Dynamic Programming

Part 1: Exercise 1 & 2 (week 7)

Mikkel Reich

University of Copenhagen, Department of Economics

Plan for today

- A bit about me
- Plan for exercises generally
- Coding tips
- A bit about dynamic programming as a course and a "concept"
- Exercise 1 & 2: Cake-eating in finite and infinite horizon

Who is your TA?

- 2nd year Master student in Economics
- Interests in macroeconomics, computional methods, (structural) econometrics, mathematical finance
- Been "around the block" TA-wise:
  - Macroeconomics 1 & 2 (Math-Econ)
  - Introduction to Social Data Science
  - Macroeconomics III
  - Probability Theory and Statistics
  - And now: Dynamic Programming
- Research Assistant for Assoc. Prof. Jeppe Druedahl, working with dynamic programming and deep reinforcement learning
- Can contact me on absalon or mail: qxb650@ku.dk

Plan for exercises

- Wednesdays 8:15 - 10 (class 1) and Thursdays 8:15 - 10 (class 2)
- Hands-off approach from my side $\rightarrow$ Exercises for coding!
  - Brief introduction to topic/model in beginning of class (5-15 min)
- Problem sets are in notebooks and py-files (to be filled out)
- Exercises are divided into 3 parts/blocks, each of 3-4 weeks:
  1. Theory and tools
  2. Dynamic Discrete Choice (structural econometrics)
  3. Discrete-Continuous Choice (more advanced applications)
- No exercices in games $\rightarrow$ See my term paper for inspiration :)
- Exercises are all <u>python-based</u>
- No assignments

Coding tips

- Work-flow: Fill out notebooks and py-files (structure for project)
  - `ex_ante` folder contains "unsolved" notebooks
  - `ex_post` folder contains notebooks with solutions → Check!
- First cell in each notebook: Notebook checks changes in py-files.

```
%load_ext autoreload
%autoreload 2
```

- If you use CoPilot: Disable while solving exercises since CoPilot looks in `ex_post` and inserts solution
- Small guide on smart use of GitHub repository

What is Dynamic Programming?

- **Dynamic programming $\approx$ Dynamic optimization**

  (US military lingo: *"finding the optimal program"*, link for story)

  (Dynamic: *"over time / multi-stage"*, another story)

- Want to find $\{c_t^*\}_{t=1}^T$ that maximizes $\mathbb{E}[\sum_{t=1}^T \beta^{t-1} u_t(c_t)]$
- Reformulate problem with **Bellman equation** (general form):

$$V_t(s_t) = \max_{c_t} u_t(c_t, s_t) + \beta \mathbb{E}_t[V_{t+1}(s_{t+1})] \quad \text{s.t. } s_{t+1} = f(s_t, c_t)$$

$\rightarrow$ An application of Bellman's **Principle of Optimality**:

When we optimize in period t, we don't have to think about the past or how we got "here", only that the future actions are optimal

Exercise 1: Cake-eating in a finite horizon

We have a cake with W slices. We want to optimally eat the cake over T periods, we recieve utility $u(c_t) = \sqrt{c_t}$. Find the optimal policy $\{c_t^*\}_{t=1}^T$ such that:

$$\max_{\{c_t\}_{t=1}^T} \sum_{t=1}^{T} \beta^{t-1}\sqrt{c_t} \quad \text{s.t.} \quad \underbrace{W = \sum_{t=1}^{T} c_t}_{\text{Can't eat more than } W} \quad , \quad \underbrace{c_t \in \mathbb{N}_0 \ \forall t}_{\text{Can only eat "full" slices}}$$

Express with **Bellman equation** with state W_t and state transition for W_{t+1}

$$V_t(W_t) = \max_{c_t \in \mathbb{N}_0, c_t \leq W_t,} \{\sqrt{c_t} + \beta V_{t+1}(W_{t+1})\} \quad \text{s.t. } W_{t+1} = W_t - c_t$$

Exercise 1: Backwards Induction algorithm

1. In last period T, it is always optimal to eat what is left, i.e. $c_T^* = W_T$ and $\Rightarrow V_T(W_T) = \sqrt{W_T}$
2. Go to period $T-1$ and solve:

$$V_{T-1}(W_{T-1}) = \max_{c_{T-1}\in\mathbb{N}_0, c_{T-1}\leq W_{T-1}} \{\sqrt{c_{T-1}} + \beta V_T(\overbrace{W_{T-1} - c_{T-1}}^{=W_T})\}$$

Since $c_{T-1} \in \mathbb{N}_0$, we do a grid search, i.e. "guessing" each $c_{T-1} \leq W_{T-1}$ and "choosing" the one with the largest "value guess".

For period $T-2, T-3, \ldots, 1$: Do the same as above, given that we know the value function for all subsequent periods.

Exercise 2: Cake-eating in an infinite horizon

Exactly the same problem as in exercise 1, except we let $T = \infty$.

Need another solution method $\rightarrow$ Value Function Iteration (VFI)

Idea/Intuition: With an infinite horizon, we "dont care" whether we are in period t and have $W_t = 2$ slices left, or we are in period $t + 1$ and have $W_{t+1} = 2$. The specific period becomes irrelevant.

$\rightarrow$ Formulate Bellman equation without time subscripts

$$V(W) = \max_{c \in \mathbb{N}_0, c \leq W} \{\sqrt{c} + \beta V(W - c)\}$$

$\rightarrow$ Find fixed point in **Bellman operator** $V = \Gamma(V)$

If $\beta \in [0, 1)$ *and some technical stuff*, the fixed point is unique

Exercise 2: Value Function Iteration Algorithm

1. Make arbitrary guess on value function, e.g. $V^{guess}(W) = 0 \ \forall W$
2. Use $V^{guess}(W)$ on RHS of Bellman equation.

$$V(W) = \max_{c \in \mathbb{N}_0, c \leq W} \{\sqrt{c} + \beta V^{guess}(W - c)\}$$

3. Check convergence $\max\{|V(W) - V^{guess}(W)|\}$
4. Update $V^{guess}(W)$ with new $V(W)$
5. Keep iterating and check convergence below some threshold

Your time to shine!