Dynamic Discrete Choice Models

An Introduction

Shihang Hou Giulio Schinaia

University of Oxford

November 22, 2021

Outline

Today

- Give an overview of what a dynamic discrete choice model is, and give a sense of what's out there
- Focus on main elements of single-agent models (the most common)

Today

- Give an overview of what a dynamic discrete choice model is, and give a sense of what's out there
- Focus on main elements of single-agent models (the most common)
- Present an example and Matlab code based on Rust (1987)

Today

- Give an overview of what a dynamic discrete choice model is, and give a sense of what's out there
- Focus on main elements of single-agent models (the most common)
- Present an example and Matlab code based on Rust (1987)
- Presentation based largely on review article by Aguirregabiria and Mira (2010) (very useful reference!)

Standard choice models

- For the standard textbook reference, see Train (2009) (recommended by Abi)
- Basic idea is to specify a utility for each choice $j \in \{1, \cdots, J\}$ (in the choice set) based on some regular, observable elements of the choice (u_j)

Standard choice models

- For the standard textbook reference, see Train (2009) (recommended by Abi)
- Basic idea is to specify a utility for each choice $j \in \{1, \cdots, J\}$ (in the choice set) based on some regular, observable elements of the choice (u_j)
- Total utility for the choice j (U_j) is sum of observable component (u_j) and a component observed by agent but not the econometrician (ε_j)

Standard choice models

- For the standard textbook reference, see Train (2009) (recommended by Abi)
- Basic idea is to specify a utility for each choice $j \in \{1, \cdots, J\}$ (in the choice set) based on some regular, observable elements of the choice (u_j)
- Total utility for the choice j (U_j) is sum of observable component (u_j) and a component observed by agent but not the econometrician (ε_j)
- Choice prob given by:

$$
\begin{aligned}
Pr(C = j) &= Pr(U_j > U_k, \forall k \neq j) \\
&= Pr(\varepsilon_j - \varepsilon_k > -(u_j - u_k), \forall k \neq j)
\end{aligned}
$$

What is "dynamic" about DDC models?

- In DDC models, choices in each period affects the choice problem in future periods

What is "dynamic" about DDC models?

- In DDC models, choices in each period affects the choice problem in future periods
  - e.g. 1: choosing to work in current period raises human capital (and hence wages) in future periods (Keane and Wolpin (1997))
  - e.g. 2: whether to retire in each period

What is "dynamic" about DDC models?

- In DDC models, choices in each period affects the choice problem in future periods
  - e.g. 1: choosing to work in current period raises human capital (and hence wages) in future periods (Keane and Wolpin (1997))
  - e.g. 2: whether to retire in each period
- If not, then standard choice model applied to panel data (with multiple observations per chooser, allowing controls for individuals)

What is "dynamic" about DDC models?

- In DDC models, choices in each period affects the choice problem in future periods
  - e.g. 1: choosing to work in current period raises human capital (and hence wages) in future periods (Keane and Wolpin (1997))
  - e.g. 2: whether to retire in each period
- If not, then standard choice model applied to panel data (with multiple observations per chooser, allowing controls for individuals)
  - Static case can be seen as a special case of the dynamic, where state variables are independent of choices.

What is difficult about DDC models?

- The choice in a given period affects choices in future periods, and thus those have to be taken into account...
- ...So have to solve for the value of each choice at each value of the observed state variables in each period.

What is difficult about DDC models?

- The choice in a given period affects choices in future periods, and thus those have to be taken into account...
- ...So have to solve for the value of each choice at each value of the observed state variables in each period.
- In general, have to do this once for each trial parameter values. $\rightarrow$ substantial computation cost!
- For sufficiently rich models, this is impossible...so have to reduce the dimensionality via discretisation of the state space or by interpolation.

An illustration of the possibilities

An illustration of the possibilities

- Single agent models (Most common; see e.g. Keane and Wolpin (1997), Arcidiacono et al. (2014), Adda et al. (2017) etc.)
  - Models behaviour of a single individual in some life domain
  - Also mixed in with some continuous components (e.g. consumption in Adda et al. (2017))

An illustration of the possibilities

- Single agent models (Most common; see e.g. Keane and Wolpin (1997), Arcidiacono et al. (2014), Adda et al. (2017) etc.)
  - Models behaviour of a single individual in some life domain
  - Also mixed in with some continuous components (e.g. consumption in Adda et al. (2017))
- General equilibrium models (see Heckman et al. (1998), Lee (2005), Lee and Wolpin (2006))
  - Similar to single agent, but incorporates a set of equilibrium conditions, which adds a degree of difficulty to the solution

An illustration of the possibilities

- Single agent models (Most common; see e.g. Keane and Wolpin (1997), Arcidiacono et al. (2014), Adda et al. (2017) etc.)
  - Models behaviour of a single individual in some life domain
  - Also mixed in with some continuous components (e.g. consumption in Adda et al. (2017))
- General equilibrium models (see Heckman et al. (1998), Lee (2005), Lee and Wolpin (2006))
  - Similar to single agent, but incorporates a set of equilibrium conditions, which adds a degree of difficulty to the solution
- Dynamic discrete choice games (see Aguirregabiria and Mira (2007, Econometrica))
  - Point of dynamic games is that current play considers future possibilities $\rightarrow$ can use structural DDC model to recover description of subject preferences

What is it good for?

