Advanced Econometrics in Labor and IO
Week 4 - Single Agent Dynamic Discrete Choice

Hannes Ullrich

DIW Berlin and University of Copenhagen

DIW Office 5.2.020, email: hullrich@diw.de

19 May 2022, DIW

Organization

- Next session: Thursday, June 2, 14:15-17:00
- Problem set 4 due at beginning of class
- This problem set will be graded.
- We will:
  - (1) discuss Rust code / problem set
  - (2) briefly introduce 2-step estimation based on conditional choice probabilities (CCP)

Plan for today

- Why dynamics?
- Infinite horizon, discounted Markov decision processes (MDP)
- Dynamic programming
- Estimating a dynamic discrete choice problem: Rust (1987)

Why Dynamics

- Economic agents often
  - are forward-looking and
  - face some intertemporal tradeoff.
- Static models may not fit (consumer) behavior well, generate biased estimates, and lead to wrong conclusions (e.g. markups in merger analysis)
- We often intend to identify preference parameters $\rightarrow$ sensibly modeling actions and payoffs crucial
- Gowrisankaran and Rysman (2012) find price coefficients strongly biased towards zero in static estimation of demand for camcorders
- Hard to form predictions in dynamic context without specifying fundamental data generating process

Dynamics in IO

- Single-agent problems (games against nature)
  - Demand for durable goods
    $\rightarrow$ optimal stopping problem: product replacement. Labor: job search, accept offer / continue search
  - Stockpiling, Switching costs, Learning, Product search
  - Firm decisions under perfect competition or monopoly: pricing, revenue management, capacity investment, advertising, learning about demand, sales agent effort compensation

Dynamics in IO: high-low pricing

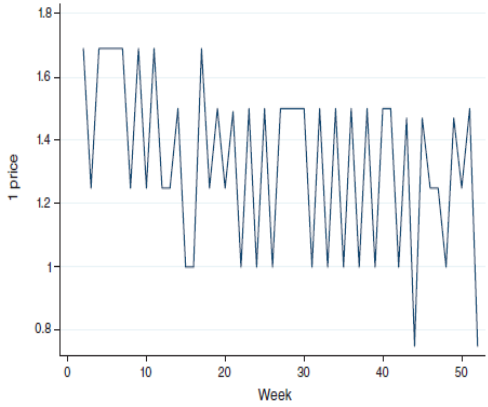

FIGURE 1. A TYPICAL PRICING PATTERN

Note: The figure presents the price of a two-liter bottle of Coke over 52 weeks in one store.

Dynamics in IO

- Single-agent problems (games against nature)
  - Demand for durable goods
    $\rightarrow$ optimal stopping problem: product replacement. Labor: job search, accept offer / continue search
  - Stockpiling, Switching costs, Learning, Product search
  - Firm decisions under perfect competition or monopoly: capacity investment, pricing, advertising
- Multi-agent problems (dynamic games)
  - Simplest game: Hyperbolic discounting, game against your future self
  - Advertising, entry/exit, innovation/R&D, investment, technology adoption, pricing and learning-by-doing
  - Collusion, Network effects
  - "Static-dynamic breakdown" assumption: separate static pricing game (per-period profit function) and dynamic interaction
- In Industrial Organization, dynamic analysis often cast in infinite-horizon problems

Rust (2019) on Dynamic Programming

- "...is a powerful tool for solving a wide class of sequential decision-making problems under uncertainty. [...] compute optimal decision rules that specify the best possible decision in any situation."

Rust (2019) on Dynamic Programming

- "...is a powerful tool for solving a wide class of sequential decision-making problems under uncertainty. [...] compute optimal decision rules that specify the best possible decision in any situation."
- "...offers substantial promise for improving decision making if we let go of the empirically untenable assumption of unbounded rationality and confront the challenging decision problems faced every day by individuals and firms."

