Single-agent dynamic optimization models

C.Conlon - Adapted from M. Shum and Paul Scott

Grad IO

October 27, 2017

Single-agent dynamic optimization models

Setup in Rust:

- Harold Zurcher manages a bus depot in Madison, WI
- Each week he must decide to continue with the current engine $i_t = 0$ and pay $c(x_t)$ or overhaul the engine $i_t = 1$ and pay fixed cost RC.
- His goal is to minimize long run average cost (discounted).

Rust (1987)

The agent makes a series of decisions $i_t, i_{t+1}, \ldots \in \{0, 1\}$.

$$\max_{\{i_1, i_2, i_3, \cdots, i_t, i_{t+1}, \cdots\}} E \sum_{t=1}^{\infty} \beta^{t-1} \pi(x_t)$$

$$\pi\left(i_t, x_t\right) = \begin{cases} -c\left(x_t\right) & \text{if } i_t = 0 \\ -c\left(0\right) - RC & \text{if } i_t = 1 \end{cases}$$

- When costs are increasing in mileage (x_{it}) then this is an optimal stopping problem.
- We know from Stokey Lucas Prescott (1989) that solutions to optimal stopping problems are characterized by critical value or cutoff rule which we call x^*.

Rust (1987)

- For those who took first year macro, this problem is trivial.
- But Rust had a different goal in mind
  - Can we use the observed decisions of the agent to identify the primitives of $\pi(i_t, x_t)$?
  - Instead of just optimizing Bellman's equation, can we actually estimate parameters?
- Need to make two modifications:
  - Parameters for $c(x_t, \theta_1)$ and RC
  - Full support errors ε_t, otherwise can't justify why $i(x) = 1$ and $i(x') = 0$ if $x' > x$ at some point in the data.

Model: Modified Payoff Function

- per-period profit function:

$$\pi\left(i_t, x_t, \theta_1\right) = \begin{cases} -c\left(x_t, \theta_1\right) + \varepsilon_t\left(0\right) & \text{if } i_t = 0 \\ -\left(RC - c\left(0, \theta_1\right)\right) + \varepsilon_t\left(1\right) & \text{if } i_t = 1 \end{cases}$$

  where

  - $c\left(x_t, \theta_1\right)$ - regular maintenance costs (including expected breakdown costs),
  - RC - the net costs of replacing an engine,
  - ε - payoff shocks.

- x_t is the observed state variable known to both agent and econometrician
- ε is the unobserved state variable known only to agent

Model: Bellman

Can define value function using Bellman equation:

$$\begin{aligned} V_\theta(x_t, \varepsilon_t) &= \max_i \left[\pi(i, x_t, \theta_1) + \beta E V_\theta(x_t, \varepsilon_t, i)\right] \\ EV_\theta(x_t, \varepsilon_t, i_t) &= \int V_\theta(x_{t+1}, \varepsilon_{t+1})\, p(x_{t+1}, \varepsilon_{t+1} | x_t, \varepsilon_t, i_t, \theta_2, \theta_3) \end{aligned}$$

It is helpful to define the choice-specific value function:

$$\begin{aligned} \tilde{V}_\theta(x_t, \varepsilon_t, i_t) &= \begin{cases} \tilde{u}(x_t, 1; \theta_1) + \beta E V(0, \varepsilon') + \varepsilon_t(1) & \text{if } i_t = 1 \\ \tilde{u}(x_t, 0; \theta_1) + \beta E_{x', \varepsilon' | x_t, \varepsilon_t, i_t} V(x', \varepsilon') + \varepsilon_t(0) & \text{if } i_t = 0 \end{cases} \\ V_\theta(x_t, \varepsilon_t) &= \max\left\{\tilde{V}_\theta(x_t, \varepsilon_t, 1) + \varepsilon_t(1), \tilde{V}_\theta(x_t, \varepsilon_t, 0) + \varepsilon_t(0)\right\} \end{aligned}$$

- θ_1 - parameters of cost function
- θ_2 - parameters of distribution of ε (these will be assumed/normalized away)
- θ_3 - parameters of x-state transition function
- RC - replacement cost
- discount factor β will be imputed (more on this later)

Assumptions

First Order Markov

Conditional Independence Assumption

The transition density of the controlled process $\{x_t, \varepsilon_t\}$ factors as:

$$p\left(x_{t+1}, \varepsilon_{t+1} | x_t, \varepsilon_t, i_t, \theta_2, \theta_3\right) = q\left(\varepsilon_{t+1} | x_{t+1}, \theta_2\right) p\left(x_{t+1} | x_t, i_t, \theta_3\right)$$

- CI assumption is very powerful: it means we don't have to treat ε_t as a state variable, which would be very difficult since it's unobserved.

