Dynamic Discrete Choice Models and Machine Learning: Methods and Applications to Marketing

Hema Yoganarasimhan [1]

University of Washington

December 1, 2018

[1]Please address all correspondence to: hemay@uw.edu.

Contents

Chapter 1

Single Agent Dynamic Discrete Choice Models

1.1 Why Dynamics?

In a static choice model, we implictly assume that consumers make choices today without regard to the future, that is, they are assumed to be myopic. In many settings, modeling consumers as myopic is either correct or a sufficiently close approxmation of their true behavior that this is not a problem. Examples of such choices are, whether to take the bus or car to work on a given day, which flavor of ice-cream to buy now, whether to click on an online advertisement or not, and so on. However, in many settings, consumers' decisions today have significant consequences for their future – both in terms of the choices they would face in the future as well as their future utility stream. Intuitively, in static choice models, consumers only trade-off the utility from different alternatives today. In dynamic choice models, consumers also trade-off utility today and utility in the future. This is referred to as "Inter-temporal Choices" in Marketing and Economics. When making such choices, consumers tend to be "forward-looking", i.e., they base their decision on not only today's utility, but also their expectation of how today's decision will influence their future utility stream. A few canonical examples of such choice-situations are:

- *Bus engine replacement trade-off*: Pay a high replacement cost today and low maintenance cost in future vs. paying no replacement cost today, but high maintainence cost in the future (Rust, 1987).
- *Durable goods trade-off*: Pay a high price (for a given quality) today and enjoy the product for sooner vs. delay gratification for lower price in the future (Song and Chintagunta, 2003).
- *Storable goods trade-off*: If you stockpile during a promotion, you save from paying low prices today, but pay higher storage cost in the future vs. if you don't stockpile, you don't pay higher prices in the future, but also pay lower storage costs (Hendel and Nevo, 2006).
- *Retirement Problem trade-off*: Retire earlier, which gives a lower yearly income, but for longer period vs. retire later, which gives a higher yearly income, but for shorter period of time (Keane and Wolpin, 1994).

- *Dynamic Auctions trade-off*: If you pick a bidder today, you can get the job done sooner but the bidder may not be the highest quality (for the price) vs. if you wait, you may receive a better bid (higher quality/lower price) and delay gratification (Yoganarasimhan, 2013)

Ignoring dynamics in these kinds of settings by assuming that consumers are myopic and using a static choice model will lead to bias in parameter estimates. For example, using a static logit model to estimate the effectiveness of price-promotions will lead to over-estimation of demand elasticity in the storable goods case because the static model ignores the correlation between current utility from consumption and the expected future utility from consumption the good. Thus, not including the latter in the model causes omitted variable bias.

1.2 Model Set up

We now expand the standard static discrete choice modeling framework to allow for dynamics. Throughout, we will follow the framework proposed by Rust (1987), and often explicitly use the bus-engine replacement example to aid exposition.

We consider the problem from the perspective of a single agent, denoted by i. There are $i = 1, \ldots, N$ agents observed in the data. Time is discrete and denoted by t. We observe $t = 1$ to T periods in the data. In every period, the agent chooses between $j = 1 \ldots J$ options. i's decision in period t is denoted by d_{it}, and $d_{it} = j$ indicates that agent i has chosen option j in period t.

1.2.1 State Variables

There are two types of state variables that affect agent i's decision in period t

- Observable state variables x_{it}, which is observable to both the agent and the researcher.
- Unobservable state variable ϵ_{it}, which is a $J \times 1$ vector with support $\mathbb{R}^J$, where the j^{th} component forms the mean-zero error associated with option j at time t. Note that ϵ_{it} is unobservable only to the researcher, not the agent.

Assumption 1. *Discrete State Space*
The observable state variables x_{it} are assumed to belong to a finite set, whose dimensionality is X.

1.2.2 Data

The data that we observe is a sequence of decisions and states for T periods:
$\{(d_{i1}, x_{i1}), (d_{i2}, x_{i2}), \ldots, (d_{it}, x_{it}), \ldots, (d_{iT}, x_{iT})\}$ for each $i \in \{1, \ldots, N\}$

The goal of the modeling exercise to estimate a set of structural parameters that rationalize the decisions and the states observed in the data.

1.2.3 Flow Utility

Each period the agent i derives an instantaneous utility, $u(d_{it}, x_{it}, \epsilon_{it})$, in period t which is a function of his decision d_{it} and the state variables $\{x_{it}, \epsilon_{it}\}$.

Assumption 2. *Additive Separability*
Error terms are assumed to enter the flow utility additively separably as follows:

$$u(d_{it} = j, x_{it}, \epsilon_{ijt}) = \bar{u}(d_{it} = j, x_{it}; \theta_1) + \epsilon_{ijt} \ \ \forall \ \ t \tag{1.1}$$

where ϵ_{ijt} is the error term associated with the j^{th} option in period t, $\bar{u}(d_{it} = j, x_{it}; \theta_1)$ is the deterministic portion of the flow utility for decision j and state x_{it}, and θ_1 are the structural parameters associated with the flow utility. This is the standard additive separability assumption commonly invoked in the literature on discrete choice models.

Flow Utility in Rust Bus Engine Example:
There are only two possible decisions in this case: a) Continue without replacing, denoted by $d_{it} = 0$, and b) Replace the bus engine, denoted by $d_{it} = 1$. Let $x_{it} \in \{0, 1, \ldots, M\}$ denote the mileage of the bus, where M denotes maximum mileage for a bus. Further, assume that this is the only observable state variable that affect HZ's decision to replace the engine. If we assume that maintainence costs are linear in mileage and there is a fixed replacement, then we have:

$$u(d_{it} = 0, x_{it}, \epsilon_{i0t}) = -c_m x_{it} + \epsilon_{i0t} \tag{1.2}$$

$$u(d_{it} = 1, x_{it}, \epsilon_{i1t}) = -c_r + \epsilon_{i1t} \tag{1.3}$$

where $\theta_1 = \{c_m, c_r\}$. There is no intercept in the utility from continuing because we can only identify one intercept term. So c_r should be interpreted as the additional fixed cost of replacement, over and above maintaining.

1.3 State Transitions

The states $\{x_{it}, \epsilon_{it}\}$ transition into new values every period. We make a few key assumptions about this state transition process.

Assumption 3. *First Order Markov Process*
The state variables in period $t{+}1$, $\{x_{it+1}, \epsilon_{it+1}\}$, *only depend on* $\{d_{it}, x_{it}, \epsilon_{it}\}$, *and not states/decisions from* $t-1$ *and earlier.*

$$Pr(x_{it+1}, \epsilon_{it+1} | d_{it}, x_{it}, \epsilon_{it}) = Pr(x_{it+1}, \epsilon_{it+1} | d_{it}, x_{it}, \epsilon_{it}, d_{it-1}, x_{it-1}, \epsilon_{it-1} .....) \tag{1.4}$$

This assumption implies that today's state and decisions are a sufficient statistic for tomorrow's state. Thus, we do not have to carry forward a long series of states and decisions to model today's outcomes.

Assumption 4. *Conditional Independence Assumption*
We make two types of conditional independence assumptions on state transitions. First, conditional

on xs, the error terms, ϵs, are independent over time. Second, conditional on $\{d_{it}, x_{it}\}$, x_{it+1} is independent of ϵ_{it}.

$$\begin{aligned} Pr(x_{it+1}, \epsilon_{it+1}|d_{it}, x_{it}, \epsilon_{it}) &= Pr(\epsilon_{it+1}|x_{it+1}, d_{it}, x_{it}, \epsilon_{it}) \cdot Pr(x_{it+1}|d_{it}, x_{it}, \epsilon_{it}) \\ &= Pr(\epsilon_{it+1}|x_{it+1}) \cdot Pr(x_{it+1}|d_{it}, x_{it}) \end{aligned} \tag{1.5}$$

The first part of this assumption implies that errors are independent over time, conditional on the observable state variable x_{it}. The second part implies that errors today affect states tomorrow only through decisions today. These constraints on how the state variables evolve simplify the Log Likelihood, as we will see in §1.5.

1.3.1 Transition of Observable State Variables

The state transition probability of observable state variables are assumed to be:

$$g(x_{it+1}|x_{it}, d_{it}; \theta_2) \tag{1.6}$$

where θ_2 is a set of parameters which defines the state transition. Some subset of state variables, x_{it}^1, can also be time-invariant, in which case, $g(x_{it+1}^1|x_i, d_{it}) = x_{it}^1 \ \forall \ t$.

State Transitions in the Bus Engine Example:
We consider a deterministic state transition. If HZ chooses to continue in period t, then:

$$x_{it+1} = x_{it} + 1 \ \text{ if } \ x_{it} < M \ \ \& \ \ d_{it} = 0 \tag{1.7}$$

$$x_{it+1} = M \ \text{ if } \ x_{it} = M \ \ \& \ \ d_{it} = 0 \tag{1.8}$$

Mileage increases by one mile each period if the engine is not replaced. If the bus is already at the maximum mileage M, then the mileage does not increase anymore. If HZ chooses to replace in period t, then the mileage goes to zero at the beginning of t. So in period $t + 1$, we have:

$$x_{it+1} = 1 \ \text{ if } \ d_{it} = 1 \tag{1.9}$$

1.3.2 Transition of Unobservable State Variables

The unobservable state variables, ϵ_{it}s, are assumed to be i.i.d across time and independent of the observable states.

Assumption 5. *IID Error Terms*

The unobservable state variables are i.i.d, as follows:

$$Pr(\epsilon_{it}|x_{it}) = Pr(\epsilon_{it}) \tag{1.10}$$

This is an important and strong assumption on how we think about unobservables. Implictly, we are ruling out the possibility of persistence in unobservables that affect agents' decisions. For example, if HZ observes the brand of a bus and the researcher does not. If different brands of buses have different maintainence costs, then this unobservable (brand) will enter the error term. Thus, we would have correlation in maintainence costs and ϵ_{it}, leading to bias in our estimate of maintence costs. In §5 and §6, we examine how to relax this assumption and allow for serial correlation in unobservables.

Further in most applications (and throughout in this book), the error terms assumed to be are drawn from a Type 1 Extreme Value Distribution. All the methods described are easily modified to accomodate a Generalized Extreme Value (GEV) distribution.

Conceptually, it is possible to allow for other error distributions. See Keane and Wolpin (1994) for an example of a DDC model with normal error terms. However, the advantage of GEV errors is that they can analytically intergrated out, which gives us closed form choice probabilities. This greatly simplifies the computational burden. Moreover, the Two-Step estimation methods discussed in §3 and §6 rely on an analytical relationship between choice probabilities

1.4 Bellman Equation

The value of being in a state $\{x_{it}, \epsilon_{it}\}$ is the discounted sum of being in this state and taking optimal decisions in the future. Thus, we define the value function $V(x_{it}, \epsilon_{it})$ as follows:

$$V(x_{it}, \epsilon_{it}) = \max_{d_{it}, d_{it+1}, \ldots} E \sum_{\tau=t}^{\infty} \beta^{\tau - t} \left[u(d_{i\tau}, x_{i\tau}, \epsilon_{i\tau}; \theta_1 | x_{it}, \epsilon_{it}) \right] \tag{1.11}$$

For a certain class of dynamic programming problems known as 'stationary problems', the value function can be written in a recursive form as follows:

$$\begin{aligned} V(x_{it}, \epsilon_{it}) &= \max_{j=1,\ldots,J} [u(d_{it} = j, x_{it}, \epsilon_{ijt}; \theta_1) \\ &+ \beta \int \int V(x_{it+1}, \epsilon_{it+1}) g(x_{it+1} | x_{it}, d_{it} = j; \theta_2) \mathrm{d}x_{it+1} f(\epsilon_{it+1}) \mathrm{d}\epsilon_{it+1}] \end{aligned} \tag{1.12}$$

where $f(\epsilon_{it+1})$ is the distribution of error terms (here Type 1 extreme value distribution). The above recursion is also known as the Bellman equation.