Rust (2019) on Dynamic Programming

- "...is a powerful tool for solving a wide class of sequential decision-making problems under uncertainty. [...] compute optimal decision rules that specify the best possible decision in any situation."
- "...offers substantial promise for improving decision making if we let go of the empirically untenable assumption of unbounded rationality and confront the challenging decision problems faced every day by individuals and firms."
- "...has proved extremely successful as an academic modeling tool, but the formulation and solution of real-world decision problems as mathematical DP problems have proved far more challenging."

Rust (2019) on Dynamic Programming

- "The key reason for the limited number of real-world applications of DP is the difficulty in applying it to formulate and solve the highly complex, fuzzy, and poorly defined decision problems that individuals and firms typically face on a daily basis."

Rust (2019) on Dynamic Programming

- "The key reason for the limited number of real-world applications of DP is the difficulty in applying it to formulate and solve the highly complex, fuzzy, and poorly defined decision problems that individuals and firms typically face on a daily basis."
- "Perhaps the most painstaking task for an academic [...] trying to [...] recommend better decisions to [...] the individual or firm [...] is to understand the structure of the actor's decision problem. Calculating an optimal solution to the wrong objective [...] is like providing the right answer to the wrong question."

Rust (2019) on Dynamic Programming

- "The key reason for the limited number of real-world applications of DP is the difficulty in applying it to formulate and solve the highly complex, fuzzy, and poorly defined decision problems that individuals and firms typically face on a daily basis."
- "Perhaps the most painstaking task for an academic [...] trying to [...] recommend better decisions to [...] the individual or firm [...] is to understand the structure of the actor's decision problem. Calculating an optimal solution to the wrong objective [...] is like providing the right answer to the wrong question."
- "[Both standard human programming and AI (ML/RL) approaches] depend critically on the ability to learn the structure of the decision problem. [...] ML is not able to acquire this type of knowledge from scratch. [...] it is not clear that DP algorithms of any type will be useful to real-world DMs if they fail to grasp the actual problem and the constraints, opportunities, and challenges that real-world DMs actually confront."

Infinite horizon, discounted Markov decision processes (MDP)

Infinite horizon, discounted MDP

Elements

- Time $t = 0, 1, 2, ..., T$ with $T = \infty$ here
- A set of states $s_t \in S$
- A set of decisions $d_t \in D$, possibly with constraints $d_t \in D_t(s_t) \subseteq D$

Focus on discrete decision processes (DDP): D finite and countable

Infinite horizon, discounted MDP

- Given current state s_t and decision d_t, decision maker obtains single-period utility $u_t(s_t, d_t)$
- Transition probabilities: s follows a probability distribution $p_t(s_{t+1}|s_t, d_t)$
- "Markov" decision process: u_t and p_t depend only on *current* s_t and d_t
  - Current state contains all information about the past.
    $\rightarrow$ conditional on s_t, future values of s independent of full history.
- Structure of the problem defined: $[\beta, u, p, D]$

Infinite horizon, discounted MDP

- (Time-separable) total discounted utility

$$U(\mathrm{s}, \mathrm{d}) = \sum_{t=0}^{T} \left[\prod_{j=0}^{t-1} \beta_j(s_j, d_j) \right] u_t(s_t, d_t)$$

- Discount functions $\beta_t(s_t, d_t) \geq 0$
- (Single-period) Decision rule: associates current state with feasible decision: $d_t = \delta_t(s)$

Infinite horizon, discounted MDP

- Agent's optimization problem: sequence of decision rules,

$$\text{optimal policy} \quad \delta^* = (\delta_0, ..., \delta_T) \quad \text{to}$$

$$\max_{\delta=(\delta_0,...,\delta_T)} E_\delta\{U(\mathrm{s}, \mathrm{d})\}$$

- A policy δ induces a probability distribution on the sequence of states and decisions, $(s_t, d_t)_{t=0}^{T}$