IID Type I EV Assumption

For now we will also assume that $q\left(\varepsilon_{t+1} | x_{t+1}, \theta_2\right) = q\left(\varepsilon_{t+1}\right)$ and q is an IID Type I Extreme Value (logit) distribution.

- This is not required for identification but commonly employed to simplify estimation.
- Rust assumes that mean is 0 and variance is $\pi^2/6$.

Implications

Given the assumptions:

$$\begin{aligned} V_\theta(x_t, \varepsilon_t) &= \max\left\{\tilde{V}_\theta(x_t, \varepsilon_t, 1) + \varepsilon_t(1), \tilde{V}_\theta(x_t, \varepsilon_t, 0) + \varepsilon_t(0)\right\} \\ Pr(i_t = 1|x_t, \varepsilon_t, \theta) &= Pr\left(\varepsilon_t(1) - \varepsilon_t(0) \geq \tilde{V}_\theta(x_t, \varepsilon_t, 0) - \tilde{V}_\theta(x_t, \varepsilon_t, 1)\right) \\ &= \frac{\exp[\tilde{V}_\theta(x_t, \varepsilon_t, 1)]}{\exp[\tilde{V}_\theta(x_t, \varepsilon_t, 0)] + \exp[\tilde{V}_\theta(x_t, \varepsilon_t, 1)]} \end{aligned}$$

This expression is logit-like. Recall the static logit:

$$Pr(i_t = 1|x_t, \varepsilon_t, \theta) = \frac{\exp[u_\theta(x_t, 1)]}{\exp[u_\theta(x_t, 0)] + \exp[u_\theta(x_t, 1)]}$$

Problem:

- The general problem is that we can often write an observationally equivalent myopic model
- In general without additional (parametric) restrictions we are under identified (especially w.r.t β).

Rust (1987)

Likelihood function for a single bus:

$$
\begin{aligned}
& l(x_1, \cdots, x_T, i_t, \cdots, i_T | x_0, i_0; \theta) \\
= & \prod_{t=1}^{T} Prob(i_t, x_t | x_0, i_0, \cdots, x_{t-1}, i_{t-1}; \theta) \\
= & \prod_{t=1}^{T} Prob(i_t, x_t | x_{t-1}, i_{t-1}; \theta) \\
= & \prod_{t=1}^{T} Prob(i_t | x_t; \theta) \cdot \prod_{t=1}^{T} Prob(x_t | x_{t-1}, i_{t-1}; \theta_3).
\end{aligned}
$$

The third line arises from the Markovian feature of the problem, and the last equality arises due to the conditional independence assumption.

Rust (1987)

Log likelihood is additively separable in the two components:

$$\log l(\theta) = \sum_{t=1}^{T} \log Prob(i_t|x_t;\theta_1) + \sum_{t=1}^{T} \log Prob(x_t|x_{t-1}, i_{t-1};\theta_3).$$

Give the factorization of the likelihood function above, we can estimate in two steps...

Step 1: Estimate Markov TPM

- Estimate θ_3, the parameters of the Markov transition probabilities for mileage, conditional on non-replacement of engine (i.e. $i_t = 0$)
- Recall that $x_{t+1} = 0$ if $i_t = 1$

We assume a discrete distribution for $\triangle x_t \equiv x_{t+1} - x_t$, the incremental mileage between any two periods:

$$\triangle x_t = \begin{cases} [0, 5000) & \text{w/prob } p \\ [5000, 10000) & \text{w/prob } q \\ [10000, \infty) & \text{w/prob } 1 - p - q \end{cases}$$

so that $\theta \equiv \{p, q\}$, with $0 < p, q < 1$ and $p + q < 1$.

- $\hat{\theta}_3$ estimated by empirical frequencies: $\hat{p} = \text{freq}\{\triangle x_t \in [0, 5000)\}$, etc.
- Note: this does not require the behavioral model!