To help with the exposition, we also define a choice-specific value function as the value of

being in state $\{x_{it}, \epsilon_{it}\}$ conditional on taking decision $d_{it} = j$

$$\begin{aligned}\tilde{V}(x_{it}, \epsilon_{ijt}, d_{it} = j) &= u(d_{it} = j, x_{it}, \epsilon_{ijt}; \theta_1) \\ &+ \beta \int \int V(x_{it+1}, \epsilon_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j; \theta_2) \mathrm{d}x_{it+1} f(\epsilon_{it+1}) \mathrm{d}\epsilon_{it+1}\end{aligned} \tag{1.13}$$

Similarly, we can define the expected choice specific value function as follows:

$$\begin{aligned}v(x_{it}, d_{it} = j) &= \bar{u}(d_{it} = j, x_{it}; \theta_1) \\ &+ \beta \int \int V(x_{it+1}, \epsilon_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j; \theta_2) \mathrm{d}x_{it+1} f(\epsilon_{it+1}) \mathrm{d}\epsilon_{it+1}\end{aligned} \tag{1.14}$$

where $\bar{u}(d_{it} = j, x_{it})$ is the expected flow utility from choosing $d_{it} = j$ in state x_{it} (i.e., intergrated over ϵ_{ijt}).

Note on stationary vs. non-stationary problems – This recursive formulation is possible only for stationary dynamic programming problems. A stationary problem satisfies two key properties – 1) It is infinite horizon, 2) Time t is not a state variable, i.e., the time period t does not directly enter state transitions and flow utilities. In contrast, in a non-stationary problem, time enters the value function as a state variable. Finite-horizon problems usually fall under this category. The solution concepts for these two classes of problems are quite different. In this chapter, we focus on stationary dynamic programming problems.

1.5 Log Likelihood

To estimate the model, we first need to define the likelihood of observing the data conditional on the structural parameters. The likelihood of observing a series of
$\{d_i, x_i\} = \{(d_{i1}, x_{i1}), (d_{i2}, x_{i2}), \ldots, (d_{it}, x_{it}), \ldots, (d_{iT}, x_{iT})\}$ for each $i \in \{1, \ldots, N\}$ is given by:

$$\begin{aligned}L_i(d_{i1}, x_i|\theta_1, \theta_2) &= \prod_{t=1}^{T} Pr(d_{it}, x_{it}|d_{it-1}, x_{it-1}, \ldots, d_{i1}, x_{i1}; \theta_1, \theta_2) \\ &= \prod_{t=1}^{T} Pr(d_{it}, x_{it}|d_{it-1}, x_{it-1}; \theta_1, \theta_2) \\ &= \prod_{t=1}^{T} Pr(d_{it}|x_{it}; \theta_1) \cdot Pr(x_{it}|d_{it-1}, x_{it-1}; \theta_2)\end{aligned} \tag{1.15}$$

Assumption 3 helps us go from the first line to the second line and Assumption 4 helps us go to the third and fourth lines in the above set of equations.

Since the two probabilities in likelihood can be Equation (1.15) are independent of each other,

the log likelihood is additively separable as follows:

$$LL_i(d_{i1}, x_i|\theta_1, \theta_2) = \sum_{t=1}^{T} \log Pr(d_{it}|x_{it}; \theta_1) + \sum_{t=1}^{T} \log Pr(x_{it}|d_{it-1}, x_{it-1}; \theta_2) \quad (1.16)$$

The main advantage of this additive separability is that we can split the estimation of θ_1 and θ_2. We discuss the estimation procedue in the next chapter.

Chapter 2

Nested Fixed Point Algorithm

We now consider the estimation of the problem defined in the preceding chapter. The estimation proposed by Rust (1987) follows a two-step procedure:

- Step 1: Estimate the state transition parameters θ_2. This can be fully non-parametric or if the researcher can impose parametric assumptions if she believes it to be appropriate.
- Step 2: We use the estimates of $\hat{\theta}_2$ from Step 1 and estimate the utility parameters θ_1

For now, we assume that the discount factor β is specified by the researcher. In §7, we discuss the estimation and identification of the discount factor.

2.1 Log Likelihood of the Decisions

With the assumption of i.i.d Type 1 extreme value errors, we can write the choice probabilities as:

$$Pr(d_{it} = j|x_{it};\theta_1) = \frac{\exp[v(x_{it}, d_{it} = j)]}{\sum_j \exp[v(x_{it}, d_{it} = j)]} \tag{2.1}$$

$$= \frac{\exp\left[\bar{u}(d_{it} = j, x_{it};\theta_1) + \beta \int\int V(x_{it+1}, \epsilon_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j;\theta_2) \mathrm{d}x_{it+1} f(\epsilon_{it+1}) \mathrm{d}\epsilon_{it+1}\right]}{\sum_j \exp\left[\bar{u}(d_{it} = j, x_{it};\theta_1) + \beta \int\int V(x_{it+1}, \epsilon_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j;\theta_2) \mathrm{d}x_{it+1} f(\epsilon_{it+1}) \mathrm{d}\epsilon_{it+1}\right]}$$

$$= \frac{\exp\left[\bar{u}(d_{it} = j, x_{it};\theta_1) + \beta \int \bar{V}(x_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j;\theta_2) \mathrm{d}x_{it+1}\right]}{\sum_j \exp\left[\bar{u}(d_{it} = j, x_{it};\theta_1) + \beta \int \bar{V}(x_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j;\theta_2) \mathrm{d}x_{it+1}\right]}$$

where $\bar{V}(x_{it+1}) = \int V(x_{it+1}, \epsilon_{it+1}) f(\epsilon_{it+1}) \mathrm{d}\epsilon_{it+1}$ is also referred to as the expected value function.

The above expression is often referred to as the 'dynamic logit'. Note that this is similar to the standard logit-style models that we estimate in static settings, except for the future value terms. The key idea behind the estimation is that if we obtain accurate numerical estimates of the value functions (and assume the discount factor), then we can estimate the structural parameters in a standard logit model.

2.2 Iterative Two-Step Procedure

The nested fixed point consists of an iterative two-step procedure, where:

- Outer Loop: Search over the parameter space for θ_1.
- Inner Loop: At each iteration, for the current value of θ_1, compute the expected choice-specific value function.

We now expand on both these steps further. One of the main ideas in Rust (1987) is that instead of iterating over the actual value functions, we can iterate over the expected value functions to avoid intergrating over the error terms. Specifically, we can use the following recursion:

$$\bar{V}(x_{it}) = \log\left[\sum_j \exp\left(\bar{u}(x_{it}, d_{it} = j; \theta_1) + \beta \int \bar{V}(x_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j; \theta_2) dx_{it+1}\right)\right] \tag{2.2}$$

So the algorithm runs as follows:

1. First, assume a value for θ_1^1, where the super-script refers to the iteration number. These can usually be initiated from the estimation of a static model to aid convergence.

2. For the inner loop, start with $\bar{V}^1(x) = 0$ for all xs. Then for the given value of θ_1 and the estimated $\hat{\theta}_2$, recurse over Equation (2.3) till the $\bar{V}^1(x)$ converge. The Bellman equation is a contraction mapping and has a unique solution for a given θ_1. Hence this recursion should converge to a unique value irrespective of the starting values used. That is:

$$\bar{V}^1(x_{it}) = \log\left[\sum_j \exp\left(\bar{u}(x_{it}, d_{it} = j; \theta_1^1) + \beta \int \bar{V}^1(x_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j; \hat{\theta}_2) dx_{it+1}\right)\right] \tag{2.3}$$

3. Then for the outer loop, take $\bar{V}^1(x)$ as given and plug them into the Log Likelihood given by Equation (2.2). Then using a simple maximum likelihood procedure, estimate θ_1^2, which are the updated values of θ_1 in the second iteration. That is:

$$\theta_1^2 = \arg\max_{\theta_1^2} \sum_{t=1}^{T} \log \frac{\exp\left[\bar{u}(d_{it} = j, x_{it}; \theta_1^2) + \beta \int \bar{V}^1(x_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j; \theta_2) \mathrm{d}x_{it+1}\right]}{\sum_j \exp\left[\bar{u}(d_{it} = j, x_{it}; \theta_1^2) + \beta \int \bar{V}^1(x_{it+1}) g(x_{it+1}|x_{it}, d_{it} = j; \hat{\theta}_2) \mathrm{d}x_{it+1}\right]} \tag{2.4}$$

4. Repeat steps 2-3 till your paramter estimates θ_1s converge.

An alternative method of iteration is to iterate over choice-specific expected value functions.

Finally, it is important to note that the standard errors for θ_1 derived from any optimization procedure are usually downward biased because we are using an estimate of the state transition

paramters $\hat{\theta}_2$ instead of the true parameter vector θ_2. The standard solution to this problem is to bootstrap. When the error-terms are i.i.d, as in this case, we can bootstrap by sampling with replacement over observations. However, it is best practice to sample over agents or over i because when the i.i.d assumption fails, the likelihood is not additively separable for observations within an agent.

Chapter 3

Two-Step Methods

Chapter 4

Finite Horizon Problems and Backward Induction

Chapter 5

Persistent Unobservables

Thus far, we have assumed that all the unobservables (ϵ_{ijt}s) are i.i.d, or that they do not persist with time. However, this is not always true. For example, in the Rust bus engine case, different brands of buses may have different replacement costs. The brand of the bus thus will affect Harold Zurcher's replacement decision in a systematic fashion. Moreover, the brand of the bus remains the same in every period. Similarly, in Yoganarasimhan's dynamic auction setting, the quality of the outside option of a buyer will persist through time and remains unobservable to us.

To accommodate such states, we now introduce the concept of persistent unobservables. These are state variables that persist with time, are observable to the decision-maker, but not to the researcher. There can be two types of persistent unobservables

- Time invariant (s_i): remain constant with time.
- Time varying (s_{it}): usually assumed to evolve in an exogenously defined AR(1) process.

In this chapter, we will focus on time invariant state persistent state variables. The problem of time-varying serially correlated states is slightly more complicated.

Persistent unobservables cause the problem of "Dynamic Selection", which implies that the set of agents in any period is not random, but rather self-selected. For example, in survival models (durable goods purchase or product adoption), the agents surviving at any given time period are now systematically different from the set of agents that started at $t = 1$. Consider the bus engine replacement problem:

- Without s_i, buses with high mileage are not systematically different from buses with low mileage at time period t. In other words, all buses will be seen in both high mileage and low mileage states with equal probability, when observed for a large enough period of time.
- In contrast, suppose we have two brands of buses in the data (each with a share of 50%), one with low maintainence cost (H quality) and another with high maintainence cost (L quality) and the brand is unobservable to us. Then, we have the following problem:
    - buses with high mileage are systematically different from those with low mileage – more likely to be of H quality brand.

- buses with high mileage are thus a "selected" sample, and not a random sample of all the buses in the data. At high mileages, Pr(bus = H quality) $> 50\%$.