  $\rightarrow$ Expectation wrt partially controlled stochastic process $\{s_t, d_t\}$

- "game against nature"

MDP: Stationarity assumption

- Assume utility u and transition probability p are stationary:
  1. u and p do not depend on t. For example, "getting tired" of a product over time cannot be modeled as $U_t(s,d) = -t$. Instead, $U(s,d) = -s$, where $s \in S = \{0,1,2,...\}$ and $p(s+1|s,d) = 1$, i.e. $s_{t+1} = s_t + 1$.
  2. Discount functions $\beta_t(s_t, d_t)$ assumed constant: $0 \leq \beta < 1$
- In infinite-horizon problems, stationarity leads to time invariant decision rule and value function (next slide): $\delta_t^\infty(s) = \delta(s)$, $V_t^\infty(s) = V(s)$
- Note: Finite-horizon problems are non-stationary.

Dynamic Programming (focus on infinite-horizon problems)

- Write agent's dynamic optimization problem as Bellman equation

$$V(s_t) = \max_{d_t \in D(s_t)} \left\{ u(s_t, d_t) + \beta \int V(s_{t+1}) p(s_{t+1}|s_t, d_t) \right\}$$

where optimal decision rule satisfies

$$\delta(s_t) = \arg\max_{d_t \in D(s_t)} \left\{ u(s_t, d_t) + \beta \int V(s_{t+1}) p(s_{t+1}|s_t, d_t) \right\}$$

- Value function $V(s_t)$: expected discounted value of utility assuming an optimal policy is followed in the future
  - valuation of the future consequences of current decisions

Bellman equation

- Splits problem into current payoff and continuation value
- Initial state s_t, current choice d_t
  $\rightarrow$ yields current reward $u(s_t, d_t)$, s_{t+1} according to $p(s_{t+1}|s_t, d_t)$,
  and maximum expected payoff, with s_{t+1} unknown, $\beta E[V(s_{t+1}|s_t, d_t)]$.
- Break down dynamic optimization problem into an (infinite) sequence of single-period decisions

Contraction mapping theorem / Banach fixed-point theorem:

- Under some regularity conditions (continuity, boundedness of $u(s,d), D(s)$),
- there is a map Γ such that $V = \Gamma V'$ where V is today's and V' tomorrow's value function, or:

$$\Gamma V(s) = \max_{d \in D(s)} \left\{ u(s,d) + \beta \int V(s') p(ds'|s,d) \right\}$$

- Γ has a unique fixed point V.
- Similarly, for the policy function, there exists a map such that $\delta = \Gamma_\delta V'$.
- Solution methods: value function iteration or policy function iteration.

Value function iteration

By the contraction mapping theorem,

- the sequence $V^{l+1} = \Gamma V^l$ converges to the infinite-horizon value function for any initial guess V^0.

Value function iteration

Step 1. Initial guess V^0 (e.g. 0), stopping criterion $\varepsilon > 0$

Step 2. For each state $i = 1, ..., s \in S$ compute

$$V_i^{l+1} = \max_{d_t \in D} \left\{ u(s_i, d_t) + \beta \sum_{j=1}^{s} V_j^l p(s_j | s_i, d_t) \right\}$$

Step 3. If $||V^{l+1} - V^l|| < \varepsilon$, go to step 4; else back to step 2.

Step 4. Compute final solution:

- $\delta^* = \Gamma_\delta V^{l+1}$, where Γ_δ is the arg max operator
- $u_i^* = u(s_i, \delta^*(s_i)), \quad i = 1, ..., s,$
- $V^* = (I - \beta p_{\delta^*})^{-1} u^*$ (Solution to $\underbrace{V}_{s \times 1} = \underbrace{u^*}_{s \times 1} + \beta \underbrace{p_{\delta^*}}_{s \times s} \underbrace{V}_{s \times 1}$)

Note: No use of policy function. Can we use it to improve convergence?