- You have a problem characterised by repeated choices made by agents where each choice affects other choices in future periods.

What is it good for?

- You have a problem characterised by repeated choices made by agents where each choice affects other choices in future periods.
- You have a panel dataset consisting for each individual i of $\{a_{it}, s_{it}, y_{it}\}_{t\in\{1,\cdots,T\}}$ (observed choice, state variables, and potentially some output variables)

What is it good for?

- You have a problem characterised by repeated choices made by agents where each choice affects other choices in future periods.
- You have a panel dataset consisting for each individual i of $\{a_{it}, s_{it}, y_{it}\}_{t\in\{1,\cdots,T\}}$ (observed choice, state variables, and potentially some output variables)
- Estimating a DDC model recovers the parameters on (i) preferences, and (ii) parameters of the process governing change in the state variables

What is it good for?

- You have a problem characterised by repeated choices made by agents where each choice affects other choices in future periods.
- You have a panel dataset consisting for each individual i of $\{a_{it}, s_{it}, y_{it}\}_{t\in\{1,\cdots,T\}}$ (observed choice, state variables, and potentially some output variables)
- Estimating a DDC model recovers the parameters on (i) preferences, and (ii) parameters of the process governing change in the state variables
  - Either set of parameters may be of inherent interest...
  - ...or, you want to simulate the implementation of some policy (Low and Pistaferri (2015))

Outline

Key elements of a dynamic discrete choice model

- Choice set ($A = \{0, 1, \cdots, J\}$): a finite set of allowable values of the control variable a_{it} in period t

Key elements of a dynamic discrete choice model

- Choice set ($A = \{0, 1, \cdots, J\}$): a finite set of allowable values of the control variable a_{it} in period t

- $s_{it} = \{s_{it}^1, \cdots, s_{it}^K\}$: a K-dimensional vector of state variables known by the agent at time t (but some possibly unobserved by the econometrician)

Key elements of a dynamic discrete choice model

- Choice set ($A = \{0, 1, \cdots, J\}$): a finite set of allowable values of the control variable a_{it} in period t
- $s_{it} = \{s_{it}^1, \cdots, s_{it}^K\}$: a K-dimensional vector of state variables known by the agent at time t (but some possibly unobserved by the econometrician)
- Utility function $U(a, s)$: A function specifying the utility that individuals derive from the collection of states s and the set of made decisions a

Key elements of a dynamic discrete choice model

- Choice set ($A = \{0, 1, \cdots, J\}$): a finite set of allowable values of the control variable a_{it} in period t
- $s_{it} = \{s_{it}^1, \cdots, s_{it}^K\}$: a K-dimensional vector of state variables known by the agent at time t (but some possibly unobserved by the econometrician)
- Utility function $U(a, s)$: A function specifying the utility that individuals derive from the collection of states s and the set of made decisions a
- State transition function $F(s_{i,t+1}|a_{it}, s_{it})$: A description of how the state variables s evolve in future periods given the choice and state in the current period

Key elements of a dynamic discrete choice model

- Choice set ($A = \{0, 1, \cdots, J\}$): a finite set of allowable values of the control variable a_{it} in period t
- $s_{it} = \{s_{it}^1, \cdots, s_{it}^K\}$: a K-dimensional vector of state variables known by the agent at time t (but some possibly unobserved by the econometrician)
- Utility function $U(a, s)$: A function specifying the utility that individuals derive from the collection of states s and the set of made decisions a
- State transition function $F(s_{i,t+1}|a_{it}, s_{it})$: A description of how the state variables s evolve in future periods given the choice and state in the current period
- A specification of the time horizon (infinite, or finite with max T), and time preference β (usually calibrated, as it is badly identified)

What does it mean to solve the model?

- In each period t, the agent's objective is to maximise expected utility in subsequent periods by choosing the optimal action depending on the state $(\alpha(s_{it}))$:

$$\alpha(s_{it}) = \arg\max_{a \in A} E\left(\sum_{j=0}^{T-t} \beta^j U(a_{i,t+j}, s_{i,t+j}) | a, s_{it}\right)$$

What does it mean to solve the model?

- In each period t, the agent's objective is to maximise expected utility in subsequent periods by choosing the optimal action depending on the state ($\alpha(s_{it})$):

$$\alpha(s_{it}) = \arg\max_{a \in A} E\left(\sum_{j=0}^{T-t} \beta^j U(a_{i,t+j}, s_{i,t+j}) | a, s_{it}\right)$$

- To handle this calculation, re-express the problem in terms of Bellman's principle of optimality

$$v(a, s_{it}) \equiv U(a, s_{it}) + \beta V(s_{i,t+1}) dF(s_{i,t+1} | a, s_{it}) \tag{1}$$

$$V(s_{it}) = \max_{a \in A} \{v(a, s_{it})\} \tag{2}$$

What does it mean to solve the model?