Econometrically, persistent unobservables lead to serial correlation in error terms. The unobserved state s_i is now in the residual and we have:

$$E(\epsilon_{ijt} \cdot \epsilon_{ijt+1}) \neq 0 \ \ \forall \ \ i, t \tag{5.1}$$

Thus, ϵ_{ijt}s are no longer i.i.d over time, which implies that the choice probability expressions we derived earlier are now invalid (since they relied on the i.i.d assumption).

5.1 Finite Mixture Models

Arcidiacono and Jones (2003) proposed a finite mixture based solution to the problem of persistent unobservables. They showed that we can combine Rust's nested fixed point with the EM algorithm to accommodate persistent unobservables in this setting. In the rest of this chapter, we describe their method.

Assume that in addition to the time-varying i.i.d error term ϵ_{it}, there is another persistent unobserved state variable s_i that is drawn from a set of K types such that $s_i \in \{1, 2, \ldots, K\}$, where the population probability of k^{th} type is π_k. By definition, $\sum_{k=1}^{K} \pi_k = 1$.

5.1.1 Flow Utilities and State Transitions

The flow utility is still assumed to be additively separable in ϵ_{it}, but not in s_i. That is:

$$u(d_{it}, x_{it}, s_i, \epsilon_{it}; \theta_1) = \bar{u}(d_{it}, x_{it}, s_i; \theta_1) + \epsilon_{it} \tag{5.2}$$

As before, $\bar{u}(\cdot)$ denotes the deterministic portion of the flow utility.

Similarly, the state transitions can also be a function of the unobserved state as follows:

$$g(x_{it}|x_{it-1}, s_i, d_{it-1}; \theta_2) \tag{5.3}$$

Note that the state transitions are now directly observable from data as in the previous case, since they are now functions of the unobservable state variable s_i. This is an important difference and precludes the possibility of estimating $g(\cdot)$ in an intial first step as we did in the previous case.

In some special cases, it is possible to relax the dependence of $g(\cdot)$ on s_i, in which case we can assume that:

$$g(x_{it}|x_{it-1}, s_i = k, d_{it-1}; \theta_2) = g(x_{it}|x_{it-1}, d_{it-1}; \theta_2) \ \ \forall \ \ k \tag{5.4}$$

As we will see in 5.2.3, this assumption somewhat simplifies the estimation.

5.1.2 Log Likelihood

The Log Likelihood of the data is given by:

$$L_i(d_i, x_i|\theta_1, \theta_2) = \ln \left[\sum_{k=1}^{K} \pi_k \left[\prod_{t=1}^{T} \left[\prod_{j=1}^{J} Pr(d_{it} = j|x_{it}, s_i = k; \theta_1, \theta_2)^{I(d_{it}=j)} \right] \cdot g(x_{it}|d_{it-1}, x_{it-1}, s_i = k; \theta_2) \right] \right]$$

where $I(d_{it} = j)$ is an indicator that is 1 if $d_{it} = j$ is the observed choice in data, and zero otherwise. A key point here is that we cannot separate the Log Likelihoods of the state transitions and choice probabilities as we did the simpler case without s_i. This is because weights π_ks are inside the ln as opposed to outside. This precludes the estimation of $g(\cdot)$ in an initial first step. Intuitively, $g(\cdots)$ are not directly observed in the data anymore since they are functions of the unobserved state s_i. We have to jointly estimate the population probabilities, the state transitions $g()$, and the choice probabilities $Pr(\cdot)$.

Recall that, with Type 1 extreme values, we can write the choice probabilities as

$$Pr(d_{it} = j|x_{it}, s_i = k; \theta_1, \theta_2) = \frac{\exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1) + \beta \int \bar{V}(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2) \mathrm{d}x_{it+1}\right)}{\sum_{j=1}^{J} \exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1) + \beta \int \bar{V}(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2) \mathrm{d}x_{it+1}\right)}$$

Thus, we have the following estimation challenges:

- Both $\bar{u}(x_{it}, s_i = k, d_{it} = j, \theta_1)$ and $\bar{V}(x_{it+1}, s_i = k)$ are functions of the unobserved state s_i.
- Population probabilities π_ks are not known.
- Sequential estimation of state transitions and choice probabilities is not possible.

The EM algorithm can help us address these challenges as we will show next.

5.2 EM Algorithm

The EM algorithm was first proposed by Dempster et al. (1977) and then expanded to a large number of settings where researchers wanted to allow for finite mixture of latent classes. The key idea behind EM is that instead of solving a difficult optimization problem where we directly optimize for π_ks, we can solve a series of easier optimization problems that instead take posterior probabilities as the input and eventually converge to the same result.

5.2.1 Equivalence in Log Likelihoods

In our setting, EM provides the following equivalence:

$$\begin{aligned}
\{\hat{\theta_1}, \hat{\theta_2}\} &= \underset{\theta_1,\theta_2,\pi}{\arg\max} \sum_{i=1}^{N} \ln \left[\sum_{k=1}^{K} \pi_k \left[\prod_{t=1}^{T} \left[\prod_{j=1}^{J} Pr(d_{it}|x_{it}, s_i = k; \theta_1, \theta_2)^{I(d_{it}=j)} \right] \cdot g(x_{it}|d_{it-1}, x_{it-1}, s_i; \theta_2) \right] \right] \\
&\equiv \underset{\theta_1}{\arg\max} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} \sum_{j=1}^{J} \rho(k|d_i, x_i; \hat{\theta}_1, \hat{\theta}_2, \hat{\pi}) \cdot \ln \left[Pr(d_{it} = j|x_{it}, s_i = k; \theta_1, \hat{\theta}_2) \right]^{I(d_{it}=j)} \\
&+ \underset{\theta_2}{\arg\max} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} \rho(k|d_i, x_i; \hat{\theta}_1, \hat{\theta}_2, \hat{\pi}) \cdot \ln \left[g(x_{it}|x_{it-1}, s_i = k, d_{it-1}; \theta_2) \right] \qquad (5.5)
\end{aligned}$$

where $\rho(k|d_i, x_i; \hat{\theta}_1, \hat{\theta}_2, \hat{\pi})$ is the posterior probability that agent i belongs to type k conditional on observed data and the parameter estimates $\{\hat{\theta}_1, \hat{\theta}_2, \hat{\pi}\}$.

How does EM help us?

- First, it re-introduces additive separability in the Log Likelihoods for the choice probabilities and the state transition parameters. This allows us to sequentially estimate them again.
- Theoretically, EM assures us that the Log Likelihoods are non-decreasing at each iteration of the algorithm. This guarantees convergence.

EM does have a few downsides. First, it does not guarantee that the algorithm will converge to the global maximum. This can be an issue because the Log Likelihoods of latent class models tend to be non-concave with multiple local maxima. So the researcher has to experiment with many different starting values and explore the Likelihood curve manually. Second, it can be quite slow. EMs especially tend to crawl towards the end.

5.2.2 Step by Step EM Algorithm

We now describe the EM algorithm with Nested Fixed Point in detail. We first present the general case here, where the state transition are also assumed to be functions of the unoobserved state s_i. In §5.2.3, we present the step by step procedure for the simpler case when $g(\cdot)$ is independent of s_i.

Step 1: Initialization Step

For K types, make an initial guess of the population probabilities as $\{\pi_1^1, \pi_2^1, \ldots, \pi_K^1\}$ such that $\sum_{k=1}^{K} \pi_k^1 = 1$. Also, make an initial guess of the utility and state transition parameters $\{\theta_1^1, \theta_2^1\}$. The superscript 1 refers to iteration number.

Step 2: Update the Value Functions

Using Nested Fixed Point, calculate the value function for each type k as follows:

$$\begin{aligned} & \bar{V}^2(x_t, s_i = k) \\ = \ & \ln\left[\sum_{j=1}^{J}\left[\exp\left(\bar{u}(x_t, s_i = k, d_t = j; \theta_1^1) + \beta \int \bar{V}^2(x_{t+1}, s_i = k) g(x_{t+1}|x_t, d_t = j, s_i = k, ; \theta_2^1) \mathrm{d}x_{t+1}\right)\right]\right] \end{aligned} \tag{5.6}$$

Here s_i and the parameters θ_1^1 and θ_2^1 are treated as known. So the value function iteration is the same as in Rust's simpler model. The only difference is that the number of dimensionality of the expected value function is:

$$\text{Dim}(\bar{V}) = X \times K \tag{5.7}$$

Step 3: Update Posteriors

Update the posterior probabilities that agent i belongs to type k (for all k) as follows:

$$\begin{aligned} & \rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1) \\ = \ & \frac{\pi_k^1 \left[\prod_{t=1}^{T}\left[\left[\prod_{j=1}^{J}\left[Pr(d_{it} = j|x_{it}, s_i = k; \theta_1^1, \theta_2^1)\right]^{I(d_{it}=j)}\right] \cdot g(x_{it}|x_{it-1}, s_i = k, d_{it-1}; \theta_2^1)\right]\right]}{\sum_{k=1}^{K}\left[\pi_k^1 \left[\prod_{t=1}^{T}\left[\left[\prod_{j=1}^{J}\left[Pr(d_{it} = j|x_{it}, s_i = k; \theta_1^1, \theta_2^1)\right]^{I(d_{it}=j)}\right] \cdot g(x_{it}|x_{it-1}, s_i = k, d_{it-1}; \theta_2^1)\right]\right]\right]} \end{aligned} \tag{5.8}$$

Note that the ρs for each person i across all ks should add up to 1.

Step 4: Expectation (or E) Step

This is the E step. Update the population probabilities for type k as follows:

$$\pi_k^2 = \frac{\sum_{i=1}^{N} \rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1)}{N} \ \ \forall \ \ k \tag{5.9}$$

where N is the total number of agents observed.

Step 5: Maximization Step

First we maximize the state transition probabilities as follows:

$$\arg\max_{\theta_2^2} \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{T} \rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1) \cdot \ln\left[g(x_{it}|x_{it-1}, s_i = k, d_{it-1}; \theta_2^2)\right] \tag{5.10}$$

where $\rho_i^2(\cdot)$s are the posterior probabilities calculated in Step 3 for this iteration (2 in this case). This gives the estimate of θ_2 for this iteration, θ_2^2.

Now write down the choice probabilities for each type as follows:

$$Pr(d_{it} = j|x_{it}, s_i = k; \theta_1^2, \theta_2^2) = \frac{\exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1^2) + \beta \int \bar{V}^2(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^2) \mathrm{d}x_{it+1}\right)}{\sum_{j=1}^{J} \exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1^2) + \beta \int \bar{V}^2(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^2) \mathrm{d}x_{it+1}\right)} \tag{5.11}$$

Once we have a numerical estimate for the continuation value $\beta \int \bar{V}^2(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^2) \mathrm{d}x_{it+1}$, the choice probability is relatively straightforward. We can use the estimates of $\bar{V}^2(x_{it+1}, s_i)$ and $g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^2)$ from the this iteration (in this case iteration 2) to calculate the continuation value and plug it into the equation above.

Then do the following maximization:

$$\arg\max_{\theta_1^2} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} \sum_{j=1}^{J} \rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1) \cdot \ln\left[Pr(d_{it} = j|x_{it}, s_i = k; \theta_1^2, \theta_2^2)\right]^{I(d_{it}=j)} \tag{5.12}$$

where $\rho_i^2(\cdot)$s are the posterior probabilities calculated in Step 3 for this iteration (2 in this case). This gives the estimate of θ_1 for this iteration, θ_1^2.

Note that in both the above maximizations (for state transition and choice) are simple because the $\rho_i^2(\cdot)$s are outside of the ln.