Step #2: Estimate Structural Parameters of Cost Function

Start by treating $(\beta, \hat{\theta}_3)$ as given:

1. Fix a guess of (RC, θ_1) the remaining parameters.
2. Iterate on the Bellman Operator for $(\beta, \theta_1, \theta_3, RC)$ using Value Function Iteration to get $V^*(x, \varepsilon)$ or $\tilde{V}^*(x, \varepsilon)$.
3. Calculate conditional choice probabilities (CCPs):

$$Pr(i_t = 1|x_t, \varepsilon_t, \theta) = \frac{\exp[\tilde{V}_\theta(x_t, \varepsilon_t, 1)]}{\exp[\tilde{V}_\theta(x_t, \varepsilon_t, 0)] + \exp[\tilde{V}_\theta(x_t, \varepsilon_t, 1)]}$$

4. Evaluate the log-likelihood:

$$\log l(\theta) = \sum_{t=1}^{T} \log \Pr(i_t|x_t; \theta_1, RC) + \underbrace{\sum_{t=1}^{T} \log \Pr(x_t|x_{t-1}, i_{t-1}; \hat{\theta}_3)}_{\text{Can Ignore! Why?}}$$

Solve via MLE. This is the Nested Fixed Point algorithm.

Computational Details

That looked easy, except that:

- I never really showed you how to recover $\tilde{V}_\theta(x, i)$.
- Directly iterating on Bellman's operator requires keeping track of ε's which are: (1) unobserved to you the econometrician and (2) continuous and full support (not a discrete grid).
  - AKA a big pain.
- You may (or may not) have learned some tricks for solving Bellman equations in Macro that you could apply here: VFI, Policy Iteration (PI), Howard's Policy Improvement, etc.
  - None of that really tells us how to deal with ε's.

Rust's Trick

- Rust has a nice trick that let's us work with a new function $EV_\theta(x, i)$ instead of $V_\theta(x, i, \varepsilon)$ we call this the ex ante or expected value function.

$$EV(x, i) \equiv E_{x', \varepsilon'|x,i} V(x', \varepsilon'; \theta)$$

- In words $EV_\theta(x, i)$ says at time $t-1$ what is the expected value of $V_\theta(x_t, \varepsilon_t)$ [eq 4.14].

$$EV(x, i) = \int_y \log \left\{ \sum_{j \in C(y)} \exp[u(y, j; \theta) + \beta EV(y, j)] \right\} p(dy|x, i)$$

- Here x, i denotes the *previous* period's mileage and replacement choice, and y, j denote the *current* period's mileage and choice.

Derivation of Rust's Trick

This ex ante value function can be derived from Bellman's equation:

$$
\begin{aligned}
V(y,\varepsilon;\theta) =& \max_{j\in 0,1}[u(y,j;\theta)+\varepsilon+\beta EV(y,j)] \\
\Longrightarrow& E_{y,\varepsilon}[V(y,\varepsilon;\theta)|x,i] \equiv EV(x,i;\theta) \\
=& E_{y,\varepsilon|x,i}\left\{\max_{j\in 0,1}[u(y,j;\theta)+\varepsilon+\beta EV(y,j)]\right\} \\
=& E_{y|x,i}E_{\varepsilon|x,i}\left\{\max_{j\in 0,1}[u(y,j;\theta)+\varepsilon+\beta EV(y,j)]\right\} \\
=& E_{y|x,i}\log\left\{\sum_{j=0,1}\exp[u(y,j;\theta)+\beta EV(y,j)]\right\} \\
=& \int_y \log\left\{\sum_{j=0,1}\exp[u(y,j;\theta)+\beta EV(y,j)]\right\} p(dy|x,i).
\end{aligned}
$$

Value Function Iteration

1. Start with an initial guess at $\tau = 0$ for $EV_\theta^\tau(x,i)$. A common guess is $EV_\theta^\tau(x,i) = 0$ for all (x,i)
2. Iterate Bellman Operator

$$EV_\theta^{\tau+1}(x,i) = \int_y \log\left\{\sum_{j=0,1} \exp[u(y,j;\theta) + \beta EV^\tau(y,j)]\right\} p(dy|x,i).$$

   with $p(dy|x,i;\hat{\theta}_3)$ estimated in Step 1.
3. Compare $\epsilon(\tau) \equiv \sup_{(x,i)} |EV_\theta^{\tau+1}(x,i) - EV_\theta^\tau(x,i)|$ to ϵ^{tol}. If $\epsilon(\tau) \leq \epsilon^{tol}$ then stop.

See my notes on Numerical Dynamic Programming for more details.

Value Function Iteration: Bounds

- Suppose we set $V_0 = 0$ then the value function iteration approach is just like solving the finite horizon problem by backward induction.
- The CMT guarantees consistency at a geometric rate or linear convergence with modulus β
- We can derive an expression for the number of steps we need to get an ϵ-approximation.

$$T(\epsilon, \beta) = \frac{1}{|\log(\beta)|} \log\left(\frac{1}{(1-\beta)\epsilon}\right)$$

- This tells us that when $\beta \to 1$ that VFI gets very very slow.