- In each period t, the agent's objective is to maximise expected utility in subsequent periods by choosing the optimal action depending on the state ($\alpha(s_{it})$):

$$\alpha(s_{it}) = \arg\max_{a \in A} E\left(\sum_{j=0}^{T-t} \beta^j U(a_{i,t+j}, s_{i,t+j}) | a, s_{it}\right)$$

- To handle this calculation, re-express the problem in terms of Bellman's principle of optimality

$$v(a, s_{it}) \equiv U(a, s_{it}) + \beta V(s_{i,t+1}) dF(s_{i,t+1} | a, s_{it}) \tag{1}$$

$$V(s_{it}) = \max_{a \in A} \{v(a, s_{it})\} \tag{2}$$

- To solve the model is to evaluate $V(s_{it})$ over the support of s_{it}

From a theoretical to an empirical DDC model (1)

- Let $\theta = \{\theta_u, \theta_f\}$ denote the parameters that parameterise the utility and transition functions

From a theoretical to an empirical DDC model (1)

- Let $\theta = \{\theta_u, \theta_f\}$ denote the parameters that parameterise the utility and transition functions
- Assume $\boldsymbol{s}_{it} = \{\boldsymbol{x}_{it}, \varepsilon_{it}\}$, where $\boldsymbol{x}_{it}$ is an subvector of observable elements of $\boldsymbol{s}_{it}$ and ε_{it} is a subvector of unobservable elements of $\boldsymbol{s}_{it}$

From a theoretical to an empirical DDC model (1)

- Let $\theta = \{\theta_u, \theta_f\}$ denote the parameters that parameterise the utility and transition functions
- Assume $\boldsymbol{s}_{it} = \{x_{it}, \varepsilon_{it}\}$, where $\boldsymbol{x}_{it}$ is an subvector of observable elements of $\boldsymbol{s}_{it}$ and ε_{it} is a subvector of unobservable elements of $\boldsymbol{s}_{it}$
- May also observe a payoff variable (y_{it}), related to $a_{it}, x_{it}, \varepsilon_{it}$ by some function $y_{it} = \mathcal{Y}(a_{it}, x_{it}, \varepsilon_{it})$ Not relevant for today's demo.

$$\text{Data} = \{a_{it}, x_{it}, y_{it} : i = 1, 2, \cdots, N; t = 1, 2, \cdots, T\} \tag{3}$$

From a theoretical to an empirical DDC model (2)

- Typically, we are trying to estimate θ from the data, usually by GMM, maximum likelihood or some simulated equivalent

From a theoretical to an empirical DDC model (2)

- Typically, we are trying to estimate θ from the data, usually by GMM, maximum likelihood or some simulated equivalent
- The most basic method is maximum likelihood, where the likelihood of observing each set of values $\{a_{it}, x_{it}, y_{it}\}, \forall t \in \{1, \cdots, T\}$ is given by $l_i(\theta)$

$$\begin{aligned} l_i(\theta) &= \log Pr\{a_{it}, y_{it}, x_{it} : t = 1, 2, \cdots, T|\theta\} \\ &= \log Pr\{\alpha(x_{it}, \varepsilon_{it}, \theta) = a_{it}, \mathcal{Y}(a_{it}, x_{it}, \varepsilon_{it}, \theta) = y_{it}, x_{it} : t = 1, 2, \cdots, T|\theta\} \end{aligned}$$

From a theoretical to an empirical DDC model (2)

- Typically, we are trying to estimate θ from the data, usually by GMM, maximum likelihood or some simulated equivalent
- The most basic method is maximum likelihood, where the likelihood of observing each set of values $\{a_{it}, x_{it}, y_{it}\}, \forall t \in \{1, \cdots, T\}$ is given by $l_i(\theta)$

$$
\begin{aligned}
l_i(\theta) &= \log Pr\{a_{it}, y_{it}, x_{it} : t = 1, 2, \cdots, T|\theta\} \\
&= \log Pr\{\alpha(x_{it}, \varepsilon_{it}, \theta) = a_{it}, \mathcal{Y}(a_{it}, x_{it}, \varepsilon_{it}, \theta) = y_{it}, x_{it} : t = 1, 2, \cdots, T|\theta\}
\end{aligned}
$$

- Evaluating the objective requires solution of the problem at each proposed parameter value $\rightarrow$ i.e. evaluating $\alpha(x_{it}, \varepsilon_{it}, \theta)$ for each θ for each x_{it}, ε_{it}

From a theoretical to an empirical DDC model (3)

- Without further functional form/distributional assumptions, the likelihood cannot be evaluated

From a theoretical to an empirical DDC model (3)