EM Loop

Repeat steps 2-5 till the parameter values converge. A few hints:

- Do not declare convergence based on Likelihood or posteriors.
- Algorithm will tend to crawl towards the end, but Log Likelihood should increase with every step. If this doesnt happen, you have a bug!
- Start with parameter estimates from the dynamic model without persistent unobservables. It is a good starting value, so usually gives us faster convergence. But try other starting values too because EM does not guarantee global maximum.

5.2.3 Estimation when State Transitions are Independent of s_i

When state transitions are not a function of s_i, we can write the Log Likelihood as follows:

$$\begin{aligned} & L_i(d_i, x_i|\theta_1, \theta_2) = L_i(d_i|x_i, \theta_1, \theta_2) + L_i(x_i|\theta_2) \\ = \; & \ln\left[\sum_{k=1}^{K} \pi_k \left[\prod_{t=1}^{T} \left[\prod_{j=1}^{J} Pr(d_{it} = j|x_{it}, s_i = k; \theta_1, \theta_2)^{I(d_{it}=j)}\right]\right]\right] + \sum_{t=1}^{T} \ln\left[g(x_{it}|d_{it-1}, x_{it-1}; \theta_2)\right] \end{aligned} \tag{5.13}$$

Step 0: State Transition Estimation

Since $g(\cdot)$ is not a function of s_i, we can estimate $\hat{\theta}_2$ in an initial step outside the EM loop.

Step 1: Initialization

As in the general case, initialize the population probabilities and utility parameters as follows: $\{\pi_1^1, \pi_2^1, \ldots, \pi_K^1\}$ such that $\sum_{k=1}^K \pi_k^1 = 1$ and θ_1^1.

Step 2: Update the Value Functions

Using Nested Fixed Point, estimate the value functions.

$$\begin{aligned} & \bar{V}^2(x_t, s_i = k) \\ = \quad & \ln\left[\sum_{j=1}^{J}\left[\exp\left(\bar{u}(x_t, s_i = k, d_t = j; \theta_1^1) + \beta \int \bar{V}^2(x_{t+1}, s_i = k) g(x_{t+1}|x_t, d_t = j; \hat{\theta}_2) \mathrm{d}x_{t+1}\right)\right]\right] \end{aligned} \tag{5.14}$$

Step 3: Update Posterior Probabilities

Here we update the posterior probabilities. Since state transitions are independent of s_i, we use the simpler formula:

$$\rho_i^2(k|d_i, x_i; \theta_1^1, \hat{\theta}_2, \pi^1) = \frac{\pi_k^1\left[\prod_{t=1}^{T}\left[\prod_{j=1}^{J}\left[Pr(d_{it}|x_{it}, s_i = k; \theta_1^1, \hat{\theta}_2)\right]^{I(d_{it}=j)}\right]\right]}{\sum_{k=1}^{K}\left[\pi_k^1\left[\prod_{t=1}^{T}\left[\prod_{j=1}^{J}\left[Pr(d_{it}|x_{it}, s_i = k; \theta_1^1, \hat{\theta}_2)\right]^{I(d_{it}=j)}\right]\right]\right]}$$

Here $\hat{\theta}_2$ is the estimated value of θ_2 from a first step procedure outside the EM loop.

Step 4: Expectation (or E) Step

Update the population probabilities as follows:

$$\pi_k^2 = \frac{\sum_{i=1}^{N} \rho_i^2(k|d_i, x_i; \theta_1^1, \hat{\theta}_2, \pi^1)}{N} \quad \forall \;\; k \tag{5.15}$$

Step 5: Maximization Step

We can write down the choice probabilities for each type as follows.

$$\begin{aligned} & Pr(d_{it} = j|x_{it}, s_i = k; \theta_1^2, \hat{\theta}_2) = \\ & \frac{\exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1^2) + \beta \int \bar{V}^2(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, d_{it} = j; \hat{\theta}_2) \mathrm{d}x_{it+1}\right)}{\sum_{j=1}^{J} \exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1^2) + \beta \int \bar{V}^2(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, d_{it} = j; \hat{\theta}_2) \mathrm{d}x_{it+1}\right)} \end{aligned} \tag{5.16}$$

Note that the state transition parameters, $\hat{\theta}_2$ are treated as known here. As before, we can use the estimates of $\bar{V}^2(x_{it+1}, s_i)$ from the this iteration (in this case iteration 2) and $g(x_{it+1}|x_{it}, d_{it} = j; \hat{\theta}_2^2)$ to calculate the continuation value and plug it into the equation above.

Then do the following maximization:

$$\arg\max_{\theta_1^2} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} \sum_{j=1}^{J} \rho_i^2(k|d_i, x_i; \theta_1^1, \hat{\theta}_2, \pi^1) \cdot \ln \left[Pr(d_{it} = j|x_{it}, s_i = k; \theta_1^2, \hat{\theta}_2) \right]^{I(d_{it}=j)} \quad (5.17)$$

where $\rho_i^2(\cdot)$s are the posterior probabilities calculated in Step 3 for this iteration (2 in this case). This gives the estimate of θ_1 for this iteration θ_1^2.

EM Loop

Repeat steps 2-5 till the parameters $\{\theta_1, \pi_1\}$ converge.

Chapter 6

Persistent Unobservables and Two-Step Methods

Arcidiacono and Miller (2011) expanded the two-step method to estimate dynamic models with persistent unosbervables. We discuss their method below.

6.0.1 Step by Step EM Algorithm with Two-Step Method

We now describe the EM algorithm with Two-Step method in detail. We consider the general case, where the state transition are assumed to be functions of the unoobserved state s_i.

Step 1: Initialization Step

Make an initial guess of the following:

- The population probabilities for the K types, $\{\pi_1^1, \pi_2^1, \ldots, \pi_K^1\}$ such that $\sum_{k=1}^K \pi_k^1 = 1$.
- The utility and state transition parameters $\{\theta_1^1, \theta_2^1\}$.
- The CCPs for all alternative j for each combination of $\{x_{it}, s_i\}$, which we denote as $\hat{p}^1(j|x_{it}, s_i = k)$. In the first iteration, we can simply assume that $\hat{p}^1(j|x_{it}, s_i = k) = \hat{p}^1(j|x_{it}) \ \ \forall \ \ k$. We use $\hat{p}^1$ to denote the set of all CCPs in iteration 1.

In all of the above, the superscript 1 refers to iteration number.

Step 2: Update the Value Functions

We first update the choice specific value functions for the first iteration as $v^1\left(x, s = k, d = j; \theta_1^1, \theta_2^1, \hat{p}^1\right)$ at each combination of state and decision using the two-step method and simulations as follows:

$$
\begin{aligned}
& v^1\left(x, s = k, d = j; \theta_1^1, \theta_2^1, \hat{p}^1\right) \qquad (6.1)\\
= \ & \frac{1}{H}\sum_{h=1}^{H}\Big[\bar{u}(x, s = k, d = j; \theta_1^1) + \beta\left[u(x_h^{'}, s = k, d_h^{'} = j; \theta_1^1) + \gamma - \ln(\hat{p}^1(j|x_h^{'}, s = k)\right)\\
+ \ & \beta\left[u(x_h^{''}, s = k, d_h^{''} = j; \theta_1^1) + \gamma - \ln(\hat{p}^1(j|x_h^{''}, s = k)) + \beta\ldots\right]\Big]\Big]
\end{aligned}
$$

where H is the total number of simulations, h is the simulation number. In each simulation h, we draw from the two distributions, $g^1(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^1)$ and $\hat{p}^1(j|x_{it}, s_i = k)$ in an

alternating fashion.

That is, we first draw $x_h'|(x, s = k, d = j)$ using $g^1(\cdot)$ and $d_h'|x_h', s = k$ using $\hat{p}^1(\cdot)$. For the next period, we then draw $x_h^{''}|(x_h', s = k, d_h')$ and $d_h^{''}|x_h^{''}, s = k$ using $g^1(\cdot)$ and $\hat{p}^1(\cdot)$ respectively, and so on, till some large but finite number of periods, say T_{sim}. We make H such simulations and take the average to get a numerical estimate of $v^1\left(x, s = k, d = j; \theta_1^1, \theta_2^1, \hat{p}^1\right)$

Of course, if we have finite dependence, then we can simplify the above simulation to just one-step ahead choice probabilities as discussed earlier.

Using the choice specific value functions, we can then calculate the expected value function for a given state $\{x_{it}, s_i\}$as follows:

$$\bar{V}^1(x_{it}, s_i = k) = \ln\left[\sum_{j=1}^{J}\left[\exp\left(v^1(x_{it}, s_i = k, d_{it} = j; \theta_1^1, \theta_2, \hat{p}^1)\right)\right]\right] \tag{6.2}$$

Step 3: Predict Choice Probabilities

Now write down the choice probabilities for each type for each state in this iteration as follows:

$$Pr(d_{it} = j|x_{it}, s_i = k; \theta_1^1, \theta_2^1, \bar{V}^1) = \tag{6.3}$$
$$\frac{\exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1^2) + \beta\int \bar{V}^1(x_{it+1}, s_i = k)g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^1)\mathrm{d}x_{it+1}\right)}{\sum_{j=1}^{J}\exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1^2) + \beta\int \bar{V}^1(x_{it+1}, s_i = k)g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^1)\mathrm{d}x_{it+1}\right)}$$

To derive the numerical estimate for the continuation value
$\beta\int \bar{V}^1(x_{it+1}, s_i = k)g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^1)\mathrm{d}x_{it+1}$, we can use the estimates of $\bar{V}^1(x_{it+1}, s_i)$ from Step 2 and $g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^1)$ from the this iteration (in this case iteration 1). We then plug the numerical estimate of continuation into the equation above, along with the current estimates of θ_1^1 to derive the choice probabilities for a given combination of $\{x_{it}, s_i = k\}$.

Step 4: Update Posterior Probabilities

Next, we use the predicted choice probabilities from Step 3 and the the state transition probabilities from this iteration to update the posteior probabilities.

The posterior probability that agent i belongs to type k (for all k) is derived as follows:

$$\rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1) \tag{6.4}$$
$$= \frac{\pi_k^1\left[\prod_{t=1}^{T}\left[\left[\prod_{j=1}^{J}\left[Pr(d_{it}|x_{it}, s_i = k; \theta_1^1, \theta_2^1, \bar{V}^1)\right]^{I(d_{it}=j)}\right]\cdot g(x_{it}|x_{it-1}, s_i = k, d_{it-1}; \theta_2^1)\right]\right]}{\sum_{k=1}^{K}\left[\pi_k^1\left[\prod_{t=1}^{T}\left[\left[\left[Pr(d_{it}|x_{it}, s_i = k; \theta_1^1, \theta_2^1, \bar{V}^1)\right]^{I(d_{it}=j)}\right]\cdot g(x_{it}|x_{it-1}, s_i = k, d_{it-1}; \theta_2^1)\right]\right]\right]}$$

Note that the ρs should add up to 1.

Step 5: Expectation (or E) Step

This is the E step. Update the population probabilities for type k as follows:

$$\pi_k^2 = \frac{\sum_{i=1}^{N} \rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1)}{N} \quad \forall \ k \tag{6.5}$$

where N is the total number of agents observed.

Step 6A: Maximization Step: State Transitions

First we maximize the state transition probabilities as follows:

$$\arg\max_{\theta_2^2} \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1) \cdot \ln\left[g(x_{it}|x_{it-1}, s_i = k, d_{it-1}; \theta_2^2)\right] \tag{6.6}$$

where $\rho_i^2(\cdot)$s are the posterior probabilities calculated in Step 4. This is the second updating for θ_2, so we denote it as θ_2^2.

