operators always converge?
- can we use versions of OPI / HPI?

Our first step is to decompose T into separate parts

First we define

- $E\colon \mathbb{R}^{\mathsf{X}} \to \mathbb{R}^{\mathsf{G}}$ by $(Ev)(x,a) = \sum_{x'} v(x')P(x,a,x')$
- $D\colon \mathbb{R}^{\mathsf{G}} \to \mathbb{R}^{\mathsf{G}}$ by $(Dg)(x,a) = r(x,a) + \beta g(x,a)$
- $M\colon \mathbb{R}^{\mathsf{G}} \to \mathbb{R}^{\mathsf{X}}$ by $(Mq)(x) = \max_{a \in \Gamma(x)} q(x,a)$

Since

$$(Tv)(x) = \max_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \sum_{x' \in \mathsf{X}} v(x')P(x,a,x') \right\}$$

we have

$$Tv = MDEv \qquad (v \in \mathbb{R}^{\mathsf{X}})$$

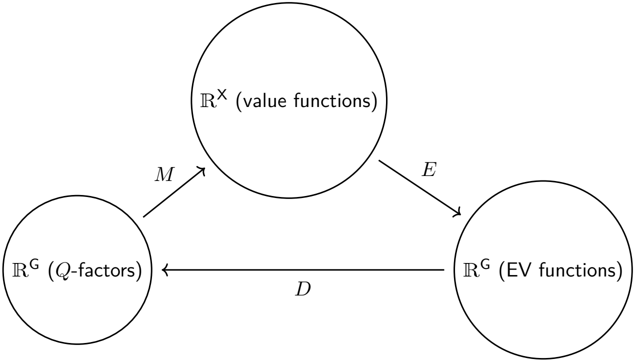
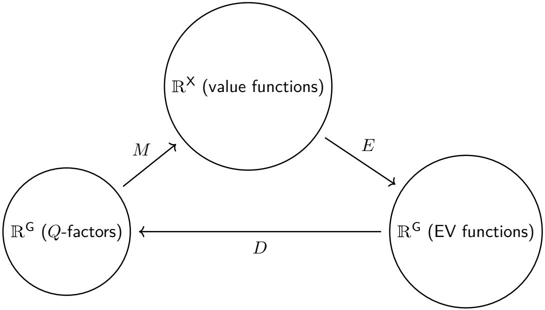


Figure: $T = MDE$

T is a round trip from the set of value functions

But notice that we have two other round trips



1. DME, from the expected value (EV) functions
2. MED, from the Q-factors

Thus we define

$$R := EMD, \quad S := DEM, \quad T := MDE$$

Ex. Show that

$$(Rg)(x,a) = \sum_{x'} \max_{a' \in \Gamma(x')} \left\{ r(x',a') + \beta g(x',a') \right\} P(x,a,x')$$

and

$$(Sq)(x,a) = r(x,a) + \beta \sum_{x'} \max_{a' \in \Gamma(x')} q(x',a') P(x,a,x')$$

We met these before!

- R is the **expected value Bellman operator**
- S is the **Q-factor Bellman operator**

Lemma. E and M are nonexpansive and D is a contraction. In particular, with $\| \cdot \| := \| \cdot \|_\infty$,

1. $\|Ev - Ev'\| \leqslant \|v - v'\|$ for all $v, v' \in \mathbb{R}^{\mathsf{X}}$
2. $\|Mq - Mq'\| \leqslant \|q - q'\|$ for all $g, g' \in \mathbb{R}^{\mathsf{G}}$
3. $\|Dg - Dg'\| \leqslant \beta\|g - g'\|$ for all $q, q' \in \mathbb{R}^{\mathsf{G}}$

Proof: For E we have

$$|(Ev)(x,a)| = \left| \sum_{x'} v(x') P(x,a,x') \right| \leqslant \sum_{x'} \left| v(x') \right| P(x,a,x') \leqslant \|v\|$$