Policy function iteration

- Each iteration: new policy & value function (new policy used forever)
- Value function iteration assumes new policy only used for one period.

Step 1. Stopping criterion $\varepsilon > 0$ and either:
Initial guess V^0 and step 2 or initial guess δ^0 and step 3

Step 2. For each state $i = 1, ..., s \in S$ compute

$$\delta_i^{l+1} = \arg\max_{d_t \in D} \left\{ u(s_i, d_t) + \beta \sum_{j=1}^{s} V_j^l p(s_j | s_i, d_t) \right\}$$

Step 3. $u_i^{l+1} = u(s_i, \delta^{l+1}(s_i)), \quad i = 1, ..., s$

Step 4. $V^{l+1} = (I - \beta p_{\delta^{l+1}})^{-1} u^{l+1}$

Step 5. If $||V^{l+1} - V^l|| < \varepsilon$, stop; else back to step 2.

Estimating a dynamic discrete choice problem: Rust (1987)

Estimating a dynamic discrete choice problem: Rust (1987)

- Harold Zurcher (HZ): maintenance manager at Madison Metropolitan Bus Company
- Decision: replace old bus engines with new ones
- Trade-off: Minimize cost due to replacement vs. unexpected engine failures
  - engine replacement: large fixed costs but lower future maintenance costs
  - no replacement: avoid fixed costs, suffer higher future maintenance costs
- Solution to optimal stopping problem: critical cutoff mileage below which no replacement / above which replacement
  - Other examples: durable goods demand (replacement), job search (accept/reject job offers: reservation wage threshold)

Estimating a dynamic discrete choice problem: Rust (1987)

- Primitives to estimate:
  - expectations of future values of state variables (transition function)
  - marginal costs of regular maintenance
  - replacement costs
- Idea: Construct model predicting time and mileage of engine replacement
- Using model predictions (given parameters θ), find parameters that fit data
- Model and parameters can then be used to compute policy counterfactuals, e.g. for an exogenous change in replacement costs due to subsidies

Rust 1987: Data

Rust observed 104 busses over time (ten years). On each bus:

- monthly mileage
- date, mileage, list of components repaired/replaced when bus in garage
- on average, engines replaced after 5 years ($>$ 200K miles)

Rust 1987: Basic setup

- Each month t, HZ decides whether to replace engine.

$$\text{Observed decision:} \qquad i_t = \begin{cases} 1, & \text{if engine replacement in month } t, \\ 0, & \text{otherwise.} \end{cases}$$

- Observed state variable x_t: accumulated engine mileage at t
- Mileage evolves exogenously according to Markov process $p(x_{t+1}|x_t, i, \theta)$
- HZ treats each bus independently: no joint replacement and utilization decisions

Rust 1987: Basic setup

- Single-period reward function of decision i_t in state x_t:

$$u(x_t, i_t, \theta) = \begin{cases} -c(x_t, \theta), & \text{if } i_t = 0, \\ -RC - c(0, \theta), & \text{if } i_t = 1. \end{cases}$$

where

- $c(x_t, \theta)$: cost of operating a bus (maintenance, social costs of breakdowns)
- Rust used several functional forms for the cost function, for example linear: $c(x_t, \theta) = \theta_{11} x$.
- RC: engine replacement cost.

Rust 1987: Key assumptions for estimation

- Observed and unobserved state variables $s = (x, \epsilon)$
- Unobserved state variables $\epsilon_t = \{\epsilon_t(i) | i \in D(x_t)\} = \{\epsilon_t(0), \epsilon_t(1)\}$
- Additive separability (AS) of ϵ_t: $u(x_t, i, \theta) + \epsilon_t(i)$
- ϵ component of utility $\rightarrow$ "structural errors"
- ϵ rationalizes observations where identical x associated with differing i