Estimates

TABLE IX

STRUCTURAL ESTIMATES FOR COST FUNCTION $c(x, \theta_1) = .001\theta_{11}x$

FIXED POINT DIMENSION = 90

(Standard errors in parentheses)

Parameter		Data Sample			Heterogeneity Test	
Discount Factor	Estimates/ Log-Likelihood	Groups 1, 2, 3 3864 Observations	Group 4 4292 Observations	Groups 1, 2, 3, 4 8156 Observations	LR Statistic ($df = 4$)	Marginal Significance Level
$\beta = .9999$	RC	11.7270 (2.602)	10.0750 (1.582)	9.7558 (1.227)	85.46	1.2E−17
	θ_{11}	4.8259 (1.792)	2.2930 (0.639)	2.6275 (0.618)		
	θ_{30}	.3010 (.0074)	.3919 (.0075)	.3489 (.0052)		
	θ_{31}	.6884 (.0075)	.5953 (.0075)	.6394 (.0053)		
	LL	−2708.366	−3304.155	−6055.250		
$\beta = 0$	RC	8.2985 (1.0417)	7.6358 (0.7197)	7.3055 (0.5067)	89.73	1.5E−18
	θ_{11}	109.9031 (26.163)	71.5133 (13.778)	70.2769 (10.750)		
	θ_{30}	.3010 (.0074)	.3919 (.0075)	.3488 (.0052)		
	θ_{31}	.6884 (.0075)	.5953 (.0075)	.6394 (.0053)		
	LL	−2710.746	−3306.028	−6061.641		
Myopia test:	LR Statistic ($df = 1$)	4.760	3.746	12.782		
$\beta = 0$ vs. $\beta = .9999$	Marginal Significance Level	0.0292	0.0529	0.0035		

Discount factor

- While Rust finds a better fit for $\beta = .9999$ than $\beta = 0$, he finds that high levels of β basically lead to the same level of the likelihood function.
- Furthermore, the discount factor is non-parametrically non-identified. Note: He loses ability to reject $\beta = 0$ for more flexible cost function specifications.

Discount factor

TABLE VIII

SUMMARY OF SPECIFICATION SEARCH[a]

Cost Function	Bus Group 1, 2, 3	4	1, 2, 3, 4
Cubic $c(x, \theta_1) = \theta_{11}x + \theta_{12}x^2 + \theta_{13}x^3$	Model 1 −131.063 −131.177	Model 9 −162.885 −162.988	Model 17 −296.515 −296.411
quadratic $c(x, \theta_1) = \theta_{11}x + \theta_{12}x^2$	Model 2 −131.326 −131.534	Model 10 −163.402 −163.771	Model 18 −297.939 −299.328
linear $c(x, \theta_1) = \theta_{11}x$	Model 3 −132.389 −134.747	Model 11 −163.584 −165.458	Model 19 −300.250 −306.641
square root $c(x, \theta_1) = \theta_{11}\sqrt{x}$	Model 4 −132.104 −133.472	Model 12 −163.395 −164.143	Model 20 −299.314 −302.703
power $c(x, \theta_1) = \theta_{11}x^{\theta_{12}}$	Model 5[b] N.C. N.C.	Model 13[b] N.C. N.C.	Model 21[b] N.C. N.C.
hyperbolic $c(x, \theta_1) = \theta_{11}/(91 - x)$	Model 6 −133.408 −138.894	Model 14 −165.423 −174.023	Model 22 −305.605 −325.700
mixed $c(x, \theta_1) = \theta_{11}/(91 - x) + \theta_{12}\sqrt{x}$	Model 7 −131.418 −131.612	Model 15 −163.375 −164.048	Model 23 −298.866 −301.064
nonparametric $c(x, \theta_1)$ any function	Model 8 −110.832 −110.832	Model 16 −138.556 −138.556	Model 24 −261.641 −261.641

[a] First entry in each box is (partial) log likelihood value ℓ^2 in equation (5.2)) at $\beta = .9999$. Second entry is partial log likelihood value at $\beta = 0$.

[b] No convergence. Optimization algorithm attempted to drive $\theta_{12} \to 0$ and $\theta_{11} \to +\infty$.

