Dynamic Discrete Choice

Aviv Nevo

University of Pennsylvania and NBER

Fall 2018

Engine replacement problem: Rust 87

- Stopping time, to replace engine
  - Harold Zurcher was the city of Madison transportation boss
  - Decided maintenance and replacement of buses
- Paper provides framework for estimation of dynamic discrete choice problems
- Shows conditions for tractability
  - Additive separability (AS)
  - Conditional Independence (CI)
- Estimation: need to generate randomness, the key is where to allow for non-deterministic behavior
  - Links dynamics to DC models we saw earlier

Model

- Data: 10 yrs or monthly mileage +replacement for 104 buses
- Engines accumulate usage, summarized by x_t
  - In this case: miles driven by the bus in service
- Operating costs increase in $x : c(x_t, \theta)$
  - θ is a parameter to estimate
- Discrete choice: the firm can keep operating the engine or replace it at cost R
  - After replacement odometer reset: $x_t = 0$
- Exogenous usage: $\Rightarrow$ older doesn't mean drive less
  - Markov process
  - x_t distributed $P(x_t | x_{t-1})$, stochastically increasing in x_{t-1}

(Simple Version of the) Model

- Flow return

$$u(\overset{\substack{\text{action}\\ \uparrow}}{x, a}, \theta) = \begin{cases} -c(x_t, \theta) & \text{if } a_t = 0 \\ -R & \text{if } a_t = 1 \end{cases} \quad \begin{array}{l} \Rightarrow \text{no replace} \\ \Rightarrow \text{replace.} \end{array}$$

(profit)

  - a is the discrete replacement choice (action)
  - if $a_t = 1$ then $x_t = 0$
  - $c(0, \theta)$ included in R, to save notation

- The value function is:

$$\begin{aligned} V(x_t) &= \max\{-c(x_t, \theta) + \beta \int V(x_{t+1}) dP(x_{t+1}|x_t), \\ &\quad -R + \beta \int V(x_{t+1}) dP(x_{t+1}|0)\} \end{aligned}$$

- The policy function involves a (deterministic) cut-off above which the engine is replaced
- A deterministic cutoff will almost surely be rejected by the data (actual replacement in the data 83-387K)

Adding Randomness: Additive Separability

- to rationalize the data, that shows a distribution of replacement times one needs to assume:
  - measurement error
  - optimization error
  - unobserved state variables
- Rust develops a model of the latter by assuming:

$$u(x, a, \varepsilon, \theta) = \begin{cases} -c(x_t, \theta) + \varepsilon_{t0} & \text{if } \ a_t = 0 \\ -R + \varepsilon_{1t} & \text{if } \ a_t = 1 \end{cases}$$

- The ε_t are unobserved shocks assumed additively separable
- First key assumption: Additive Separability (AS)

Empirical Model

- The value function becomes:

$$V(x_t, \varepsilon_t) = \max\{-R + \varepsilon_{1t} + \beta \int V(x_{t+1}, \varepsilon_{t+1}) dP(x_{t+1}, \varepsilon_{t+1} | 0, \varepsilon_t),$$

$$-c(x_t; \theta) + \varepsilon_{0t} + \beta \int V(x_{t+1}, \varepsilon_{t+1}) dP(x_{t+1}, \varepsilon_{t+1} | x_t, \varepsilon_t)\}$$

- Usually we will assume that ε iid over time (and options)
  - Serial correlation in ε complicates the problem but still feasible
- We can solve the model by discretizeing the state space
  - need transition matrix T (#of states by #of states)
  - search for vector V that solves system of equations
  - can use PFI or VFI (more later)
- If we discretize in all 3 dimensions the state space increases very rapidly
  - say K values in each dimension then K^3 (or more generally K^{J+1} if we have J options)

Conditional Independence

- To further simplify the problem we will want to write it a bit differently
- We will need an assumption of conditional independence (CI):

$$P(x_{t+1}, \varepsilon_{t+1} | x_t, \varepsilon_t) = P_1(x_{t+1} | x_t) P_2(\varepsilon_{t+1} | \varepsilon_t)$$

  - ($P_2(\varepsilon_{t+1} | \varepsilon_t, x_t)$ would be fine too)