Rust 1987: Dynamic problem

- HZ's decision: maximize expected present discounted sum of future utilities
- β unidentified (Magnac and Thesmar 2002)
- Intuition: Difficult to identify β apart from fixed costs. If HZ myopic ($\beta \approx 0$) and replacement cost RC low, observed decisions similar to HZ forward-looking ($\beta \approx 1$) and RC large.
- Rust (1987) assumes $\beta = 0$ and $\beta = 0.9999$

Rust 1987: Dynamic problem

- Infinite-horizon MDP written as Bellman equation, with unobserved state ϵ,

$$V(x_t, \epsilon_t) = \max_{i \in D(x_t)} \{u(x_t, i, \theta) + \epsilon_t(i) + \beta EV(x_t, \epsilon_t, i, \theta)\}$$

where

$$EV(x_t, \epsilon_t, i) = \int_y \int_\eta V(y, \eta) p(y, \eta | x_t, \epsilon_t, i, \theta)$$

- Optimal policy defined by

$$\delta(x_t, \epsilon_t, \theta) = \arg\max_{i \in D(x_t)} \{u(x_t, i, \theta) + \epsilon_t(i) + \beta EV(x_t, \epsilon_t, i, \theta)\}$$

Rust 1987: Key assumptions for estimation

- Conditional independence (CI):

$$p(x_{t+1}, \epsilon_{t+1} | x_t, \epsilon_t, i, \theta) = q(\epsilon_{t+1} | x_{t+1}, \theta) p(x_{t+1} | x_t, i_t, \theta)$$

  - Transition probability of x is unaffected by ϵ.
  - Any dependence between ϵ_t and ϵ_{t+1} entirely transmitted through x_{t+1}
  - Intuition: Rule out serially persistent forms of unobserved heterogeneity

- With CI, can write Bellman equation as

$$V(x_t, \epsilon_t, \theta) = \max_{i \in D(x_t)} \{u(x_t, i, \theta) + \epsilon_t(i) + \beta EV(x_t, i, \theta)\}$$

- $\rightarrow EV(x_t, i, \theta)$ instead of $EV(x_t, \epsilon_t, i, \theta)$

Rust 1987: Key assumptions for estimation

- Using AS and CI, we can integrate out ϵ_t as in static discrete choice models:

$$P(i_t|x_t,\theta) = \int\int \mathbb{1}(V^j(x_t,\theta)+\epsilon_{jt} \leq V^i(x_t,\theta)+\epsilon_{it}) p(\epsilon_{jt},\epsilon_{it}|x_t,\theta)$$

where the choice-specific value function, net of $\epsilon_t(i)$,

$$V^i(x_t,\theta) = u(x_t,i_t,\theta) + \beta E[V(x_{t+1},\epsilon_{t+1},\theta)|x_t,i_t,\theta].$$

Rust 1987: Key assumptions for estimation

- Assuming ϵ distributed i.i.d. extreme value type 1 (EV), obtain dynamic version of logit choice probabilities, conditional choice probabilities (CCP):

$$P(i_t|x_t,\theta) = \frac{exp\{V^i(x_t,\theta)\}}{\sum_{i\in\{0,1\}} exp\{V^i(x_t,\theta)\}},$$

- $V^0(x_t,\theta)$, $V^1(x_t,\theta)$: values of no replacement / replacement

Remark: Static vs. dynamic discrete choice models

- With choice-specific value functions, the predictions of a dynamic discrete choice model can be expressed as for the static discrete choice model.
- In static model, specify a parametric choice-specific utility function, $u_j(x, \theta)$
- Unlike $u_j(x, \theta)$, $V^i(x_t, \theta)$ not a structural object; solution of a dynamic decision process that depends on the model primitives.
- Estimation requires a routine that nests this solution $\rightarrow$ nested fixed point algorithm

Rust 1987: Nested fixed point algorithm

- Inner loop: Solve dynamic optimization problem
- Outer loop: Search parameters that maximize the likelihood function (match the model to the observed data)