- Without further functional form/distributional assumptions, the likelihood cannot be evaluated
- This part is fairly open-ended, with computation and tractability as the main limiting factors

From a theoretical to an empirical DDC model (3)

- Without further functional form/distributional assumptions, the likelihood cannot be evaluated
- This part is fairly open-ended, with computation and tractability as the main limiting factors
- Well-known 'off-the-shelf' sets of assumptions:

From a theoretical to an empirical DDC model (3)

- Without further functional form/distributional assumptions, the likelihood cannot be evaluated
- This part is fairly open-ended, with computation and tractability as the main limiting factors
- Well-known 'off-the-shelf' sets of assumptions:
    - Rust framework - simplest framework which assumes independence and conditional logit errors, with faster estimation methods which leverage the unique functional form assumptions

From a theoretical to an empirical DDC model (3)

- Without further functional form/distributional assumptions, the likelihood cannot be evaluated
- This part is fairly open-ended, with computation and tractability as the main limiting factors
- Well-known 'off-the-shelf' sets of assumptions:
  - Rust framework - simplest framework which assumes independence and conditional logit errors, with faster estimation methods which leverage the unique functional form assumptions
  - Eckstein-Keane-Wolpin models - joint normal errors, allowing for covariance between errors for each choice, and using mixture models to allow for non-additive heterogeneity

Rust's assumptions (1)

- (AS) Additive separability

$$U(a, x_{it}, \varepsilon_{it}) = u(a, x_{it}) + \varepsilon_{it}(a)$$

- (IID) Unobserved state variables are i.i.d across agents and time, with common cdf $G_{\varepsilon}(\varepsilon_{it})$
- (CIX) Next period observable state variables do not depend on unobserved state variables (ε_{it})

$$CDF(x_{it}|x_{it}, a_{it}, \varepsilon_{it}) = F_x(x_{i,t+1}|a_{it}, x_{it})$$

  Denote parameters of F_x by θ_f

- CIX + IID $\rightarrow F(x_{i,t+1}, \varepsilon_{i,t+1}|a_{it}, x_{it}, \varepsilon_{it}) = G_{\varepsilon}(\varepsilon_{i,t+1})F_x(x_{i,t+1}|a_{it}, x_{it})$

Rust's assumptions (2)

- (CIY) Conditional on current values of the decision and observable state variables, the value of the payoff variable is independent of ε_{it}

$$Y(a_{it}, x_{it}, \varepsilon_{it}) = Y(a_{it}, x_{it})$$

- (CLOGIT) The unobserved state variables $\{\varepsilon_{it}(a) : a = 0, 1, \cdots, J\}$ are independent across alternatives and have an extreme value type I distribution.

- (Discrete support of x) The support of x_{it} is discrete and finite: $x_{it} \in X = \{x^{(1)}, x^{(2)}, \cdots, x^{(|X|)}\}$ with $|X| < \infty$.

What Rust's assumptions buys us (1) - evaluating the Emax function

- Fully flexible model

$$V(x_{it}, \varepsilon_{it}) = \max_{a \in A} \left\{ U(a, x_{it}, \varepsilon_{it}) + \beta V(x_{i,t+1}, \varepsilon_{i,t+1}) dF(x_{i,t+1}, \varepsilon_{i,t+1} | a, x_{it}, \varepsilon_{it}) \right\}$$

- Rust's model (by AS, CIX, IID, CLOGIT, discrete support)

$$\bar{V}(x_{it}) = \int \max_{a \in A} \left\{ u(a, x_{it}) + \varepsilon_{it}(a) + \beta \sum_{x_{i,t+1} \in X} \bar{V}(x_{i,t+1}) f_x(x_{i,t+1} | a, x_{it}) \right\} dG_\varepsilon(\varepsilon_{it}) \tag{4}$$

$$= \log \left(\sum_{a=0}^{J} \exp \left(u(a, x_{it}) + \beta \sum_{x_{i,t+1} \in X} \bar{V}(x_{i,t+1}) f_x(x_{i,t+1} | a, x_{it}) \right) \right) \tag{5}$$

Equation 4 is known as the Emax function, with a known form for logit errors.

What Rust's assumptions buys us (2) - factorising the log-likelihood

- Under CIX and IID, the observable state vector x_{it} is sufficient to determine the current choice, and allows the factorisation of the various terms that enter the log-likelihood contribution.

$$\begin{aligned} l_i(\theta) = &\sum_{t=1}^{T_i} \log(Pr(a_{it}|x_{it},\theta)) + \sum_{t=1}^{T_i} \log(f_Y(y_{it}|a_{it},x_{it},\theta_Y)) \\ &+ \sum_{t=1}^{T_i-1} \log(f_X(x_{i,t+1}|a_{it},x_{it},\theta_f)) + \log(Pr(x_{i1}|\theta)) \end{aligned} \tag{6}$$

What Rust's assumptions buys us (3) - functional form for choice probability

- Recall the optimal decision rule is $\alpha(x_{it}, \varepsilon_{it}) = \arg\max_{a \in A}\{v(a, x_{it}) + \varepsilon_{it}(a)\}$, so...