Application

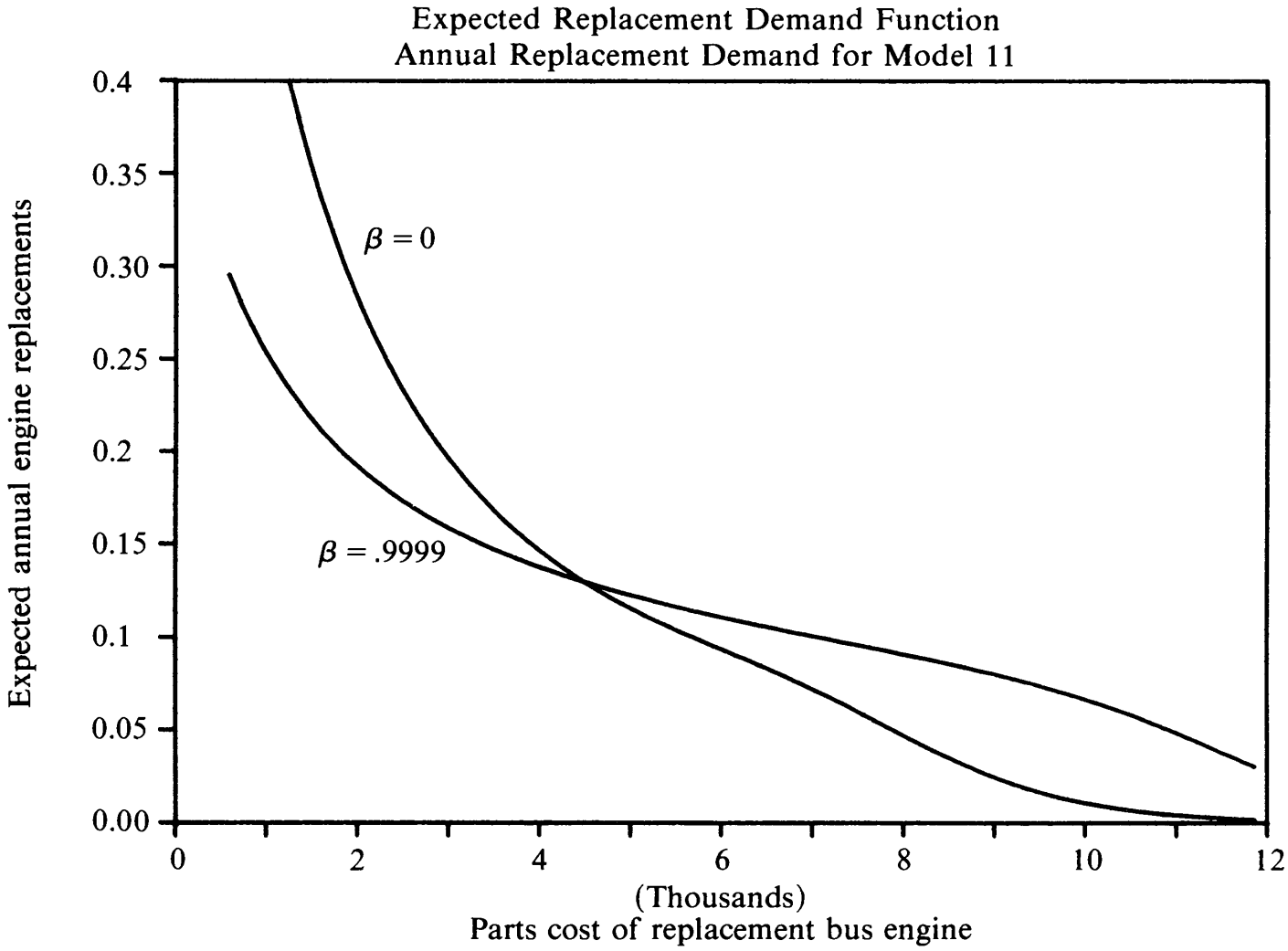

FIGURE 7

Motivation

- In Rust, we started with a guess of parameters θ, iterated on the Bellman operator to get $V(x, i, \theta)$ and then constructed CCP's $P(i|x, \theta)$.
- A disadvantage of Rust's approach is that it can be computationally intensive
  - With a richer state space, solving value function (inner fixed point) can take a very long time, which means estimation will take a very, very long time.
- Hotz and Miller's idea is to use observable data to form an estimate of (differences in) the value function from conditional choice probabilities (CCP's)
  - We observe $P(i|x)$ directly in the data!
- The central challenge of dynamic estimation is computing continuation values. In Rust, they are computed by solving the dynamic problem. With Hotz-Miller (or the CCP approach more broadly), we "measure" continuation values using a function of CCP's.