Step 6B: Maximization Step: Utility Parameters

Now write down the choice probabilities for each type as follows:

$$Pr(d_{it} = j|x_{it}, s_i = k; \theta_1^2, \theta_2^2, \bar{V}^1) = \tag{6.7}$$
$$\frac{\exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1^2) + \beta \int \bar{V}^1(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^2) \mathrm{d}x_{it+1}\right)}{\sum_{j=1}^{J} \exp\left(\bar{u}(x_{it}, s_i = k, d_{it} = j; \theta_1^2) + \beta \int \bar{V}^1(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^2) \mathrm{d}x_{it+1}\right)}$$

Once we have a numerical estimate for the continuation value $\beta \int \bar{V}^1(x_{it+1}, s_i = k) g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^2) \mathrm{d}x_{it+1}$, the choice probability is relatively straightforward. We can use the estimates of $\bar{V}^1(x_{it+1}, s_i)$ from Step 2 and $g(x_{it+1}|x_{it}, s_i = k, d_{it} = j; \theta_2^2)$from Step 6A to calculate the continuation value and plug it into the equation above.

Then do the following maximization:

$$\arg\max_{\theta_1^2} \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1) \cdot \ln\left[Pr(d_{it} = j|x_{it}, s_i = k; \theta_1^2, \theta_2^2, \bar{V}^1)^{I(d_{it})}\right] \tag{6.8}$$

where $\rho_i^2(\cdot)$s are the posterior probabilities calculated in Step 3. This gives the updated estimate of θ_1, which we refer to as θ_1^2.

Note that in both the above maximizations (for state transition and choice) are simple because the $\rho_i^2(\cdot)$s are outside of the ln.

Step 7: Update CCPs

There are two possible ways to update the CCPs. First, we could use the data and the posteriors to update the CCP as follows:

$$\hat{p}^2(j|x, s = k) = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T}\rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1)I(x_{it} = x, d_{it} = j)}{\sum_{i=1}^{N}\sum_{t=1}^{T}\rho_i^2(k|d_i, x_i; \theta_1^1, \theta_2^1, \pi^1)I(x_{it} = x)} \tag{6.9}$$

Note that if the state space is very large, then we may have to use a weighted flexible logit or some semi-parametric functional form to estimate CCPs.

The second way to update the CCPs is using the model restrictions and the current estimates of parameters as follows:

$$\hat{p}^2(j|x, s = k) = \tag{6.10}$$
$$\frac{\exp\left(\bar{u}(x_{it} = x, s_i = k, d_{it} = j; \theta_1^2) + \beta\int \bar{V}^1(x_{it+1}, s_i = k)g(x_{it+1}|x_{it} = x, s_i = k, d_{it} = j; \theta_2^2)\mathrm{d}x_{it+1}\right)}{\sum_{j=1}^{J}\exp\left(\bar{u}(x_{it} = x, s_i = k, d_{it} = j; \theta_1^2) + \beta\int \bar{V}^1(x_{it+1}, s_i = k)g(x_{it+1}|x_{it} = x, s_i = k, d_{it} = j; \theta_2^2)\mathrm{d}x_{it+1}\right)}$$

EM Loop

Repeat steps 2-7 till the parameter values converge. A few hints:

- Do not declare convergence based on Likelihood or posteriors.
- Algorithm will tend to crawl towards the end, but Log Likelihood should increase with every step. If this doesnt happen, you have a bug!
- Start with parameter estimates from the dynamic model without persistent unobservables. It is a good starting value, so usually gives us faster convergence. But try other starting values too because EM does not guarantee global maximum.

Chapter 7

Identification

We now discuss the identification of dynamic discrete choice models. The ideas presented below are drawn from Rust (1987), Hotz and Miller (1993), Magnac and Thesmar (2002), Bajari, Chernozhukov, Hong, and Nekipelov (2007), Hall and Zhou (2003), Kasahara and Shimotsu (2009), and Arcidiacono and Miller (2011).

Identification refers to whether a structural parameter can be uniquely recovered from a dataset tending to infinity. In this chapter, we concern ourselves with non-parametric identification of parameters of interest.

To clarify ideas, we first start with the identification of static discrete choice models.

7.1 Identification of Static Discrete Choice Models

As usual, we consider the decision of agent i who faces J discrete options at time t. The agent's decisions are influenced by one observable state variable x_{it} whose dimensionality is X and one unobservable state variable ϵ_{it}, which is a $J \times 1$ vector with support $\mathbb{R}^J$ alternatives.

We are interested in identifying the deterministic portion of the agent's flow utility $\bar{u}(x_{it}, j)$ for all combinations of x and j. To non-parametrically identitfy these values, we need to make two assumptions.

Assumption 6. *ϵ_{it} is assumed to be drawn from a known distribution $F(\epsilon)$.*

Assumption 7. *The deterministic portion of the flow utility is normalized to zero for all xs for one of the options, say $j = 1$. That is: $\bar{u}(x, 1) = 0$.*

With Assumption 7, we need to identify $X \times (J-1)$ flow utility values $\bar{u}(x, j)$. In the data, we directly observe choice probabilities $Pr(j|x)$. Since there are J options and X observables, this translates to $J \times X$ observed choice probabilities. However, we know that choice probabilities add up to 1, so the number of unique observables is $X \times (J-1)$.

Next, by invoking Assumption 6, we can see that there is a direct one to one relationship between choice probabilities and flow utilities, which allows us uniquely identify $\bar{u}(x, j)$

This can be illustrated for the case of Type 1 extreme value errors easily as follows. For each x, we have the following $J-1$ equations:

$$\begin{aligned}
Pr(j=2|x) &= \frac{\exp(\bar{u}(x,j=2))}{1+\exp(\bar{u}(x,j=2))+\ldots+\exp(\bar{u}(x,j=J))} \\
Pr(j=3|x) &= \ldots \\
&\ldots\ldots\ldots \\
Pr(j=J|x) &= \frac{\exp(\bar{u}(x,j=J))}{1+\exp(\bar{u}(x,j=2))+\ldots+\exp(\bar{u}(x,j=J))}
\end{aligned}$$

Thus, for each x, we have the $J-1$ equations listed above, where $Pr(j|x)$ are known and we have $J-1$ unknowns which are: $\{\bar{u}(x,j=2),\bar{u}(x,j=3),\ldots,\bar{u}(x,j=J)\}$. Since this system of equations is solvable without any additional restrictions, $\bar{u}(x,j)$s are non-parametrically identified.

We can also specify these equalities over all xs, in which case we would have $X\times(J-1)$ equations and $X\times(J-1)$ unknowns. Thus, for all Xs, the flow utilities $\bar{u}$s are uniquely identified without imposing any parametric restrictions on the functional form of $\bar{u}$.

7.2 Identification of Stationary Dynamic Discrete Choice Models

We now expand the above arguments to include dynamics. In this section, we focus on stationary dynamic discrete choice models. Please see§7.4, for identification results for non-stationary or finite horizon dynamic discrete choice models.

In the case of stationary dynamic discrete choice model that follows a first-order markov process, we observe the following probabilities directly from data.

$$Pr(j,x|x',j') = Pr(d|x)\times g(x|x',j') \tag{7.1}$$

The first, $Pr(j|x)$, is the conditional choice probability or CCP of observing the choice of option j given state x. The second probability, $g(x|x',j')$, is the first-order markovian state transition probability. Both of these are non-parametrically available from the data directly. These are our observables. For J options and X states, we have $X\times(J-1)$ CCPs and $X\times J\times(X-1)$ state transition probabilities.

We also make the following assumption.

Assumption 8. *The discount factor β is assumed to known.*

In a dynamic setting, we have two types of unknowns:

- $X\times(J-1)$ flow utilities $\bar{u}(x,j)$.
- X expected value functions $\bar{V}(x)$.

Thus, we have a total of $X\times(J-1)+X$ unknowns to identify.

Now, consider the restrictions. We have two types of restrictions – 1) Bellman equations and 2) Choice probabilities. The first can be written as:

$$\bar{V}(x) = \max_j \left[\bar{u}(x,j) + \epsilon_{ijt} + \beta \int \bar{V}(x') g(x'|x,j) \mathrm{d}x' \right] \tag{7.2}$$

When the distribution of ϵ is known, we have X such Bellman equations. In addition, we have $X \times (J-1)$ equations from the choice probabilities which are functions of $\bar{u}(x,j)$, $\bar{V}(x)$, and $g(x|x',j')$.

As a specific case, with the assumption of Type 1 extreme value error, we have the following set of equations:

$$\bar{V}(x) = \ln \left[\sum_{j=1}^{J} \exp \left[\bar{u}(x,j) + \beta \int \bar{V}(x') g(x'|x,j) \mathrm{d}x' \right] \right] \tag{7.3}$$

which consists of X Bellman equations.

In addition, we have the following $X \times (J-1)$ choice probability equations:

$$\begin{aligned}
Pr(j=2|x) &= \frac{\exp\left[\bar{u}(x,j=2) + \beta \int \bar{V}(x') g(x'|x,j=2) \mathrm{d}x'\right]}{D(x)} \\
Pr(j=3|x) &= \ldots \\
&\ldots\ldots\ldots \\
Pr(j=J|x) &= \frac{\exp\left[\bar{u}(x,j=J) + \beta \int \bar{V}(x') g(x'|x,j=J) \mathrm{d}x'\right]}{D(x)}
\end{aligned}$$

where $D(x)$ is defined as the denominator in the choice probabilities defined above. It is the exponentiation of the expected value function at x, and is defined as follows:

$$\begin{aligned}
D(x) &= \sum_{j=1}^{J} \exp \left[\bar{u}(x,j) + \beta \int \bar{V}(x') g(x'|x,j) \mathrm{d}x' \right] \\
&= \exp\left[V(x)\right]
\end{aligned} \tag{7.4}$$

We thus have a total of $X \times (J-1) + X$ equations and since we have the same number of unknowns, we can solve for the unknowns. Thus, both $\bar{u}(x,j)$ and $\bar{V}(x)$ are non-parametrically identified when we assume a functional form for ϵ and β. A key point of note is that the discount factor β is not non-parametrically identified above, and that we have assumed it.

7.3 Identification of the Discount Factor in Stationary Settings

In general, most papers using dynamic discrete choice models assume the discount factor β. However, in some cases, we can non-parametrically identify β from the data. We now examine the conditions under which this is possible in stationary dynamic discrete choice models.

This requires the following exclusion restriction: There exists a state variable variable z that affects state transitions, but not flow utilities. That is, we make the following assumption.

Assumption 9. *Exclusion Restriction for Identifying β: There exists a state variable z such that $\bar{u}(x,z,j) = \bar{u}(x,j)\ \forall\ x,j$ and $g(x,z|x',z',j') \neq g(x,z|x',j')$ for at least some z.*

We observe the following from the data directly (i.e., known quantities):

- $X \times Z \times (J-1)$ choice probabilities
- $(X \times Z \times J) \times (X \times Z - 1)$ state transition probabilities.

Next, we have the following unknowns to idenfity:

- $X \times (J-1)$ flow utilities $\bar{u}(x,j)$.
- $X \times Z$ expected value functions $\bar{V}(x,z)$.
- The discount factor β.

This is a total of $X \times (J-1) + X \times Z + 1$ unknowns that we need to non-parametrically identify.