- Using the CI assumption we can define the *integrated value function:*

$$\begin{aligned} EV(x_{t+1}) &= \int V(x_{t+1}, \varepsilon_{t+1}) dP(\varepsilon_{t+1} | x_t, \varepsilon_t, x_{t+1}) \\ &= \int V(x_{t+1}, \varepsilon_{t+1}) dP_2(\varepsilon_t) \end{aligned}$$

t+1

  - meaning of $EV(x_t)$: value before drawing ε's

Rewriting the Bellman Equation

- Define:

$$\begin{aligned} \underbrace{\delta_0}_{\text{mean util}}(x_t) &= -c(x_t;\theta) + \beta \int EV(x_{t+1})dP(x_{t+1}|x_t) \\ \delta_1(x_t) &= -R + \beta \int EV(x_{t+1})dP(x_{t+1}|0) \end{aligned} \tag{1}$$

- Using the integrated value function:

$$\int V(x_{t+1}, \varepsilon_{t+1})dP(x_{t+1}, \varepsilon_{t+1}|x_t, \varepsilon_t) = \int EV(x_{t+1})dP(x_{t+1}|x_t)$$

- Thus (under CI) the Bellman Equation can be written as

$$V(x_t, \varepsilon_t) = \max\{\delta_0(x_t) + \varepsilon_{0t}, \delta_1(x_t) + \varepsilon_{1t}\} \tag{2}$$

- Taking expectations:

$$EV(x_t) = \int [\max\{\delta_0(x_t) + \varepsilon_{0t}, \delta_1(x_t) + \varepsilon_{1t}\}] \, dP(\varepsilon_t)$$

How does CI help?

- CI helps reduce the dimensionality, current ε affect behavior but past ones are not part of the state
- Can write the Bellman equation in $V_\sigma = EV(\cdot)$
- We can iterate to find EV (see below)
- Lower dimension (AND does not increase in J)
    - we will see how to compute the integral easily
- In some sense EV is not what we are after, but ...
    - use EV jointly with equation (2) to get the value function $V(x_t, \varepsilon_t)$
    - to get the policy function only need EV

Getting a Likelihood

- Optimal behavior in every state is given by:

$$j = \arg\max_k \{\delta_k(x) + \varepsilon_{kt}\}$$

- Just like static DC, but (conditional) value replaces (conditional) utility
- Thus ,probability of an action:

$$\Pr(j|x_t) = \Pr(j = \arg\max_k \{\delta_k(x_t) + \varepsilon_{kt}\}) \tag{3}$$

- If ε_{kt} is iid extreme value

$$\Pr(j|x_t) = \frac{\exp(\delta_j(x_t))}{\sum_k \exp(\delta_k(x_t))}$$

- AS+CI+logit errors allows us to write a likelihood that is very similar to the static case
- Key: we need to solve for $V(x_t, \varepsilon_t)$ (or $EV(x_t)$) to compute $\delta_k(x_t)$

Extreme Value and EV

- The iid extreme value assumption can help further simplify $EV(x_t)$

$$
\begin{aligned}
EV(x_t) &= \int \left[\max_k \{\delta_k(x_t) + \varepsilon_{kt}\} \right] dP(\varepsilon_t) \\
&= \ln\left(\sum_k \exp(\delta_k(x_t)) \right)
\end{aligned}
\tag{4}
$$

the last equality comes from the extreme value distribution (recall inclusive value)

Estimation: Nested FP algorithm

- Data: panel of buses (n) over time (t)
    - binary choice $a(x_{nt}) = 1$ if replace engine
- We want choose the parameters that

$$\max \prod_n \prod_t \Pr(1|x_{nt})^{a(x_{nt})} \Pr(0|x_{nt})^{1-a(x_{nt})}$$

  where the probability has a logit form
- Nested algorithm:
    - Inner loop: for a given guess of the parameters, solve for EV, plug it into $\delta_0(x_{nt})$ and $\delta_1(x_{nt})$ and compute the likelihood
    - Outer loop: search over parameters that maximizes likelihood

Inner Loop: solving for EV

- Many possible solution methods, I will only discuss one
- Discretize the problem (if we work with EV, only need to discretize x)
    - Rust discretizes x into 90 values, $K = 90$. Thus, optimal behavior is characterized by the 90 values EV
- Value function iteration:
    - start with any guess vector EV (90 by 1)
    - iterate on (4) until convergence

Other Elements

- The other elements in EV are:
  - R, a scalar to be estimated
  - $c(x_t; \theta)$, the cost function – will be parametrized, for example, $\theta_1 x + \theta_2 x^2$
  - $P(x_{t+1}|x_t)$ defines a $K * K$ transition matrix that reflects the probability of moving across states
- Usually will be estimated from the data before the NFP
- Can also be parametrized, for example, $x_{t+1} = x_t + 1$ with probability θ_3, and $x_{t+1} = x_t$ with probability $1 - \theta_3$