Rust's Theorem 1: Values to CCP's

- In Rust (1987), CCPs can be derived from the value function:

$$p_j(x) = \frac{\partial}{\partial \pi_j(x)} W\left(\pi(x) + \beta E\left[V\left(x'\right)|x, j\right]\right)$$

where $W(u) = \int \max_j \{u_j + \varepsilon_j\} dG(\varepsilon)$ is the surplus function.

- For the logit case:

$$p_j(x) = \frac{\exp\left(v_j(x)\right)}{\sum_{j' \in J} \exp\left(v_{j'}(x)\right)}$$

where the choice specific value function for action j in state x is

$$v_j(x) \equiv \pi_j(x) + \beta E\left[V\left(x'\right)|x, j\right]$$

HM's Proposition 1: CCP's to Values

- Notice that CCP's are unchanged by subtracting some constant from every conditional (choice-specific) value function. Thus, consider

$$D_{j,0}v\left(x\right) \equiv v_j\left(x\right) - v_0\left(x\right)$$

  where 0 denotes some reference action.

- Let $Q : \mathbb{R}^{|\mathbf{I}|-1} \to \Delta^{|\mathbf{I}|}$ be the mapping from the differences in conditional (choice-specific) values to CCP's.

- Note: we're taking for granted that the distribution of ε is identical across states, otherwise Q would be different for different x.

Hotz-Miller Inversion

Q is invertible.

HM inversion with logit errors

- Again, let's consider the case of where ε is i.i.d type I EV.
- Expression for CCP's:

$$p_j(x) = \frac{\exp(v_j(x))}{\sum_{j' \in \mathbf{J}} \exp(v_j(x))}.$$

- The HM inversion follows by taking logs and differencing across actions:

$$\ln p_j(x) - \ln p_0(x) = v_j(x) - v_0(x)$$

- Thus, in the logit case (this looks a lot like Berry (1994)):

$$Q_j^{-1}(p) = \ln p_j - \ln p_0$$

- From now on, I will use $\phi(p)$ to denote $Q^{-1}(p)$.

Arcidiacono and Miller's Lemma

An equivalent result to the HM inversion was introduced by Arcidiacono and Miller (2011). It's worth introducing here because it makes everything from now on much simpler and more elegant.

Arcidiacono Miller Lemma

For any action-state pair (a, x), there exists a function ψ such that

$$V(x) = v_a(x) + \psi_a(p(x))$$

Proof:

$$\begin{aligned} V(x) &= \int \max_j \{v_j(x) + \varepsilon_j\} dG(\varepsilon_j) \\ &= \int \max_j \{v_j(x) - v_a(x) + \varepsilon_j\} dG(\varepsilon_j) - v_a(x) \\ &\quad \int \max_j \{\phi_{ja}(p(x)) + \varepsilon_j\} dG(\varepsilon_j) - v_a(x) \end{aligned}$$

Letting $\psi_a(p(x)) = \int \max_j \{\phi_{ja}(p(x)) + \varepsilon_j\} dG(\varepsilon_j)$ completes the proof

Important relationships

- The Hotz-Miller Inversion allows us to map from CCP's to differences in conditional (choice-specific) value functions:

$$\phi_{ja}\left(p\left(x\right)\right) = v_j\left(x\right) - v_a\left(x\right)$$

- The Arcidiacono and Miller Lemma allows us to relate ex ante and conditional (choice specific) value functions:

$$V\left(x\right) = v_j\left(x\right) + \psi_j\left(p\left(x\right)\right)$$

- For the logit case:

$$\begin{aligned}\phi_{ja}\left(p\left(x\right)\right) &= \ln\left(p_j\left(x\right)\right) - \ln\left(p_a\left(x\right)\right)\\ \psi_j\left(p\left(x\right)\right) &= -\ln\left(p_j\left(x\right)\right) + \gamma\end{aligned}$$

where γ is Euler's gamma

Estimation example: finite state space I

- Let's suppose that X is a finite state space. Furthermore, let's "normalize" the payoffs for a reference action $\pi_0(x) = 0$ for all x.
  - We'll discuss soon whether this should really be called a "normalization"
- Using vector notation, recall the definition of the choice-specific value function for the reference action:

$$\begin{aligned} v_0 &= \pi_0 + \beta F_0 V \\ v_0 &= \beta F_0 V \end{aligned}$$