In terms of equations, we have the following $X \times Z$ Bellman equations (for convenience we have assumed a Type 1 extreme value distribution for ϵ, though other distributions would work as well):

$$\bar{V}(x,z) = \ln\left[\sum_{j=1}^{J} \exp\left[\bar{u}(x,j) + \beta \int\int \bar{V}(x',z')g(x',z'|x,z,j)\mathrm{d}x'\mathrm{d}z'\right]\right] \tag{7.5}$$

In addition, we have the following $X \times Z \times (J-1)$ choice probability equations:

$$\begin{aligned}
Pr(j=2|x,z) &= \frac{\exp\left[\bar{u}(x,j=2) + \beta\int\int \bar{V}(x',z')g(x',z'|x,z,j=2)\mathrm{d}x'\right]}{D(x,z)} \\
Pr(j=3|x,z) &= \ldots \\
&\cdots\cdots\cdots \\
Pr(j=J|x,z) &= \frac{\exp\left[\bar{u}(x,j=J) + \beta\int\int \bar{V}(x',z')g(x',z'|x,j=J)\mathrm{d}x'\mathrm{d}z'\right]}{D(x,z)}
\end{aligned}$$

where $D(x,z)$ is the denominator in the choice probabilities above. It is the exponentiation of the expected value function at $\{x,z\}$, and is defined as follows:

$$\begin{aligned}
D(x,z) &= \sum_{j=1}^{J} \exp\left[\bar{u}(x,j) + \beta \int\int \bar{V}(x',z')g(x',z'|x,z,j)\mathrm{d}x'\mathrm{d}z'\right] \\
&= \exp\left[V(x,z)\right]
\end{aligned} \tag{7.6}$$

Thus, we have a total of $X \times Z + X \times Z \times (J-1)$ equations and $X \times (J-1) + X \times Z + 1$ unknowns. This system of equations is solvable and we can identify the unknowns if:

$$\begin{aligned} X \times Z + X \times Z \times (J-1) &\geq X \times (J-1) + X \times Z + 1 \\ X \times Z \times (J-1) &\geq X \times (J-1) + 1 \\ X \times (Z-1) \times (J-1) &\geq 1 \end{aligned} \tag{7.7}$$

The above inequality is satisfied as long as $Z > 1$ since $J \geq 2$ and $X > 1$. Thus, if we have a excluded state z with a dimensionality of at least 2, we can non-parametrically identify the discount factor β. Note that this is true by assumption (the state transition is a function of z; this would not be possible if $Z = 1$.)

7.4 Identification of Non-stationary Dynamic Discrete Choice Models

We now discuss how identification works in the case of non-stationary dynamic discrete choice models. The identification proof for these models can be a generalization of the proof for a stationary infinite-horizon case.

The main difference here is that time t is a state variable now. Assume that we have data for $t = 1, \ldots T$ periods. Thus, the dimensionality of our state space is now XT. Therefore, we expand Assumption 7 as follows for non-stationary settings:

Assumption 10. *The deterministic portion of the flow utility is normalized to zero for all $\{x, t\}$ for one of the options, say $j = 1$. That is: $\bar{u}(x, t, 1) = 0 \; \forall \; x, t$.*

Given infinite data, we can directly observe $XT(J-1)$ conditional choice probabilities (CCP) and $XJ(X-1)(T-1)$ state transitions from the data. For state transitions, we have $(T-1)$ matrices of XJ by X, each of which gives us $XJ(X-1)$ state transition probabilities (since the probabilities always add up to one). It is worth noting that transitions for time are deterministic.

On the other hand, we have the following unknowns to identify:

- $XT(J-1)$ utility values, $\bar{u}(x, t, j)$
- $X(T-1)$ value functions, $\bar{V}(x, t)$

We have $X(T-1)$ value functions instead of XT since the value function in the last period is simply the flow utility.

Assuming that error terms are Type I extreme value, we can write $X(T-1)$ Bellman equations as follows:

$$\bar{V}(x, t) = \ln \left[\sum_{j=1}^{J} \exp \left[\bar{u}(x, t, j) + \beta \int \bar{V}(x', t+1) g(x' \mid x, t, j) \mathrm{d}x' \right] \right] \tag{7.8}$$

Next, we can write $XT(J-1)$ choice probability equations for $j = 2, 3, \ldots, J$ as follows:

$$\Pr(j = 2 \mid x, t) = \frac{\exp\left(\bar{u}(x, t, j = 2) + \beta \int \bar{V}(x', t+1) g(x' \mid x, t, j) dx'\right)}{\exp\left[\bar{V}(x, t)\right]}, \tag{7.9}$$

where the denominator is simply $D(x, t) = \exp\left[\bar{V}(x, t)\right]$.

Taken together, we have $XT(J-1)$ CCP equations and $X(T-1)$ Bellman equations as constraints. Further, we have $X(T-1)$ value functions and $XT(J-1)$ flow utilities. Since these numbers equally add up, our model is identified.

Note that we can also use backward induction and directly calculate value functions when we identified the utility values. In other words, we can first use $XT(J-1)$ CCP equations to identify $XT(J-1)$ utility values, $\bar{u}(x, t, j)$, and then calculate the value functions using the fact that the horizon is finite from those. That is, the identification results can simply rely on $XT(J-1)$ choice probabilities.

7.5 Identification of Dynamic Discrete Choice Models with Persistent Unobservables

We now consider the identification for an infinite horizon stationary dynamic discrete choice model, with the following assumptions:

- The persistent unobservable s is time-invariant.
- There are no exlcuded variables z.
- The discount factor, β, is assumed by the researcher.
- The state transitions are independent of s, i.e., $g(x|x', j', s) = g(x|x', j')$.
- State transitions are unconstrained, i.e., $g(x|x', j') \neq 0$ for all combinations of x', j', and x.

More complex cases where these assumptions are relaxed can be solved using similar methods.

The first issue here is that we no longer directly observe the CCPs since they are functions of the unobservable state s. So we need to first need to recover them. We do that using the path data as follows.

Let S be the number of latent classes. Suppose that we observe T periods of data and suppose that agent are randomly assigned to x in period 1. If $T = 1$, we observe the following probabilities in the data:

$$Pr(x_1, j_1) = \sum_{s=1}^{S} i(x_1, s) Pr(j_1|x_1, s) \tag{7.10}$$

where the subscript denotes the time period. In the above set of equations, we have the following unobservables:

- $X \times S \times (J-1)$ CCPs, $Pr(j|x, s)$.

- $S(X-1)$ stationary probabilities that type s is in state x_1, $i(x_1, s)$. These can also be interpreted as initial condition probabilities.

Note that the population probability of type s, π_s can be derived from the stationary probabilities, $i(x_1, s)$, as follows:

$$\pi_s = \sum_{x_1} i(x_1, s) \tag{7.11}$$

That is, in a stationary DDC model, when we sum the stationary probabilities for a given type over all states, then we should obtain the population probability of that type. Therefore, π_ss do not have to be separately identified from the data.

Thus, we have a total of $X \times S \times (J-1) + S(X-1)$ parameters to be identified. However, with $T = 1$, we only have $X \times J - 1$ observed probabilities or equations. Clearly, this is less than the number of unknowns that we need to identify. So with just one period of data, we cannot identify this model.

Now consider $T = 2$. Then, we have:

$$Pr(\{x_1, j_1\}, \{x_2, j_2\}) = \sum_{s=1}^{S} i(x_1, s) Pr(j_1|x_1, s) g(x_2|x_1, j_1) Pr(j_2, |x_2, s) \tag{7.12}$$

In this case, we can have observed a total of $(X \times J)^2 - 1$ possible paths. Depending on number of latent classes, we wish to identify, this model is identifiable. For example, if $J = 2$, then we need to satisfy the following constraint for identification:

$$\begin{aligned} 4X^2 - 1 &\geq 2XS - 1 \\ 2X &\geq S \end{aligned} \tag{7.13}$$

So if $X = 2$, then we can at most identify four latent classes.

In general, the number of paths is of the order of X^T, whereas the number of parameters is of the order of X. So as T increases, for a reasonably small and finite S, this model is identified. It is also useful to note that the identification is easier as the number of co-variates (X) increases.

Once the CCPs and population probabilities are identified, the rest of the parameters to be identified are:

- $X \times S \times (J-1)$ utility values, $\bar{u}(x, s, j)$.
- $X \times S$ value functions, $\bar{V}(x, s)$

The identification of these proceeds similar to that in §7.2 since at this stage the CCPs are treated as known.

Chapter 8

Mathematical Programming with Equilibrium Constraints

8.1 MPEC as a Solution Concept

The two solution concepts that we have considered so far, both Nested Fixed Point and Two-Step methods have problems. The former is computationally expensive and the later is asymptotically inefficient. MPEC tries to address both these problems (though it is not clear that it can handle truly extensive state spaces like two-step methods can.) Nevertheless, it offers a solution for somewhat large state spaces without sacrificing efficiency at reasonable computational times. It was proposed by Su and Judd (2012) for dynamic discrete choice models and shown to be applicable for a broader set of problems by Dube, Fox, and Su (2012).

MPEC is applicable to any structural estimation setting where we need to maximize a gain function (or minimize a loss function) such as Maximum Likelihood, GMM or Simulated Maximum Likelihood.. In MPEC, we solve the following constrained optimization problem:

$$\max_{\theta,\sigma} L(\theta, \sigma; x) \quad \text{subject to} \quad h(\theta, \sigma) = 0 \tag{8.1}$$

where x is the set of state variables that affect the decision, θ is the set of structural parameters that need to be estimated, σ is a vector of endogenous parameters, implicitly defined by the set of equilibrium constraints or equations $h(\theta, \sigma) = 0$, and $L(\theta, \sigma; x)$ is the loss/gain function. The goal is to solve for θ and σ by maximizing (or minimizing) the function $L(\theta, \sigma; x)$ subject to the equilibrium constraints $h(\theta, \sigma) = 0$.

8.2 Applying MPEC to Dynamic Discrete Choice Models

We now apply this method to dynamic discrete choice model with a discrete state space. In this case, the equilibrium constraints can be expressed as a function of the EVs as follows:

$$EV(x,j) = \sum_{x'} \log \left[\sum_{j'=1}^{J} \exp\left(\bar{u}(x',j';\theta_1) + \beta EV(x',j')\right) \right] g(x'|x,j;\theta_2) \tag{8.2}$$

where $EV(x,j)$ is the future choice-specific expected value from choosing option j in state x. Note that here we have discretized the state space and hence the first summation is essentially equivalent to an intergal over x'.

With X states and J decisions, we thus have a vector of $X \times J$ EV values. Together they form the set of $X \times J$ constraints as follows:

$$EV(x,j) - \sum_{x'} \log \left[\sum_{j'=1}^{J} \exp\left(\bar{u}(x',j';\theta_1) + \beta EV(x',j')\right) \right] g(x'|x,j;\theta_2) = 0 \tag{8.3}$$

Denote the above as a system of equations of the form $EV = T(EV;\theta_1)$.

Next, consider the optimization problem. As usual, the estimates of the state transition probabilities are derived in the first stage from the following maximization (or available non-parametrically):

$$\hat{\theta}_2 = \arg\max_{\theta_2} \sum_{n=1}^{N} \sum_{t=1}^{T} \ln g(x_{it}|d_{it-1}, x_{it-1};\theta_2) \tag{8.4}$$

These can be fed into the EV constraints in Equation (8.3).