$$\begin{aligned} Pr(a|x,\theta) &\equiv \int I\{\alpha(x,\varepsilon;\theta) = a\} dG_\varepsilon(\varepsilon) \\ &= \int I\{v(a, x_{it}) + \varepsilon_{it}(a) > v(a', x_{it}) + \varepsilon_{it}(a'), \forall a' \neq a\} dG_\varepsilon(\varepsilon_{it}) \\ &= \frac{\exp(v(a,x))}{\sum_{j=0}^{J} \exp(v(j,x))} \end{aligned} \tag{7}$$

- Note slight abuse of notation

$$v(a, x_t) = u(a, x_t) + \beta \sum_{x_{t+1} q \in X} \bar{V}(x_{t+1}) f_x(x_{t+1}|a, x_t)$$

Some examples of the restrictiveness of Rust's assumptions

- **CIX**: Consider an example an observed state variable x_t is human capital and an unobserved state variable is the worker's health ε_t. CIX implies then that health does not affect the rate of accumulation of human capital.
- **CLOGIT and iid**: Choice probabilities exhibit IIA and does not allow for covariance of unobservable state variables - e.g. in occupation choice, unobserved utility are independent between occupations
- **AS**: Implies that marginal utility with respect to observable state variables does not depend on unobservables. e.g. the marginal effect on wage of human capital does not depend on health shocks

Alternative assumptions (e.g. in Eckstein-Keane-Wolpin models)

- Relaxing AS by using non-linear functional forms
- Observable Y variables that are choice-censored and do not satisfy CIY
- Permanent unobserved heterogeneity using mixture models
- Unobservables that are correlated across choice options (departing from CLOGIT)

Outline

Overview

- Based on Rust (1987), Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher, Econometrica
- Harold Zurcher is an employee in a bus company who decides whether to replace a bus engine depending on its current mileage (and other unobserved factors).
- One of the first examples of a dynamic discrete choice model (many online code examples R/Julia/Python/Gauss and *now* Matlab for different solution methods available —check Aguirregabiria's website)

Cost Function

- Harold has to decide in each period t for each bus b whether he wants to replace the engine of the bus.
- He observes a state variable that is observed by the researcher (mileage since last replacement) and others which are not observed by the econometrician (e.g. reports from the drivers).
- Standard cost of maintenance depends on mileage: $c(x_t, \theta_1)$
- If replace engine, incur (net) replacement cost (RC) : $\overline{P} - \underline{P}$ ...
- ...but incur lower maintenance costs $c(0, \theta_1)$

$$U(x_t, i_t, \theta_1) = \begin{cases} -c(x_t, \theta_1) + \varepsilon_t(1) \text{ if } i_t = 0 \\ -[\overline{P} - \underline{P} + c(0, \theta_1)] + \varepsilon_t(0) \text{ if } i_t = 1 \end{cases} \tag{8}$$

State variable transition

- Transition of mileage from period to period is summarised by a stochastic process described by $P(x_{t+1}|x_t, i_t, \theta_2)$
- For example, we may *assume* that the gain in mileage is a draw from a exponential distribution:

$$P(x_{t+1}|x_t, i_t, \theta_2) = \begin{cases} \theta_2 \exp\{\theta_2(x_{t+1} - x_t)\} \text{ if } i_t = 0 \text{ and } x_{t+1} \geq x_t \\ \theta_2 \exp\{\theta_2(x_{t+1})\} \text{ if } i_t = 1 \text{ and } x_{t+1} \geq 0 \\ 0 \text{ otherwise} \end{cases} \tag{9}$$

- Exponential distribution would buy us a closed form solution to the optimal control decision rule, but it is not supported by the data (see Section III of the paper).

Value Function

- Optimal value function:

$$V_\theta(x_t, \varepsilon_t) = \max_{i_t \in \mathcal{C}(x_t)} [u(x_t, i, \theta_1) + \varepsilon_t(i) + \beta EV_\theta(x_t, i_t)] \tag{10}$$

- Emax function:

$$EV_\theta(x_t, i_t) = \int_{-\infty}^{\infty} \int_0^{\infty} V_\theta(y, \varepsilon) p(y|x_t, i_t, \theta_2) dy dG(\varepsilon) \tag{11}$$

$$EV_\theta(x_t) = \int \max_{i_t \in \mathcal{C}(x_t)} \left\{ u(i, x_t) + \varepsilon_t + \beta \int EV_\theta(x_{t+1}) P(x_{t+1}|x_t, i_t) dx_{t+1} \right\} dG_\varepsilon(\varepsilon_t) \tag{12}$$

Eq 11 leads to eq 12 from Rust's assumptions.