Rust 1987: NFP inner loop

- Numerical solution: value function $V(x, \epsilon, \hat{\theta})$
- Avoid computing value functions at (unobserved) state variables ϵ_0 and ϵ_1?
- With CI: Iterate over *expected* value function $E[V(x, i, \theta)]$
- Use extreme value distribution assumption to simplify $E[V(x, i, \theta)]$ further

- Recall Bellman equation under CI

$$V(x_t, \epsilon_t, \theta) = \max_{i \in D(x_t)} \left\{ u(x_t, i, \theta) + \epsilon_t(i) + \beta \int_y \int_\eta V(y, \eta) p(y, \eta | x_t, i, \theta) \right\}$$

and choice-specific value function

$$V^i(x_t, \theta) = u(x_t, i_t, \theta) + \beta E[V(x_{t+1}, \epsilon_{t+1}, \theta) | x_t, i_t, \theta]. \quad (1)$$

Rust 1987: NFP inner loop

- Value function for Zurcher's binary choice problem:

$$V(x_t, \epsilon_t, \theta) = \max_{i_t} \left\{ V^0(x_t, \theta) + \epsilon_{0t}, V^1(x_t, \theta) + \epsilon_{1t} \right\} \tag{2}$$

- Plugging equation (2) at $t+1$ in the choice-specific value functions in (1):

$$V^i(x_t, \theta) = \tag{3}$$
$$u(x_t, i_t, \theta) + \beta E \left[\max_{i_{t+1}} \left\{ V^0(x_{t+1}, \theta) + \epsilon_{0t+1}, V^1(x_{t+1}, \theta) + \epsilon_{1t+1} \right\} | x_t, i_t, \theta \right]$$

- Simplifying further, extreme value error terms yield closed-form expression for the expectation (wrt ϵ) of the maximum, the "log sum":

$$E[\max\{\delta_1 + \epsilon_1, ..., \delta_J + \epsilon_J\}] = 0.5772 + \ln\left(\sum_{j=1}^{J} exp(\delta_j)\right)$$

use in equation (3) so that

$$V^0(x_t, \theta) = u(x_t, 0, \theta) + \beta E\left[0.5772 + \ln\left\{e^{V^0(x_{t+1}, \theta)} + e^{V^1(x_{t+1}, \theta)}\right\} | x_t, 0, \theta\right],$$
$$V^1(x_t, \theta) = u(x_t, 1, \theta) + \beta E\left[0.5772 + \ln\left\{e^{V^0(x_{t+1}, \theta)} + e^{V^1(x_{t+1}, \theta)}\right\} | x_t, 1, \theta\right]$$

- Contraction mapping over function $E[\cdot]$
- After convergence, compute $V^0(x_t, \theta)$, $V^1(x_t, \theta)$ once more.

Rust 1987: outer loop

- Observe conditional choice probabilities
- Estimate parameters by maximum likelihood
- Using CI, likelihood function

$$\begin{aligned}\mathcal{L}(\theta) &= Pr(x_1, ..., x_T, i_1, ..., i_T | x_0, i_0, \theta) \\ &= \prod_{t=1}^{T} Pr(i_t | x_t, \theta) p(x_t | x_{t-1}, i_{t-1}, \theta)\end{aligned}$$

- $Pr(i_t|x_t, \theta)$: given by the solution to the dynamic programming problem,
- $p(x_t|x_{t-1}, i_{t-1}, \theta)$: transition probabilities estimated in (separate) first step.
  - Rational expectations assumption: subjective beliefs about $p(x_{t+1}|x_t, i)$ coincide with population probability measures.

Issues

- Serially correlated unobservables (ϵ)
- Non-additive shocks (ϵ)
- Persistent unobserved heterogeneity
- $\rightarrow$ AS, CI, iid errors are helpful but often strong assumptions
- Rational expectations. Can combine with subjective expectations data (Manski, Ecta 2004).