Then, we can write the "augmented" log likelihood for the decisions as:

$$L(\theta_1, EV) = \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{j=1}^{J} \ln \left[Pr(d_{it} = j|x_{it};\theta_1, EV) \right]^{I(d_{it}=j)} \tag{8.5}$$

where the choice probabilities are a function of θ_1 and EVs as follows:

$$Pr(d_{it} = j|x_{it};\theta_1, EV) = \frac{\exp\left[\bar{u}(x_{it}, d_{it} = j;\theta_1) + \beta EV(x_{it}, d_{it} = j)\right]}{\sum_{j=1}^{J} \exp\left[\bar{u}(x_{it}, d_{it} = j;\theta_1) + \beta EV(x_{it}, d_{it} = j)\right]} \tag{8.6}$$

We then perform the following constrained optimization:

$$\max_{\theta_1, EV} L(\theta_1, EV) \quad \text{subject to} \quad EV = T(EV, \theta_1) \tag{8.7}$$

With the contraction mapping properties of the value function, there is a unique set of EVs for each

θ_1. However, in games settings with multiple equilibriua, there could be a multiple values of EVs satisifying the constraint for a given θ.

Chapter 9

Machine Learning: Methods and Applications

9.1 Overview

Machine learning techniques are focused on producing accurate out of sample predictions. Machine learning tools differ from traditional marketing methods on several dimensions. First, machine learning methods are focused on obtaining the best out-of-sample predictions, while causal econometric methods aim to derive the best unbiased estimators. As we will show in §9.2.2, the best unbiased estimator does not always provide the best out-of-sample prediction, and there are instances where a biased estimator would perform better for out-of-sample data. Second, ML tools are designed to work in situations when we do not have an a priori theory about the process through which outcomes are generated. This is in contrast to econometric methods that are designed for testing a specific causal theory. Third, unlike standard empirical methods used in marketing, ML techniques can accommodate an extremely large number of variables, and indeed uncover which variables should be retained vs. which dropped. An example where such factors play a big role is predicting which URL a user will click on. We do not have a good theory/model of how users' clicking behavior occurs. We can, of course, come up with a parametric specification for the user's utility of each URL, but such a model is unlikely to accurately capture all the factors that influence the user's decision to click on a certain link. In addition to the underlying process being extremely complex, it is also potentially affected by a very large number of factors, such as all the text and images on the web page, and the user's entire previous browsing history. Finally, scalability is a key consideration in ML methods, and techniques such as feature selection and efficient optimization methods help achieve this. Scalability is a now increasingly important for marketers since many of these algorithms need to be run in real time to response to consumer needs.

Unlike the related field of data mining, machine learning methods typically do assume a model or structure to learn, but use a very general class of models that can be very rich. Methods for analyzing and modeling data can be divided into two groups: supervised learning and unsupervised learning. Supervised learning requires input data that has both predictor (independent) variables

and a target (dependent) variable whose value is to be estimated. By various means, the process "learns" how to predict the value of the target variable based on the predictor variables. Decision trees, regression analysis and neural networks are examples of supervised learning. If the goal of an analysis is to predict the value of some variable, then supervised learning is the usual approach. Unsupervised learning does not identify a target (dependent) variable, but rather treats all of the variables equally. In this case, the goal is not to predict the value of a variable but rather to look for patterns, groupings or other ways to characterize the data that may lead to an understanding of the way the data interrelates. Cluster analysis, correlation, factor analysis (principle components analysis), EM-algorithms, and topic modeling (text mining) are examples of unsupervised learning. In this chapter, we focus on supervised learning.

9.2 Supervised Learning

The goal of a supervised learning model is to predict an outcome variable y_i given a vector of input/explanatory variables x_i. Formally, consider a set of N data points such that $\{(y_1, x_1), \ldots, (y_i, x_i), (y_N, x_N)\}$, where x_i is a vector of explanatory variables used to predict the outcome y_i. x_i are also referred to as features.

The statistical learning approach to predictive modeling is one of function estimation, i.e., our goal is to infer a function $f(x_i)$ that best predicts y_i. This is different from the probabilistic parametric approach commonly adopted in the econometric paradigm, where y_i is modeled as being drawn from a probability distribution $f(x_i; \beta)$, and econometrician infers β conditional on $f(\cdot)$. The statistical learning approach can be interpreted as non-parametric or one that allows for a very general class of models.

9.2.1 Loss Functions

In order to estimate $f(x_i)$, we first need to specify a "Loss function" or an objective function. This function is the summation:

$$L(y, f(x)) = \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{9.1}$$

Loss functions can take many forms. The most commonly used loss functions are:

- Squared loss: $L(y, f(x)) = \frac{1}{2}(y - f(x))^2$
- Absolute error: $L(y, f(x)) = |y - f(x)|$

For classification, we also often use:

- Exponential loss: $L(y, f(x)) = \exp(-\tilde{y} \cdot f(x))$
- Log loss: $L(y, f(x)) = \log(1 + \exp(-\tilde{y} f(x)))$
- 0-1 Loss: $L(y, f(x)) = 1$ if $\tilde{y} f(x) > 0$, else $L(y, f(x)) = 0$

where $\tilde{y} \in \{-1, 1\}$, which corresponds to $y \in \{0, 1\}$.

The following is an illustration of different types of loss functions. Make a figure.

The loss function is typically chosen with the problem objectives in mind. For example, the exponential loss function puts a lot of weight on mistakes and this penalty is exponential increasing with the size of the mistake, whereas Log loss penalizes mistakes less severely in the extremes. In contrast, a 0-1 loss does not penalize the severity of the misprediction, but only its direction.

9.2.2 Bias-Variance Trade-off

The bias-variance tradeoff demonstrates the key difference between prediction and causal inference problems. In causal inference problems, the goal is to obtain unbiased estimates of the model parameters. However, when the goal is the best out-of-sample prediction, it is not necessary for parameter values to be unbiased. Therefore, methods built for causal inference are not optimized for prediction because they restrict themselves to unbiased estimators.

When assessing how good a model will be at making predictions, we make a distinction between two different sources of error: bias and variance. Error due to bias is the *systematic* error that we can expect from estimating the model on a new dataset. That is, if we were to collect new data and estimate the model several times, how far off would these models' predictions be on average. The error due to variance is the extent to which predictions for a point differ across different realizations of the data. For example, a model that overfits to the training data will have high variance-error because it would produce very different estimates on different datasets. Overfitting occurs when a model is fit too closely to a finite sample dataset. Thus, when the model is applied to a different finite sample, it performs poorly.

Let us now examine how these two sources of error affect a model's predictive ability. Let $y = f(x) + \epsilon$, where ϵ is normally distributed with mean 0 and variance σ_ϵ. Our goal is to estimate a model, $\hat{f}(x)$, to minimize mean square error of the prediction. The expected squared prediction error at point x is $MSE(x) = E\left[(y - \hat{f}(x))^2\right]$. This can be decomposed as follows:

$$MSE(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\hat{f}(x) - E[\hat{f}(x)]\right]^2 + \sigma_\epsilon^2 \tag{9.2}$$

The last term, σ_ϵ^2 is inherent noise in the data so it cannot be minimized and is not affected by our choice of $\hat{f}(x)$. The first term is the squared bias of the estimator; the second term is the variance. We can see that both the bias and variance contribute to predictive error. Therefore, when we are trying to come up with the best predictive model, there is an inherent tradeoff between bias and variance of the estimator. By ensuring no bias, unbiased estimators (such as OLS regression) allow no tradeoff. We refer readers to (Hastie et al., 2009) for the formal derivation of the above.

9.2.3 Regularization

In order to allow for a tradeoff, we introduce the concept of *regularization*. Instead of minimizing in-sample error alone, we introduce an additional term and solve the following problem:

$$\text{minimize} \quad \sum_{i}^{n} L(y_i, f(x_i)) + \lambda R(f(\hat{x}_i)) \tag{9.3}$$

The term $R(\hat{f})$ is a regularizer. It penalizes functions that create a lot of variance. The specific form of $R(\hat{f})$ will depend on the model to be estimated, $\hat{f}$, and is typically chosen a priori. The weight given to the regularizer relative to in-sample fit is captured by λ, which controls the amount of regularization and allows us to maximize predictive performance by optimally trading off bias and variance. A key idea in ML is that λ can be optimally derived from the data itself instead of being imposed exogenously. Usually it is selected using cross-validation, by splitting the data into several training and validation sets. (See §9.2.4 for additional details on validation). By repeatedly holding out some subset of the data for validation, we can determine the value of λ that leads to the best prediction for the hold-out data. Therefore, the model is explicitly optimized to make the best out-of-sample prediction given the data. Note that by introducing regularization, we have sacrificed the unbiasedness of the estimator in favor of getting better out of sample predictions.

By empirically making the bias-variance tradeoff, regularization allows us to consider a much broader class of models. For example, we can have models with many more predictors than observations, or models with many parameters such as high degree polynomials, or highly nonlinear models such as decision trees or random forests.

Being able to consider a very rich class of models is important for applications where there is no hypothesized parametric model that can be estimated on the data. For example, in the computer science literature, a commonly studied problem is image recognition, where the goal is to recognize the object in a picture, and the data are pixels. There are, of course, be a lot more predictors than there are data points in this case, and there is no model for how the pixels actually combine to make an image of, say, a dog or a house. As such, in classical ML applications, there is much less focus on modeling than in econometrics or classical statistics. Rather, the focus is on "learning" from the data, but doing so in a rigorous fashion. In such settings, weak assumptions about model structure combined with large data sets that are often characterized by high dimensions and a lot of missing data lead to natural concerns in (1) computation and (2) data overfitting. To deal with these challenges, several techniques have been developed, including regularization, cross-validation, and approximate optimization methods.

9.2.4 Validation

We now discuss how to derive the regularization parameter λ from the data. Note that λ is a hyper-parameter, i.e., it cannot be optimized from training data alone. Indeed, optimization procedures that only use one dataset (training data) will naturally set , λ to zero because they would only be optimizing in-sample fit. Therefore, we introduce the concept of validation. Validation is the process of splitting the data into two or more groups and using the data to identify the optimal λ for the problem.

There are two commonly used procedures for performing validation and deriving the regularization parameter (and other hyper-parameters).

Holdout Validation

Split the data into three sets – 1) training data, S_t, 2) validation data, S_v, and 3) test data, S_e. Initialize the hyper-parameters and estimate the model on S_t. Then predict on S_v. Do this for all values of hyper-parameters considered. Then choose the set of hyper-parameters (and corresponding models) that give the best performance on S_v. Pick this as the final model and predict on the test data S_e, which is the final performance reached by the model.

k=fold Cross Validation

In this case, the training and validation data are kept in one group $S_{tv} = S_t + S_v$ and as usual the test data S_e is kept separate. Now do the following steps: 1. Divide the training and validation data S_{tv} data into k equal subsets (folds) and label them $s = 1, ..., k$. Start with $s = 1$. 2. Pick an initial value for the hyper-parameter. 3. Fit your model using the $k - 1$ subsets other than s. 4. Predict for subset s and measure the associated loss. 5. Stop if $s = k$, otherwise increment s by 1 and go to step 2. Average the best models from each of the k folds and this is the final model. Use this model to predict on the test data S_e, and this is thefinal performance reached by the model.