- Using the Arcidiacono-Miller Lemma,

$$\begin{aligned} V - \psi_0(p) &= \beta F_0 V \\ &\Rightarrow \\ V &= (I - \beta F_0)^{-1} \psi_0(p) \end{aligned}$$

Estimation example: finite state space II

- Now we have an expression for the ex ante value function that only depends on objects we can estimate in a first stage:

$$V = (I - \beta F_0)^{-1} \psi_0 (p)$$

- To estimate the utility function for the other actions,

$$\begin{aligned} v_j &= \pi_j + \beta F_j V \\ V - \psi_j (p) &= \pi_j + \beta F_j V \\ \pi_j &= -\psi_j (p) + (I - \beta F_j) V \\ \pi_j &= -\psi_j (p) + (I - \beta F_j) (I - \beta F_0)^{-1} \psi_0 (p) \end{aligned}$$

Identification of Models I

- If we run through the above argument with π_0 fixed to an arbitrary vector $\widetilde{\pi}_0$ rather than 0, we will arrive at the following:

$$\pi_j = A_a \widetilde{\pi}_0 + b_j$$

where A_a and b_a depend only on things we can estimate in a first stage:

$$\begin{aligned} A_j &= (1 - \beta F_j)(1 - \beta F_0)^{-1} \\ b_j &= A_j \psi_0(p) - \psi_j(p) \end{aligned}$$

- We can plug in any value for $\widetilde{\pi}_0$, and each value will lead to a different utility function (different values for π_j). Each of those utility functions will be perfectly consistent with the CCP's we observe.

Identification of Models II

- Another way to see that the utility function is under-identified: If there are $|X|$ states and $|J|$ actions, the utility function has $|X|\,|J|$ parameters. However, there are only $|X|\,(|J|-1)$ linearly independent choice probabilities in the data, so we have to restrict the utility function for identification.
- Magnac and Thesmar (2002) make this point as part of their broader characterization of identification of DDC models. Their main result says that we can specify a vector of utilities for the reference action $\widetilde{\pi}$, a distribution for the idiosycratic shocks G, and a discount factor, and we will be able to find a model rationalizing the CCPs that features $(\widetilde{\pi}, \beta, G)$.

Identification of Counterfactuals

- Note that imposing a restriction like $\forall x : \pi_0(x) = 0$ is NOT a normalization in the traditional sense. If we were talking about a static normalization, each x would represent a different utility function, and $\pi_0(x) = 0$ would simply be a level normalization. However, in a dynamic model, the payoffs in one state affect the incentives in other states, so this is a substantive restriction.

- What is less clear *a priori* is whether these restrictions matter for counterfactuals. It turns out that some (but not all!) counterfactuals ARE identified, in spite of the under-identification of the utility function. What this means is that whatever value $\widetilde{\pi}_0$ we impose for the reference action, the model will not only rationalize the observed CCP's but also predict the same counterfactual CCP's. Kalouptsidi, Scott, and Souza-Rodrigues (2016) sort out when counterfactuals of DDC models are identified and when they are not.

Extensions

- Just like we can write the BLP (1995) problem as an MPEC, we can do the same with Rust (1987). [See my MPEC notes].
- We cheated a bit because we assumed that not only were actions discrete but so was the state space. This trick is often attributed to Pesendorfer and Schmidt-Dengler (ReStud 2008).
- If the state space is not discrete we need to do some forward-simulation [next slide]. (Hotz, Miller, Sanders, Smith ReStud 1994).
- Others have extended these ideas to dynamic games. See Aguirragabiria and Mira (Ecma 2002/2008) and Bajari Benkard and Levin (Ecma 2007).
- Srisuma and Linton (2009) [very hard] show how to use Friedholm integral equations of 2nd kind to extend to continuous case.

From NDP Notes: Policy Iteration (Howard 1960)

An alternative to value function iteration is policy function iteration.

- Make a guess for an initial policy, call it $a(s) = \arg\max_a U(a, s)$ that maps each state into an action
- Assume the guess is stationary compute the implied $V(a, s)$
- Improvement Step: improve on policy a_0 by solving

$$a' = \arg\max_a U(a, s) + \beta \sum_{s'} V(a, s') p(s'|s, a)$$

- Helpful to define $p(\tilde{s}'|s)$ as transition probability under optimal choice $a(s)$.
- Determine if $\|a' - a\| < \epsilon$. If yes then we have found the optimal policy a^* otherwise we need to go back to step 2.

From NDP Notes: Policy Iteration (Howard 1960)

Policy Iteration is even easier if choices AND states are discrete.