The logit assumption

- Usually, would need to compute the double integrals in eq 11, which would be computationally intensive.
- There is a well-known functional form for the Emax function if ε is distributed according to a type I extreme value distribution.

$$EV_\theta(x_t) = \log\left(\sum_{i \in \mathcal{C}(x_t)} \exp\left(u(i, x_t) + \beta \int EV_\theta(x_{t+1}) P(x_{t+1}|x_t, i) dx_{t+1}\right)\right) \tag{13}$$

Discretisation of the observed state space

- For tractability, Rust divides the continuous state space into a discrete set of points.

$$EV_\theta(x_t) = \log\left(\sum_{i\in\mathcal{C}(x_t)} \exp\left(u(i,x_t) + \beta \sum_{x_{t+1}} EV_\theta(x_{t+1})P(x_{t+1}|x_t,i)\right)\right) \tag{14}$$

- For a problem with an infinite time horizon and a finite state space, we can solve for the emax functions as the unique solution to a finite system of equations

$$\bar{\boldsymbol{V}} = \log\left(\sum_{a=0}^{J} \exp\left\{\boldsymbol{u}(a,\theta) + \beta \boldsymbol{F}_x(a)\bar{\boldsymbol{V}}\right\}\right) \tag{15}$$

Deriving the Emax values by contraction mapping

- Rust (1987) shows that equation 14 is a contraction mapping from $\bar{\boldsymbol{V}} \rightarrow \bar{\boldsymbol{V}}$.
- A straightforward, though computationally complicated approach is to iterate values of the policy functions until the difference between iterations is sufficiently small.

$$\bar{\boldsymbol{V}}_{h+1} = \log\left(\sum_{a=0}^{J} \exp\left\{\boldsymbol{u}(a,\theta) + \beta \boldsymbol{F}_x(a)\bar{\boldsymbol{V}}_h\right\}\right) \tag{16}$$

- Note: For finite horizon applications, backward induction is most common solution method (see example later).

Checklist (1)

1. For each trial value of the parameters θ, compute the value of the Emax functions at all points in X:
   - For infinite horizon problems, use the contraction mapping approach described.
   - For finite time horizons, use backward induction...i.e. compute $EV_\theta^T(x)$ for all values of x, then work backwards
2. Compute the log-likelihoods via equation 6

$$l_i(\theta) = \sum_{t=1}^{T_i} \log(Pr(a_{it}|x_{it},\theta)) + \sum_{t=1}^{T_i-1} \log(Pr(x_{i,t+1}|a_{it},x_{it},\theta_f)) + \log(Pr(x_{i1}|\theta))$$

, where $Pr(a_{it}|x_{it},\theta) = \frac{\exp(u(a_{it},x_{it},\theta_u)+\beta \mathbf{F}_x(a_{it},x_{it};\theta_f)'\bar{\mathbf{V}}(\theta))}{\sum_{j=0}^{J}\exp(u(j,x_{it},\theta_u)+\beta \mathbf{F}_x(j,x_{it};\theta_f)'\bar{\mathbf{V}}(\theta))}$.

Checklist (2)

3. Compute iteration step according to equation 17. (Or just plug into your favourite optimiser.)

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \left(\sum_{i=1}^{N} \frac{\partial \ell_i(\hat{\theta}_k)}{\partial \theta} \frac{\partial \ell_i(\hat{\theta}_k)}{\partial \theta'} \right)^{-1} \left(\sum_{i=1}^{N} \frac{\partial \ell_i(\hat{\theta}_k)}{\partial \theta} \right) \quad (17)$$

Note that for these functional forms, analytical derivatives ($\frac{\partial \ell_i(\hat{\theta}_k)}{\partial \theta}$) are available and can smoothen out the optimisation process. See Aguirregabiria and Mira (2010) for details! (NOT INCLUDED IN CODE DEMO TODAY)

Outline

Examples in Matlab

Goal: Estimate the replacement cost, maintenance costs parameters, and the transition probabilities for the state variable.

Examples in Matlab

Goal: Estimate the replacement cost, maintenance costs parameters, and the transition probabilities for the state variable.

Code is in:
`https://github.com/bzdiop/structuralwg/tree/main/presentations/code/SHGS/rust_code`

Examples in Matlab

Goal: Estimate the replacement cost, maintenance costs parameters, and the transition probabilities for the state variable.

Code is in:
`https://github.com/bzdiop/structuralwg/tree/main/presentations/code/SHGS/rust_code`

There are two sets of scripts:

- finite time horizon (with suffix `_finite`)
- infinite time horizon (with suffix `_inf`), which we illustrate first, because it is what the paper implements.
  - Maybe obvious point: The model assumes the agent makes decisions indefinitely, but the estimation only requires a finite panel.

Code structure 1/2

`generate_data_inf.m` : Only need it run this once, but illustrative as it generates the data according to the structural model (so you can see "all the pieces of the puzzle"). The script:

1. initialises the parameters and static utilities
2. discretises the state-variable and creates the transition probabilities,
3. evaluates the value function for each choice
4. calculates conditional choice probabilities
5. simulates a synthetic dataset

Code structure 1/2

`generate_data_inf.m` uses the following functions:

- `cost.m` : Defines a piecewise function that gives the utility for undertaking each option
  - `c.m` : Defines a linear function used as an input to `cost.m`.
- `inner_algo.m` : Executes the inner-algorithm based on the contraction mapping to converge to the value functions in the infinite horizon problem
  - `iteration.m` : Performs an iteration on the vector of Emax values (for each value of x) using the Emax function, as in 16.