Transition matrices

Define the transition matrices $F(a)$ for $a = 0, 1$

$$F(0) = \begin{bmatrix} 1-\theta_3 & \theta_3 & 0 & \dots & 0 \\ 0 & 1-\theta_3 & \theta_3 & & \\ \dots & 0 & \dots & \dots & 0 \\ \dots & & & 1-\theta_3 & \theta_3 \\ 0 & \dots & & 0 & 1-\theta_3 \end{bmatrix}$$

and

$$F(1) = \begin{bmatrix} 1-\theta_3 & \theta_3 & 0 & \dots & 0 \\ 1-\theta_3 & \theta_3 & & & \\ \dots & & & & \\ \dots & & & & \\ 1-\theta_3 & \theta_3 & 0 & \dots & 0 \end{bmatrix}$$

Methods for Solving DP Problems

- Write the dynamic programming problem as

$$V_t(s_t) = \max_{a_t} \left\{ R_t(s_t, a_t) + \beta \mathbb{E}[V_{t+1}(s_{t+1}) | s_t, a_t] \right\}$$

- DP problems taxonomy:
  - finite/infinite horizon;
  - stationary/non-stationary.
- We'll focus on stationary problems:

$$V(s) = \max_{a} \left\{ R(s, a) + \beta \mathbb{E}[V(s') | s, a] \right\}$$

Discretizing the Problem

Assume the state space S is finite. Then we can write

$$\begin{aligned} V(s) &= \max_a \left\{ R(s,a) + \beta \mathbb{E}[V(s')|s,a] \right\} \\ &= \max_a \left\{ R(s,a) + \beta \sum_{s'=1}^{S} V(s')\mathbb{P}(s'|s,a) \right\} \quad s = 1, \ldots, S \end{aligned}$$

Note that this is a system of S equations in S unknowns.

Discretizing the Problem

Value Function Iteration

Step 1: guess a value for $V(\cdot)$ - i.e., S numbers. Call it V_0.

Step 2: compute V_t by solving

$$V_t(s) = \max_a \left[R(s,a) + \beta \sum_{s'=1}^{S} V_{t-1}(s') \mathbb{P}(s'|s,a) \right]$$

Step 3: is $\| V_t - V_{t-1} \| \leq \varepsilon$?

- If it is, stop.
- If it's not, go back to step 2.

Note that

- VFI is guaranteed to converge by the Contraction Mapping Theorem.
- Can be considered timing.
- The algorithm can be very slow for β close to 1.

Discretizing the Problem

Policy Function Iteration

Define $a^*(s) := \text{argmax}_a \left[R(s,a) + \beta \sum_{s'=1}^{S} V(s')\mathbb{P}(s'|s,a) \right]$.
Then

$$V(s) = R(s, a^*(s)) + \beta \sum_{s'=1}^{S} V(s')\mathbb{P}(s'|s, a^*(s))$$

Define

$$v = \begin{pmatrix} V(1) \\ \vdots \\ V(S) \end{pmatrix} \quad \text{and} \quad R(a^*) = \begin{pmatrix} R(1, a^*(1)) \\ \vdots \\ R(S, a^*(S)) \end{pmatrix}$$

Discretizing the Problem

Policy Function Iteration (cont)

Also define

$$\mathbb{P}(a^*) = \begin{bmatrix} \mathbb{P}(1|1, a^*(1)) & \dots & \mathbb{P}(S|1, a^*(1)) \\ \vdots & & \vdots \\ \mathbb{P}(1|S, a^*(S)) & \dots & \mathbb{P}(S|S, a^*(S)) \end{bmatrix}$$

It follows that

$$V = R(a^*) + \beta\mathbb{P}(a^*)V \Rightarrow V = (I - \beta\mathbb{P}(a^*))^{-1} R(a^*) \tag{5}$$

Discretizing the Problem

Policy Function Iteration (cont)

Step 1: guess $a^*(s), s = 1, \ldots, S$.

Step 2: compute V_{t-1} by equation (5).

Step 3: update a_t^* by

$$a_t^*(s) := \text{argmax}_a \left[R(s,a) + \beta \sum_{s'=1}^{S} V_{t-1}(s') \mathbb{P}(s'|s,a) \right].$$

Step 4: If $\|a_{t-1}^* - a_t^*\| < \varepsilon$, stop. If not, go back to step 2.

Discretizing the Problem

Policy Function Iteration (cont)

Comments on PFI:

- PFI tends to require less steps than VFI.
- It is guaranteed to converge if the state space is finite.
  - Intuition: $V(\cdot)$ might still change even though actions do not.
- Can be demanding with large S.