9.3 CART: Classification and Regression Trees

CART recursively partitions the input space corresponding to a set of explanatory variables into multiple regions and define a local model on each region, which could be as simple as assigning an output value for each region. This type of partitioning can be represented by a tree structure, where each leaf of the tree represents an output region. Consider a dataset with two input variables $\{x_1, x_2\}$ that are used to predict or model an output variable y using a CART. An example tree with three leaves (or output regions) is shown in Figure 9.1. This tree first asks if x_1 is less than or equal to a threshold t_1. If yes, it assigns the value of 1 to the output y. If not (*i.e.*, if $x_1 > t_1$), it then asks if x_2 is less than or equal to a threshold t_2. If yes, then it assigns $y = 2$ to this region. If not, it assigns the value $y = 3$. The chosen y value for a region corresponds to the mean value of y in that region in the case of a continuous output and the dominant y in case of discrete outputs. It is also worth noting that CART can operate with both discrete and continuous input values, an

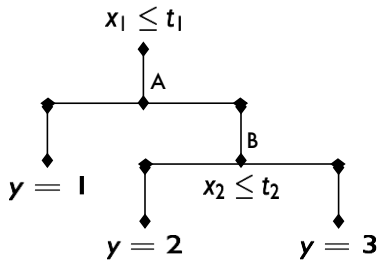

Figure 9.1: Example of a CART model.

algorithmic feature that distinguishes it from regression techniques.

A general tree model can be expressed as follows:

$$y = f(x) = \sum_{k=1}^{K} w_k I(\mathbf{x} \in R_k) = \sum_{k=1}^{K} w_k \phi(\mathbf{x}; \mathbf{v}_k), \tag{9.4}$$

where $\mathbf{x}$ denotes the vector of features or explanatory variables, R_k the k^{th} region of the K regions used to partition the space, w_k the predicted value of y in region k, and $\mathbf{v}_k$ the choice of variables to split on as well as their threshold values for the path to the k^{th} leaf. When y is continuous, w_k is the mean response in the k^{th} region. For classification problems where the outcome is discrete, w_k refers to the distribution of the y's in the k^{th} leaf.

Estimating tree-based models Growing a tree requires us to optimally partition the data to derive the points of split (threshold values of $\mathbf{x}$ at each tree node) as well as the value of y in each leaf. This is a NP-complete problem (Hyafil and Rivest, 1976), and hence commonly solved using a greedy algorithm that incrementally builds the tree by choosing the best feature and the best split value for that feature at each step of the tree construction process. Trees are trained (or 'grown') by specifying a cost function that is minimized at each step of the tree using a greedy algorithm. For a tree that uses two-way splits, the split function determines the best feature (j^*) and its corresponding split value (v^*) as follows:

$$(j^*, v^*) = \arg \min_{j \in (1,\ldots,d), v \in X_j} L(x_i, y_i : x_{ij} \leq v) + L(x_i, y_i : x_{ij} > v), \tag{9.5}$$

where d is the number of input variables, X_j is the domain of values assumed by x_j, and L is the loss function that characterizes the loss in prediction accuracy due to a given split. The cost function that is used for evaluating splits depends on the setting in which the decision tree would be used. For example, the cost function could be the mean squared error of the predictions in the case of the decision tree being used in a regression setting, or the misclassification rate in a classification setting. The split procedure evaluates the costs of using all of the input variables

and at every possible value that a given input variable can assume and chooses a variable (j^*) and the value (v^*) that yields the lowest cost. The stopping criteria for the tree construction can either be based on the cost function or on desired properties of the tree structure. For example, tree construction can be stopped when the reduction in cost as a consequence of introducing a new tree node becomes small or when the tree grows to a pre-defined number of leaves or a pre-defined depth.

The greedy algorithm implies that at each split, the previous splits are taken as given, and the cost function is minimized going forward. For instance, at node B in Figure 9.1, the algorithm does not revisit the split at node A. It however considers all possible splits on all the variables at each node, even if some of the variables have already been used at previous nodes. Thus, the split points at each node can be arbitrary, the tree can be highly unbalanced, and variables can potentially repeat at latter child nodes. All of this flexibility in tree construction can be used to capture a complex set of flexible interactions, which are learned using the data.

CART is popular in the machine learning literature due to many reasons. The main advantage of a simple decision tree is that it is very interpretable – the effect of each variable and its interaction effects are easy to infer. Trees can accept both continuous and discrete explanatory variables, can work with variables that have many different scales, and allow any number of interactions between features (Murphy, 2012). A key advantage of CART over regression models is the ability to capture rich nonlinear patters in data, such as disjunctions of conjunctions (Hauser et al., 2010). CART models are also robust to errors, both in the output and in the explanatory variables, as well as missing explanatory variable values for some of the observations. Further, CART can do automatic variable selection in the sense that CART uses only those variables that provide better accuracy in the regression or classification task. Finally, since the CART technique is non-parametric, it doesn't require data to be linearly separable and its accuracy is not unduly influenced by outliers. These features make CART the best off-the-shelf predictor or classifier available.

Nevertheless, CART has accuracy limitations because of its discontinuous nature and because it is trained using greedy algorithms thus can converge to a local maximum. Also, decision trees tend to overfit data and provide the illusion of high accuracy on training data, only to underperform on the out-of-sample data, particularly on small training sets.

In general, deep trees have high in-sample fit (i.e., low bias), but because of over-fitting problems tend to have high out-of-sample variance. However, we can improve the variance problem by "bagging" or averaging deep trees by boot-strapping. This gives us *Random Forests*. In contrast, shallow trees have poor in-sample fit (high bias), but low out-of-sample variance. Additively adding a series of weak trees that minimize the residual error at each step by a process known as boosting can help us improve the bias problem, while retaining the low variance and other positive properties.

It should be noted that bagging is not a modeling technique; it is simply a variance reduction technique. Boosting however is a method to infer the underlying model $f(x)$. Thus, they are conceptually completely different.

9.4 Boosting

The concept of boosting was first introduced by Schapire (1990) and Freund (1995) who showed that it is possible to generate a strong learner using a combination of weak learners. Soon after, Freund and Schapire (1996) developed the first boosting algorithm, AdaBoost. In an important paper, Breiman (1998) showed that boosting can be interpreted as gradient descent in function space. This view was expanded by Friedman (2001), who showed how boosting can be applied to a large set of loss functions using any underlying weak learner (e.g., CART, logistic functions, regressions). Since then, boosted regression trees, also known as MART (multivariate adaptive regression trees) have been used to solve a variety of prediction problems.

There are many different types of boosting algorithms. Some commonly used ones are listed below:

- Squared loss: L2Boosting
- Exponential loss: AdaBoost
- Log loss: LogitBoost

Note that these boosting algorithms are defined based on the loss function, and not the weak learner used. Later in §9.4.2, we will see that instead of specifying/deriving a separate algorithm for each loss function, we can get a general algorithm for all loss functions.

Boosting is essentially forward stagewise additive modeling and consists of a linear combination weak learners. We start with:

$$f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \phi(x_i; \gamma)) \tag{9.6}$$

Then at each iteration k, we compute:

$$(\beta_k, \gamma_k) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{k-1}(x_i) + \beta\phi(x_i; \gamma) \tag{9.7}$$

and then:

$$f_k(x) = f_{k-1}(x) + \beta_k\phi(x; \gamma_k) \tag{9.8}$$

where ϕ is the weak learner used, γ_k is the parameter vector associated with the weak learner for step k, and β_k is the multiplier for step k. Note that we can use any weak learner here – CART, logistic regressions, regressions, neural nets etc. and the results are valid.

9.4.1 L2Boosting

We first present an example of a simple boosting algorithm with a quadratic loss function, which is referred to as L2Boosting.

With a squared loss function, at the k^{th} step, the loss function takes the form:

$$L(y_i, f_{k-1}(x_i) + \beta_k \phi(x_i; \gamma_k)) = (r_{ik} - \phi(x_i; \gamma_k))^2 \tag{9.9}$$

Here, we have set $\beta_k = 1$ without loss of generality. Thus, at the k^{th} step of L2Boosting, we fit a model $\phi(x, \gamma_k)$ on the residual from the previous step.

9.4.2 Boosting Interpreted as Functional Gradient Descent

Instead of deriving a new boosting algorithm for each loss function, we can come up with a more general algorithm. The key idea here is that boosting can be interpreted as performing gradient descent in function space.

At the k^{th} step, let the gradient of the loss function evaluated at $f(x) = f_{k-1}(x)$ is given by:

$$g_{ik} = -\Big[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\Big]_{f(x)=f_{k-1}(x)}. \tag{9.10}$$

Then, in the usual gradient descent style, we make the following update:

$$f_k = f_{k-1} - \rho_k g_k \tag{9.11}$$

where the step size ρ_k is calculated by

$$\rho_k = \arg\min_{\rho} \sum_{i=1}^{n} L(f_{k-1}(x_i), -\rho_k g_{ik}) \tag{9.12}$$

Then to obtain $\phi(x; \gamma_k)$ that fits the gradient we do:

$$\gamma_k = \arg\min_{\rho} \sum_{i=1}^{n} (-g_{ik} - \phi(x_i; \gamma_k))^2 \tag{9.13}$$

Interestingly, we can skip the step that optimizes the step-size ρ_k and simply use a generic learning rate μ, and the algorithm does not fundamentally change. We stop adding steps when the increment in the fit is below a pre-specified number.

It is worth noting that the gradients are easy to compute for the traditional loss functions. For example, when the loss function is the squared error loss function $1/2(y_i - F(x_i))^2$, the gradient is simply the residual $y_i - F(x_i)$. Thus, boosting techniques can accommodate a broad range of

loss functions and customized by plugging in the appropriate functional form for the loss function and its gradient.

9.4.3 MART: Multivariate

Applying the additive boosting technique to CART produces MART, which has now been shown empirically to be the best classifier available (Caruana and Niculescu-Mizil, 2006; Hastie et al., 2009). MART can be interpreted as a weighted linear combination of a series of regression trees, each trained sequentially to improve the final output using a greedy algorithm.

MART can be viewed as performing gradient descent in the function space using "shallow" regression trees (i.e., trees with a small number of leaves). MART works well because it combines the positive aspects of CART with those of boosting. CART, especially shallow regression trees, tend to have high bias, but have low variance. Boosting CART models addresses the bias problem while retaining the low variance. Thus, MART produces high quality classifiers.

Bibliography

L. Breiman. Arcing Classifier. *The Annals of Statistics*, 26(3):801–849, 1998.

R. Caruana and A. Niculescu-Mizil. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168. ACM, 2006.

Y. Freund. Boosting a Weak Learning Algorithm by Majority. *Information and Computation*, 121(2): 256–285, 1995.

Y. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm. In *ICML*, volume 96, pages 148–156, 1996.

J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, pages 1189–1232, 2001.

T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani. *The Elements of Statistical Learning*, volume 2. Springer, 2009.

J. R. Hauser, O. Toubia, T. Evgeniou, R. Befurt, and D. Dzyabura. Disjunctions of Conjunctions, Cognitive simplicity, and Consideration Sets. *Journal of Marketing Research*, 47(3):485–496, 2010.

I. Hendel and A. Nevo. Measuring the Implications of Sales and Consumer Inventory Behavior. *Econometrica*, 74(6):1637–1673, 2006.

L. Hyafil and R. L. Rivest. Constructing Optimal Binary Decision Trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

M. P. Keane and K. I. Wolpin. The Solution and Estimation of Discrete Choice Dynamic Programming Models by Simulation and Interpolation: Monte Carlo Evidence. *The Review of Economics and Statistics*, pages 648–672, 1994.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

J. Rust. Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher. *Econometrica*, 55(5):999–1033, September 1987.

R. E. Schapire. The Strength of Weak Learnability. *Machine learning*, 5(2):197–227, 1990.

I. Song and P. K. Chintagunta. A Micromodel of New Product Adoption with Heterogeneous and Forward-looking Consumers: Application to the Digital Camera Category. *Quantitative Marketing and Economics*, 1(4):371–407, 2003.

H. Yoganarasimhan. The Value of Reputation in an Online Freelance Marketplace. *Marketing Science*, 27 (5):861–885, 2013.