Introduction to Dynamic Programming

Part I: An Overview

PPE Phil(e)[1]

[1]material @ https://github.com/PPEphile

September 2021

Table of Contents

Dynamic Programming

Dynamic Programming is a technique for solving (dynamic) **optimization** problems.

- Dynamic: your **rewards** and **limits** change, depending on your actions
- Programming: you try to identify the best strategies (plans/policies/programs) (i.e. which maximise or minimise a goal function)

A dynamic optimization problem often has the form:

$$\max_{(c_t)_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t U(c_t) \quad s.t. \quad \text{constraints fulfilled for } \forall t \tag{1}$$

where we for now assume that a maximum really exists.

An Example: Growing corn

- Infinite horizon: t=0,1,2, ...
- Each period we decide how much of the harvest to consume: c_t
- Remainder is planted for next period: $f(k_t) - c_t = k_{t+1}$
- We start with a stock of corn of k_0
- Eating corn gives us utility $U(c_t)$, but we discount future meals at the rate β^t

Substituting c_t for $f(k_t) - k_{t+1}$, we have

$$\max_{0 \leq k_{t+1} \leq f(k_t)} \sum_{t=0}^{\infty} \beta^t U(f(k_t) - k_{t+1}) \tag{2}$$

Hence, we have an infinite number of variables and constraints!

An indirect attack: Dynamic Programming

"Direct attacks" (e.g. Kuhn-Tucker & Lagrange) only sometimes work in such scenarios. Dynamic Programming approaches the problem indirectly via the so-called **value function**.

- Translate $U(f(k_t) - k_{t+1}) \to F(k_t, k_{t+1})$

We now call the **Sequence Problem (SP)**

$$\max_{0 \leq k_{t+1} \leq f(k_t)} \sum_{t=0}^{\infty} \beta^t F(k_t, k_{t+1}) \tag{SP}$$

The **value function** is defined as the solution to the (SP)

$$v^*(k_0) = \max_{0 \leq k_{t+1} \leq f(k_t)} \sum_{t=0}^{\infty} \beta^t F(k_t, k_{t+1})$$

From (SP) to Bellman Equation

But now, something interesting happens:

$$\begin{aligned}
v^*(k_0) &= \max_{0 \le k_{t+1} \le f(k_t)} \sum_{t=0}^{\infty} \beta^t F(k_t, k_{t+1}) \\
&= \max_{0 \le k_1 \le f(k_0)} \left\{ F(k_0, k_1) + \max_{0 \le k_{t+1} \le f(k_t)} \sum_{t=1}^{\infty} \beta^t F(k_t, k_{t+1}) \right\} \\
&= \max_{0 \le k_1 \le f(k_0)} \left\{ F(k_0, k_1) + \beta \max_{0 \le k_{t+2} \le f(k_t)} \sum_{t=0}^{\infty} \beta^t F(k_{t+1}, k_{t+2}) \right\} \\
&= \max_{0 \le k_1 \le f(k_0)} \{ F(k_0, k_1) + \beta v^*(k_1) \}
\end{aligned}$$

This is called the Bellman equation, which uses Bellman's **Principle of Optimality**: For a policy to be optimal, whatever we choose today, the remaining decisions must still be optimal.

In the Bellman equation we just saw that

$$v^*(k_0) = \max_{0 \le k_1 \le f(k_0)} \{F(k_0, k_1) + \beta v^*(k_1)\}$$

As a result, we know that for any candidate function v to be the true value function v^*, it needs to map onto itself! We call this **necessary condition** the **Functional Equation (FE)**

Functional Equation

$$v(x) = \max_{y \in \Gamma(x)} \{F(x, y) + \beta v(y)\} \tag{FE}$$

where $\Gamma(x)$ is the set of admissible values of y given the current state x.

- This is a "functional" equation because the unknown is a *function*.

Wrap-up

- We can now start to see what it means that Dynamic Programming approaches the (SP) through the value function.
  - As we will see in the next videos, Dynamic Programming makes use of the fact the we can often learn a lot about the value function, which, in turn, helps us learning about the optimal policy.

Check-points for today:

- Understand what a dynamic optimization problem is
- Understand what a value function is
- Conceptualize that we are searching for an unknown function

Additional Material

In notation and exposition, the series closely orients itself on Stokey and Lucas (1989), which is one of the most popular textbooks on Dynamic Programming. If you don't understand something or wish to dig deeper, please check-out the textbook.

Helpful references:

- Simon, C.P. and Blume, L. (1994) *Mathematics for Economists*. New York.
- Stokey, N.L., Lucas, R.E. and Prescott, E.C. (1989) *Recursive Methods in Economic Dynamics*. Harvard University Press. doi:10.2307/j.ctvjnrt76.