Variations on PFI:

- Can update states one at a time.
- Multistage look ahead.

Discretizing the Problem

Linear Programming

Solve the LP problem

$$\min_{V \in \mathbb{R}^S} \sum_{s=1}^{S} V_s$$

$$\text{s.t.} \quad V_s \geq R(s,a) + \beta \sum_{s'=1}^{S} V_{s'} \mathbb{P}(s'|s,a), \forall a$$

If $a \in \{a_1, \ldots, a_K\}$, this is a LP problem with S controls and $S \times K$ constraints.

Approximation Methods

Parametric Approximation of VF with PFI

The appropriate discrete state space might be too large

- Several state variables
- Several players

Discretizing also throws away information. In those cases we need different methods.

Assume $\tilde{V}(s) = \sum_{k=1}^{K} \theta^k r_k(s)$.

- $\tilde{V}(s)$ is the approximation of $V(s)$.
- $r(s)$ are the approximation basis, e.g., $1, s, s^2, s^3, \ldots$ - though many others can (and should) be used.
- θ's are unknown parameters.

Approximation Methods

Parametric Approximation of VF with PFI Cont.

Step 1 Guess $a^*(s)$. Choose points $s = 1, \dots, S$ to evaluate approximation.

Step 2 Compute θ_t. If a_t^* is optimal then

$$V(s) = R(s, a_t^*) + \beta \mathbb{E}[V(s')|s, a_t^*]$$

Using the approximation

$$\sum_{k=1}^{K} \theta^k r_k(s) \approx R(s, a_t^*) + \beta \mathbb{E}\left[\sum_{k=1}^{K} \theta^k r_k(s')|s, a_t^*\right]$$

or

$$\sum_{k=1}^{K} \theta^k \left[r_k(s) - \beta \mathbb{E}[r_k(s')|s, a_t^*]\right] \approx R(s, a_t^*)$$

$\hat{\theta}_t$ can be computed by least squares: $\hat{\theta} = (X'X)^{-1}X'Y$ where

$$X = \underbrace{r - \beta \mathbb{E}[r|a_t^*]}_{S \times K} \text{ and } \underbrace{Y = R(a_t^*)}_{S \times 1}$$

Step 3 Policy improvement.

$$a^*_{t+1}(s) = \underset{a}{\operatorname{argmax}} \left\{ R(s,a) + \beta \mathbb{E} \left[\sum_{k=1}^{K} \hat{\theta}^k_t r(s') \middle| s, a \right] \right\}$$

Step 4 Repeat 2-3 until convergence (of $\hat{\theta}$, a^* or $\tilde{V}$).

Advantages of this method:

- No curse of dimensionality
- Can be very fast

Approximation Methods

Parametric Approximation of VF with PFI Cont.

Issues:

- Not a contraction mapping.
- No guarantee that it will converge to the true $V(\cdot)$ when it does converge.

Choices:

- Functions $r(\cdot)$. Shape preserving.
- How to evaluate $\mathbb{E}r$
  - Simulation.
  - Numerical quadrature.
- Computing the maximum in step 3.
- Computing $\hat{\theta}$ in step 2.
- Points for evaluation.

Alternative Estimators

- The NFP is in many ways intuitive and can be extended to many cases (heterogeneity, serial correlation, additional unobserved states)
- However, in large problems/games it might not be feasible
- Therefore, an alternative set of methods have been explored (Hotz-Miller, HMSS, AM)
- These methods aim to exploit the one-to-one mapping (under AS+CI+iid) between choice probabilities and conditional values
- Similar to the "Berry inversion" we saw in static models
- But it differs in 2 important ways
  - in the static model the mapping is with conditional utility
  - here with conditional value: how do we get EV
  - also the values in different states are linked (unlike the static case)
- We will now show how we can still exploit this mapping for estimation.

- We can rewrite $EV(x_t)$ as:

$$\begin{aligned} EV(x_t) &= \int \left[\max_k \{\delta_k(x_t) + \varepsilon_{kt}\} \right] dP(\varepsilon_t) \\ &= \sum_k P(k|x)(\delta_k(x_t) + e_k(x_t)) \end{aligned}$$

Prob * utility

where

$$e_j(x_t) = E(\varepsilon_j | j = \arg\max_k \{\delta_k(x_t) + \varepsilon_{kt}\})$$