Taking the sup gives $\|Ev\| \leqslant \|v\|$ and linearity now gives

$$\|Ev - Ev'\| = \|E(v - v')\| \leqslant \|v - v'\|$$

For M we can use the bound

$$|\max_{s\in S} f(s) - \max_{s\in S} g(s)| \leqslant \max_{s\in S} |f(s) - g(s)|$$

to obtain

$$\begin{aligned} |(Mq)(x) - (Mq')(x)| &= |\max_{a\in\Gamma(x)} q(x,a) - \max_{a\in\Gamma(x)} q'(x,a)| \\ &\leqslant \max_{a\in\Gamma(x)} |q(x,a) - q'(x,a)| \leqslant \|q - q'\| \end{aligned}$$

Now take the supremum over x to get

$$\|Mq - Mq'\| \leqslant \|q - q'\|$$

Regarding D, for fixed $g, g' \in \mathbb{R}^{\mathsf{G}}$, we have

$$\begin{aligned}
|(Dg)(x,a) - (Dg')(x,a)| &= |r(x,a) + \beta g(x,a) - r(x,a) - \beta g'(x,a)| \\
&= \beta |g(x,a) - g'(x,a)| \\
&\leqslant \beta \|g - g'\|
\end{aligned}$$

$$\therefore \quad \|Dg - Dg'\| \leqslant \beta \|g - g'\|$$

Proposition. R, S and T are all contractions of modulus β

Proof: Let's just prove this for $R = EMD$

Fixing $g, g' \in \mathbb{R}^{\mathsf{G}}$, we have

$$\begin{aligned} \|Rg - Rg'\| &= \|EMDg - EMDg'\| \\ &\leqslant \|MDg - MDg'\| \\ &\leqslant \|Dg - Dg'\| \\ &\leqslant \beta \|g - g'\| \end{aligned}$$

The proofs for S and T are very similar

It follows that R, S and T all have unique fixed points

We denote them by g^*, q^* and v^* respectively:

$$Rg^* = g^*, \quad Sq^* = q^*, \quad \text{and} \quad Tv^* = v^*$$

As suggested by notation, $v^* = \bigvee_{\sigma \in \Sigma} v_\sigma$

- We proved above that v^* is the unique fixed point of T in $\mathbb{R}^{\mathsf{X}}$

How are these fixed points related to each other?

Proposition. The fixed points of R, S and T are connected via

1. $g^* = Ev^*$
2. $q^* = Dg^*$
3. $v^* = Mq^*$

Proof of 1:

We have $Ev^* = ETv^* = EMDEv^* = REv^*$

Hence Ev^* is a fixed point of R

But R has only one fixed point, which is g^*

Therefore, $g^* = Ev^*$

The proofs of 2–3 are analogous

The results in the last proposition can be written more explicitly as

$$g^*(x,a) = \sum_{x'} v^*(x') P(x,a,x') \quad ((x,a) \in \mathsf{G})$$

$$q^*(x,a) = r(x,a) + \beta g^*(x,a) \quad ((x,a) \in \mathsf{G})$$

and

$$v^*(x) = \max_{a \in \Gamma(x)} q^*(x,a) \quad (x \in \mathsf{X})$$

A policy σ is called v-**greedy** if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \sum_{x'} v(x') P(x,a,x') \right\}$$

for all $x \in \mathsf{X}$

Fix $g, q \in \mathbb{R}^{\mathsf{G}}$

We call $\sigma \in \Sigma$ g-**greedy** if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(y)} \{ r(x,a) + \beta g(x,a) \} \qquad (x \in \mathsf{X})$$

and q-**greedy** if

$$\sigma(x) \in \operatorname*{argmax}_{a \in \Gamma(y)} q(x,a) \qquad (x \in \mathsf{X})$$

Proposition. For $\sigma \in \Sigma$, the following statements are equivalent:

1. σ is v^*-greedy
2. σ is g^*-greedy
3. σ is q^*-greedy

In particular,

$$\sigma \text{ is optimal } \iff \text{ any one (and hence all) of 1–3 holds}$$

"Refactored" versions of Bellman's principle of optimality

One consequence: we can modify VFI to operate on either expected value functions or Q-factors

Example. Suppose we find it more convenient to iterate in expected value space

Then we can proceed as follows:

1. Fix $g \in \mathbb{R}^{\mathsf{G}}$
2. Iterate with R to obtain $g_k := R^k g \approx g^*$
3. Compute a g_k-greedy policy

Since $g_k \approx g^*$, the resulting policy will be approximately optimal

Refactored OPI

We saw above that VFI is often outperformed by HPI / OPI

Can we apply these methods to refactored MDPs?

Then we could combine

1. the speed gains from HPI/OPI
2. the potential efficiency gains obtained by refactoring

Below we provide an affirmative answer: build a version of OPI that can compute expected value functions

The same is true for OPI for Q-factors, HPI — details omitted

First, given $\sigma \in \Sigma$, we introduce

$$(M_\sigma \, q)(x) := q(x, \sigma(x)) \qquad (x \in \mathsf{X}, \; q \in \mathbb{R}^{\mathsf{G}})$$

Then we define

$$R_\sigma := E \, M_\sigma \, D \qquad S_\sigma := D \, E \, M_\sigma \qquad T_\sigma := M_\sigma \, D \, E$$

In fact T_σ is just the ordinary MDP policy operator:

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

Let's call R_σ and S_σ the **expected-value policy operator** and **Q-factor policy operator** respectively

Here's an expected value version of the OPI algorithm

input $g_0 \in \mathbb{R}^{\mathsf{G}}$, an initial guess of g^*
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 g_k-greedy policy
    $g_{k+1} \leftarrow R^m_{\sigma_k} g_k$
    $\varepsilon \leftarrow \|g_k - g_{k+1}\|_\infty$
    $k \leftarrow k + 1$
end
return σ_k

Expected value OPI is globally convergent in the same sense as regular OPI

Indeed, suppose we

1. pick $v_0 \in \mathbb{R}^{\mathsf{X}}$,
2. apply regular OPI starting from v_0, and
3. apply expected value OPI applied to $g_0 := E v_0$,

then

- the sequences $(v_k)_{k \geqslant 0}$ and $(g_k)_{k \geqslant 0}$ generated by the two algorithms are connected via $g_k = E v_k$ for all $k \geqslant 0$
- the policy sequences generated by the two algorithms are identical in value

Proof: (assuming for convenience that greedy policies are unique)

Consider the claim that $g_k = Ev_k$ for all $k \geqslant 0$

True by assumption when $k = 0$

Suppose, as an induction hypothesis, that $g_k = Ev_k$ holds at arbitrary k

If σ is g_k-greedy, then

$$\sigma(x) = \operatorname*{argmax}_{a \in \Gamma(y)} \{r(x,a) + \beta(Ev_k)(x,a)\}$$

$$= \operatorname*{argmax}_{a \in \Gamma(y)} \left\{ r(x,a) + \beta \sum_{x'} v_k(x') P(x,a,x') \right\}$$

Hence σ is both g_k-greedy and v_k-greedy

Therefore σ is the next policy selected by both versions of OPI

Moreover, updating via expected value OPI,

$$
\begin{aligned}
g_{k+1} = R_\sigma^m g_k &= E T_\sigma^{m-1} M_\sigma D g_k \\
&= E T_\sigma^{m-1} M_\sigma D E v_k \\
&= E T_\sigma^m v_k
\end{aligned}
$$

Since σ is v_k-greedy, we have $v_{k+1} = T_\sigma^m v_k$

Hence $g_{k+1} = E v_{k+1}$

This completes the proof that $g_k = E v_k$ for all k

In fact we just showed that the policy functions generated by the algorithms are identical as well