Code to initialise the parameters and static utilities

generate_data_inf.m

```
% Parameter values taken from col 2 of table 9 in Rust (1987)
%beta = 0.999;                % discount factor, could be smaller to speed up convergence
%p_x0 = 0.3919;               % probability of making [0-5000) miles
%p_x1 = 0.5953;               % probability of making [5000-10000) miles
%p_x2 = 1- p_x0 - p_x1;       % probability of making [10000 - \infty) miles

pars = [10.0750;0.00005293;0.3919;0.5953];   % Starting values for data generation
rc = pars(1);               % Replacement cost
theta1_1 = pars(2);         % Maintenance cost paramater with a linear cost function
p_x0 = pars(3);             % Transition probs
p_x1 = pars(4);

beta = 0.8;                 % Discount factor
p_x2 = 1- p_x0 - p_x1;
it_tol = 1e-6;             % tolerance paramater to define minimum distance in contraction mapp

x_grid = transpose(0:5000:350000); %Set up discretised state space of x

% Vector of costs in each period, for each choice
u_1 = arrayfun(@(x) cost(1,x,pars),x_grid); % Compute u(a,x) for a = 1 (71 times 1)
u_0 = arrayfun(@(x) cost(0,x,pars),x_grid); % Compute u(a,x) for a = 0 (71 times 1)
```

Create matrix of transition probabilities for all possible states

generate_data_inf.m

```
% Create matrix Fx_0 (71 times 71)
% Remember that this is because you may transition from any state to any
% other state, but the probabilities of transitioning to most states are set to zero,
Fx_0 = zeros(length(x_grid),length(x_grid));
for i = 1:length(x_grid)
    Fx_0(i,i) = p_x0;
    if i <= length(x_grid) - 1
        Fx_0(i,i+1) = p_x1;
    end
    if i <= length(x_grid) - 2
        Fx_0(i,i+2) = p_x2;
    end
end
Fx_0(length(x_grid)-1,length(x_grid)) = 1- p_x0;
Fx_0(length(x_grid),length(x_grid)) = 1;

% Create matrix Fx_1 (71 times 71)
Fx_1row = zeros(1,length(x_grid));
Fx_1row(1) = p_x0;
Fx_1row(2) = p_x1;
```

Matrix of transition probabilities given no replacement

Fx_0

71x71 double

	1	2	3	4	5
1	0.3919	0.5953	0.0128	0	0
2	0	0.3919	0.5953	0.0128	0
3	0	0	0.3919	0.5953	0.0128
4	0	0	0	0.3919	0.5953
5	0	0	0	0	0.3919
6	0	0	0	0	0
7	0	0	0	0	0

Matrix of transition probabilities given no replacement

- At zero miles (row 1), there is a 0.3919 probability of only doing 0-5000 miles (cell 1,1), a 0.5953 probability of doing 5001-10000 miles, or a 0.0128 probability of doing 10000-∞ miles.
- **By assumption**, we are bounding the steps to be up to 15000 miles.
- This means that there is zero probability of going from zero to beyond 15000 miles.
- There is a zero probability of going back from 5000-10000 to 5000-10000 (cell 2,1).

Contraction mapping algorithm to evaluate following 16

inner_algo.m

```
function [Vbar_1] = inner_algo(Fx_1,Fx_0,u_0,u_1,beta,it_tol)
    Vbar_1 = log(exp(u_0) + exp(u_1)); % Start from the static case
    max_itdiff = 1;                    % Start with a value of the difference between iterations
    it_counter = 0;
        while max_itdiff > it_tol
            Vbar_0 = Vbar_1;           % Update the current Vbar with the previously computed t+1
            Vbar_1 = iteration(Vbar_0,Fx_1,Fx_0,u_0,u_1,beta); % Update the value of Vbar
            it_diff = abs(Vbar_1 - Vbar_0);
            max_itdiff = max(it_diff);
            %display(max_itdiff)
            it_counter = it_counter + 1;
            %display(it_counter)
        end
end
```

Code structure 2/2

`estimate_inf.m` : Uses the dataset to optimise the log-likelihood function in order to estimate the structural parameters, their standard errors, and confidence intervals.

- `rust_loglik_inf.m`
  1. performs similar steps as `generate_data_inf.m` to generate the choice probabilities using the inner algorithm, but using different parameters for each iteration of the optimising algorithm.
     - Uses the same functions to get to the choice probabilities: `cost.m` , `c.m` , `inner_algo.m`, `iteration.m`
  2. Computes the likelihood for the choice probabilities and the transition probability components.