- For Markov transition matrix $\sum_j P_{ij} = 1$,we want $\pi P = \pi$
- $\lim_{t\to\infty} P^t = \pi$ where the jth element of π represents the long run probability of state j.
- We want the eigenvalue for which $\lambda = 1$.

Now updating the value function is easy for kth iterate of PI

$$\begin{aligned} V^k(s) &= Eu(a^k(s), s) + \beta \tilde{P}^k V^k(s) \\ \Rightarrow V^k(s) &= [1 - \beta \tilde{P}^k]^{-1} Eu(a^k(s), s) \end{aligned}$$

- Very fast when $\beta > 0.95$ and s is relatively small. (Rust says 500 more like 5000).
- Inverting a large matrix is tricky
- This trick is implicit in the HM/AM formulation.

Continuous State Space

When state space is continuous instead of discrete:
Approximation to the problem:

$$\hat{V}(s) = \max_{a \in \hat{A}(s)} \left[(1-\beta)u(s,a) + \beta \sum_{k=1}^{N} \hat{V}(s')p_N(s'_k|s,a) \right]$$

Exact Problem

$$V(s) = \max_{a \in A(s)} \left[(1-\beta)u(s,a) + \beta \int V(s')p(ds'|s,a) \right]$$

- Now we need to do actual numerical integration instead of just summation.
- We cannot use the $[I - \beta P]^{-1}$ to get the ergodic distribution.
- Usually requires interpolating between grid points to evaluate $EV(\cdot)$.

Forward Simulation

In practice, "truncate" the infinite sum at some period T:

$$\tilde{V}(x, d = 1; \theta) =$$
$$u(x, d = 1; \theta) + \beta E_{x'|x,d=1} E_{d'|x'} E_{\epsilon''|d',x'} [u(x', d'; \theta) + \epsilon'$$
$$+\beta E_{x''|x',d''} E_{d''|x''} E_{\epsilon'|d'',x''} [u(x'', d''; \theta) + \epsilon'' + \cdots$$
$$\beta E_{x^T|x^{T-1},d^{T-1}} E_{d^T|x^T} E_{\epsilon^T|d^T,x^T} [u(x^T, d^T; \theta) + \epsilon^T]]]$$

Also, the expectation $E_{\epsilon|d,x}$ denotes the expectation of the ϵ conditional choice d being taken, and current mileage x. For the logit case, there is a closed form:

$$E[\epsilon|d, x] = \gamma - \log(PR(d|x))$$

where γ is Euler's constant $(0.577\cdots)$ and $Pr(d|x)$ is the choice probability of action d at state x.

Both of the other expectations in the above expressions are observed directly from the data.

Forward Simulation

Choice-specific value functions can be simulated by (for $d = 1, 2$):

$$\begin{aligned}\tilde{V}(x, d; \theta) \approx= \quad & \tfrac{1}{S} \textstyle\sum_s [u(x, d; \theta) + \beta[u(x'^s, d'^s; \theta) + \gamma - \log(\hat{P}(d'^s|x'^s)) \\ & + \beta[u(x''^s, d''^s; \theta) + \gamma - \log(\hat{P}(d''^s|x''^s)) + \beta \cdots]]]\end{aligned}$$

where:

- $x'^s \sim \hat{G}(\cdot|x, d)$
- $d'^s \sim \hat{p}(\cdot|x'^s), x''^s \sim \hat{G}(\cdot|x'^s, d'^s)$
- & etc.

In short, you simulate $\tilde{V}(x, d; \theta)$ by drawing S sequences of (d_t, x_t) with a initial value of (d, x), and computing the present-discounted utility correspond to each sequence. Then the simulation estimate of $\tilde{V}(x, d; \theta)$ is obtained as the sample average.

Forward Simulation

- Given an estimate of $\tilde{V}(\cdot, d; \theta)$, you can get the *predicted choice probabilities*:

$$\tilde{p}(d = 1|x; \theta) \equiv \frac{\exp\left(\tilde{V}(x, d = 1; \theta)\right)}{\exp\left(\tilde{V}(x, d = 1; \theta)\right) + \exp\left(\tilde{V}(x, d = 0; \theta)\right)} \quad (1)$$

and analogously for $\tilde{p}(d = 0|x; \theta)$. Note that the predicted choice probabilities are different from $\hat{p}(d|x)$, which are the actual choice probabilities computed from the actual data. The predicted choice probabilites depend on the parameters θ, whereas $\hat{p}(d|x)$ depend solely on the data.