- $e_i(x)$ is only a function of $P(i|x)$ (and perhaps parameters)
- For the extreme value case (to be used later):

$$e_i(x) = \gamma - \ln(P(i|x)) \qquad \gamma = 0.577216$$

Since x is discrete, for the Rust model, we can stack states

$$EV = P(0).*(-c(\theta) + e_0 + \beta F(0)EV) + P(1).*(-R + e_1 + \beta F(1)EV)$$

elt by elt mult

rearranging:

$$M * EV = P(0).*(-c(\theta) + e_0) + P(1).*(-R + e_1)$$

where

$$M = (I_{K*K} - \beta(P(0) * ones(1,k).*F(0) + P(1) * ones(1,k).*F(1))$$

therefore,

$$EV = M^{-1}\{P(0).*(-c(\theta) + e_0) + P(1).*(-R + e_1)\}$$

the expected value of behaving according to $g(a = 1|x) = P(1|x)$

if know choice probs, can compute EV

Notation is impossible

Hotz and Miller: 2 step procedure

- Step 1: get estimates of the conditional choice probabilities (CCP) $P(a|x)$, call them $\widehat{P(a|x)}$
  - Probit, or frequency of a in state x (or some other NP estimate)
- Compute EV
  - use first step estimates to compute $e_a(x)$: for Logit $e_a(x) = \gamma - \ln(\widehat{P(a|x)})$
  - compute EV using the above formula using $\widehat{P(a|x)}$

plug in $\hat{p}$ for p

$$EV(\widehat{P(a|x)}, \theta) = M^{-1}(\widehat{P(a|x)}, \theta)\{\widehat{P(0|x)}. * (-c(\theta) + e_0) + \widehat{P(}$$

- Step 2: Estimate the parameters by ML
  - use computed EV to compute $\delta_k(x, \theta)$ and choice probabilities

$$\Pr(j|x_t) = \frac{\exp(\delta_j(EV(\widehat{P(j|x_t)}, \theta)))}{\sum_k \exp(\delta_k(EV(\widehat{P(k|x_t)}, \theta)))}$$

Aguirregabiria-Mira: Recursive CCP

- Start as in HM with guess/estimate of CCP $\widehat{P^k(a|x)}$ ($k=1$ for initial guess)

$$EV(\widehat{P^k(a|x)},\theta) = \Psi(\widehat{P^k(a|x)},\theta)$$

- Use $EV(\widehat{P^k(a|x)},\theta)$ to get an estimate of $\theta : \widehat{\theta}^k$
- Update the choice probabilities

$$P^{k+1}(a|x) = \frac{\exp(\delta_a(EV(\widehat{P^k(a|x_t)},\widehat{\theta}^k)))}{\sum_j \exp(\delta_j(EV(\widehat{P^k(j|x_t)},\widehat{\theta}^k)))}$$

- Iterate this process ↳ fully converge => Rust
  - $k=1$ will yield H-M
  - The FP, $P(a|x)=\Psi(P(a|x),\theta)$, equals Rust's NFP
  - it is the equivalent of PFI (in terms of probabilities of taking an action)
  - key result: as long as initial $\widehat{P^k(a|x)}$ are consistent then no need to iterate until convergence 7 or 8 iterations enough

Hotz-Miller-Sanders-Smith

- Like HM estimate conditional choice probabilities (CCP): $\widehat{P(a|x)}$
- Compute estimate of $\widehat{\delta}(a|x) = \log \frac{\widehat{P(a|x)}}{\widehat{P(1|x)}}$ (after normalizing $\widehat{\delta}(1|x) = 0$)
- Compute optimal behavior given $\widehat{\delta}(a|x)$: $g_a(x, \varepsilon) = \arg\max_a \{\widehat{\delta}(a|x) + \varepsilon_a\}$
- Simulate future path of states $\{\widetilde{x}_t, \widetilde{\varepsilon}_t\}$and actions
- Compute $\widetilde{\delta}(x, \varepsilon|\theta) =$ $u(x, a|\theta) + \varepsilon(a) + \sum \beta^t \{u(\widetilde{x}_t, g_a(\widetilde{x}_t, \widetilde{\varepsilon}_t)|\theta) + \widetilde{\varepsilon}_t(g_a(\widetilde{x}_t, \widetilde{\varepsilon}_t))\}$
- Choose θ to minimize $\|\widetilde{\delta}(x, \varepsilon|\theta) - \widehat{\delta}(a|x)\|$

Final Comments

- The CCP based methods have been around for a while
- Have recently been used more (maybe due to larger problems/games)
- A main concern with them is that they do not allow for unobserved heterogeneity
  - In NFP can just integrate it out
  - here, need CCP for different "types"
- Some recent progress on the topic
- Dynamic games
  - we will not cover
  - rely on some of the same ideas: optimal behavior holding other agents and owen future behavior fixed.