Estimation steps in the demo

1. Start with guess $\hat{\theta}_0$.
2. Evaluate $\bar{\boldsymbol{V}}(\hat{\theta}_0)$, either by iteration until convergence using the contraction mapping (see equation 15 or backwards induction for the finite case).
3. Use $\bar{\boldsymbol{V}}(\hat{\theta}_0)$ and parameter guesses $\hat{\theta}_0$ to compute the choice probability, $P(a|x,\theta)$.
4. Evaluate the log-likelihood and repeat until a local optimum has been reached.

Rust's log-likelihood

rust_loglik_inf.m

```
    end
    % Likelihood function for all the structural paramaters
    function [sumloglik] = loglik(data,choiceprob_1,choiceprob_0,Fx_0,Fx_1)
        N = size(data,1);
        loglikvec = zeros(N,1);
        for i = 1:N
            loglikvec(i) = choiceprob(data(i,1),data(i,2),choiceprob_1,choiceprob_0) + ...
                           transprob(data(i,2),data(i,3),data(i,1),Fx_0,Fx_1);
        end
        sumloglik = -sum(loglikvec);
    end
```

Data contains binary choice to replace in column 1, current mileage in column 2, and next period's mileage in column 3.

Components of the log-likelihood

rust_loglik_inf.m

```
    % Likelihood component for the choice probability
    function [logchoiceprob] = choiceprob(a,x,choiceprob_1,choiceprob_0)
        modx = floor(x./5000)+1;
        if a == 1
            logchoiceprob = log(choiceprob_1(modx));
        elseif a == 0
            logchoiceprob = log(choiceprob_0(modx));
        else
            error('a should be 0 or 1.')
        end
    end
    % Likelihood component for the transition probability
    function [logtransprob] = transprob(x,x_f,a,Fx_0,Fx_1)
        modx = floor(x./5000)+1;
        modxf = floor(x_f./5000)+1;
        if a == 1
            logtransprob = log(Fx_1(modx,modxf));
        elseif a == 0
```

Outline

Finite horizon introduction

- Suppose that a planner runs a bus service, knowing at the start that his franchise ends in 12 periods.
- This then becomes a finite time horizon problem, which is typically solved by **backwards induction**.

The final period

- First, consider the last period T. The value of each choice j in T is simply U_j, given the value of the state at that time x_T.
- The expected value at time T given state value x_T is thus the E-max function:

$$EV_T(x_T) = \int \max_{a_T \in \{0,1\}} \{u(x_T, a_T) + \varepsilon_T(a_T)\}\, dG_\varepsilon(\varepsilon_T) \tag{18}$$

$$= \log\left(\sum_{a=0}^{J} \exp(u_T(a, x_T))\right) \tag{19}$$

- In matrix form:

$$\bar{\boldsymbol{V}}_T(\hat{\theta}) = \log\left(\sum_{a=0}^{J} \exp(\boldsymbol{u}_T(a, \hat{\theta})\right) \tag{20}$$

The penultimate period $(T-1)$

- The value of choosing any option j is now $U_j(a_{T-1}, x_{T-1}) + \beta \mathbf{F}_{x,t}(a, x_{T-1})\bar{\mathbf{V}}_T(\hat{\theta})$.
- The expected value at T-1 given state value x_{T-1} is again the E-max function, this time with the implications for the future period included

$$EV_{T-1}(x_{T-1}) = \int \max_{a_{T-1} \in \{0,1\}} \left\{ U(x_{T-1}, a_{T-1}) + \beta \mathbf{F}_{x,t}(a, x_{T-1})\bar{\mathbf{V}}_T(\hat{\theta}) \right\} dG_{\varepsilon}(\varepsilon_{T-1}) \tag{21}$$

$$= \log\left(\sum_{a=0}^{J} \exp(u_{T-1}(a, x_{T-1}) + \beta \mathbf{F}_{x,t}(a, x_{T-1})\bar{\mathbf{V}}_T(\hat{\theta})) \right) \tag{22}$$

Backwards induction

- In general, for the Rust assumptions, in matrix form, the values can be solved for recursively using the following expression.

$$\bar{\boldsymbol{V}}_t(\hat{\theta}) = \log\left(\sum_{a=0}^{J} \exp\left\{\boldsymbol{u}_t(a,\hat{\theta}) + \beta \boldsymbol{F}_{x,t}(a)\bar{\boldsymbol{V}}_{t+1}(\hat{\theta})\right\}\right) \tag{23}$$

- These solved values can then be used to evaluate the probability of observing the given choices:

$$Pr_t(a|\hat{\theta}) = \frac{\exp(u_t(a,\hat{\theta}) + \beta \boldsymbol{F}_{x,t}(a)\bar{\boldsymbol{V}}(\hat{\theta}))}{\sum_{j=0}^{J} \exp(u_t(j,\hat{\theta}) + \beta \boldsymbol{F}_{x,t}(j)\bar{\boldsymbol{V}}(\hat{\theta}))} \tag{24}$$