A Recursive Partitioning Approach for Dynamic Discrete Choice Models in High-Dimensional Settings

Ebrahim Barzegary *
Independent Contributor

Hema Yoganarasimhan*
University of Washington

November 21, 2023

*We would like to thank Simha Mummalaneni and the participants of the 2023 ISMS Marketing Science Conference for feedback that has improved the paper. Please address all correspondence to: ebzgry@gmail.com, hemay@uw.edu.

Abstract

Dynamic discrete choice models are widely employed in marketing and economics to answer substantive and policy questions in settings where individuals' current choices have future implications. However, the estimation of these models is computationally intensive and/or infeasible in high-dimensional settings. Indeed, even specifying the structure for how the utilities/state transitions enter the agent's decision is challenging in high-dimensional settings when we have no guiding theory. In this paper, we present a semi-parametric formulation of dynamic discrete choice models that incorporates a high-dimensional set of state variables, in addition to the standard variables used in a parametric utility function. The high-dimensional variable can include all the variables that are not the main variables of interest but may potentially affect people's choices and must be included in the estimation procedure, i.e., control variables. We present a data-driven recursive partitioning algorithm that reduces the dimensionality of the high-dimensional state space by taking the variation in choices and state transition into account. Researchers can then use the discretized state space from the first-stage with any estimator of their choice in the second-stage. Using Monte-Carlo simulations, we show that our approach can both reduce estimation bias and make estimation feasible in high-dimensional settings.

Keywords: Structural Models, Dynamic Discrete Choice Models, Machine Learning, Recursive Partitioning

1 Introduction

Dynamic discrete choice models are extensively used in marketing and economics to study problems where agents make intertemporal trade-offs; see Aguirregabiria and Mira (2010); Aguirregabiria (2021) for detailed surveys. They have been used in a variety of settings including demand estimation for durable goods (Song and Chintagunta, 2003; Esteban and Shum, 2007; Gowrisankaran and Rysman, 2012; Li, 2019), career decisions (Keane and Wolpin, 1997), demand estimation for storable goods (Erdem et al., 2003; Hendel and Nevo, 2006), intertemporal substitution of consumption (Hartmann, 2006), and search and optimal stopping problems (Kim et al., 2010; Yoganarasimhan, 2013). Models in this class serve two primary purposes. First, they allow researchers to estimate the underlying structural parameters associated with an agent's utility function within an institutional context. Second, they serve as tools to evaluate new (counterfactual) policies, e.g., new pricing policies for durable goods (Nair, 2007), positioning and product strategies by manufacturers and retailers (Hitsch, 2006; Ellickson et al., 2012; Melnikov, 2013), and consumer welfare (Wang, 2015).

In these models, an agent's decisions are dynamic, in the sense that the agent's current decisions affect their future utility stream through future states. As such, agents' current decisions depend on their expectations about the evolution of future states and their own future decisions in those states. Thus, estimating the underlying primitives of an agent's behavior in these settings requires the researcher to incorporate both the current utility an agent receives from a decision as well as her expected future stream of utilities conditional on this decision. Specifically, agents are assumed to maximize expected intertemporal payoffs in each time period. Thus, a key aspect of the standard solution concept is to calculate the value function, which is defined as the expected value of being in a specific state. Formally, this is equal to the discounted sum of the expected utility stream that an agent gets from making optimal choices starting with the current state. Typically, researchers calculate the value function by solving a dynamic programming problem using the Bellman equation (Smammut and Webb, 2010). This is usually done using recursion when the problem is regenerative, or using backward induction when it is finite-horizon.

There are three main issues that make the specification and estimation of these models challenging. First, the well-known *curse of dimensionality* implies that it is exponentially costly to numerically solve for the value function as the dimensionality of the state space increases. Thus, researchers have historically faced a trade-off between including a larger number of state variables (that can help improve the explanatory power of the model and give more realistic counterfactuals) vs. keeping the problem computationally tractable; see §2 for detailed discussions. Second, even if the researcher has access to a high-dimensional set of variables, there is often no guiding theory on

if and how these variables enter the agent's utility function and state transitions. Indeed, in many cases, the researcher may not be interested in the effect of these extraneous variables per se, and may simply wish to control for them to ensure that there is no omitted variable bias. Third, in most settings, the researcher is working with finite or limited samples of data. The combination of finite samples and high-dimensional state variables often implies that there may simply be no observations in many parts of the state space. Thus, the researcher has to reduce the dimensionality of the state space to ensure that there is sufficient data in all parts of the state space for inference.

To address these challenges, we propose a novel approach to model and control for high-dimensional state variables in dynamic discrete choice models. We specify the dynamic discrete choice model in terms of two sets of observable state variables – (1) $\mathbf{X}$ – a set of low dimensional state variables for which the researcher has some theory, i.e., we know that $\mathbf{X}$ affects the utility function/state transition (and may even know the functional form in which it does, though this is not necessary). The structural parameters associated with these variables form the main estimands of interest. (2) $\mathbf{Q}$ – a set of high-dimensional state variables that act as nuisance variables in our estimation process, i.e., they may affect agents' utility function and/or state transition, but we do not know how if and how. For example, in the case of renewal decisions for a cloud subscription service like DropBox or AdobeCloud, $\mathbf{X}$ could consist of prices and product features offered, which we naturally expect to affect purchase decisions. And $\mathbf{Q}$ could consist of a large number of potential factors that may or may not affect demand, e.g., macroeconomic variables such as inflation, detailed data on consumers' product usage patterns (e.g., amount of time spent on the service, regularity of usage, diversity of features used), product descriptions on the product page, reviews and ratings from prominent websites, prices and features of related product categories, etc.

We present an approach to reduce the dimensionality of $\mathbf{Q}$ using a data-driven discretization approach based on recursive partitioning. We assume that there exists a lower dimensional partitioning on $\mathbf{Q}$ ($\mathbf{\Pi}^* : \mathbf{Q} \rightarrow \mathcal{P}$, where $\mathcal{P} = \{1, \ldots, k\}$, i.e., k partitions) such that this partitioning is a sufficient statistic for $\mathbf{Q}$. We define this discretization of $\mathbf{Q}$ as *perfect discretization*, i.e., a discretization where all the observations within a partition have similar decision and transition probabilities (conditional on $\mathbf{X}$ and the unobservable error terms). We then recast the dynamic discrete choice model from the high-dimensional $\mathbf{Q}$-space to the lower-dimensional $\mathcal{P}$-space, and show the equivalence of these two models.

The main challenge now is that we need to learn the discretization $\mathcal{P}$ and estimate the structural parameters of interest, simultaneously. A naive approach here would be to simply use the log-likelihood of the data (as a function of the structural parameters) as the objective function for the recursive partitioning algorithm and generate the splits. However, this approach is practically

infeasible for three reasons. First, under this approach, for each *potential* split, at each step of the partitioning algorithm, we need to solve the dynamic programming problem and estimate the structural parameters. Then we would compare likelihoods across all possible splits and then choose the split that maximizes the empirical likelihood of the data. This is prohibitively expensive when $\mathbf{Q}$ is high-dimensional. Second, the likelihood function contains two sets of outcomes: (i) choice probabilities and (ii) state transition probabilities. Therefore, any data-driven approach to discretize the state space must account for both of these outcomes. This makes the splitting process more complex than the standard CART/causal tree, where there is only one set of estimands to be considered. Third, unlike a standard recursive partitioning algorithm, where we split on all the state variables, here we only split on $\mathbf{Q}$ since $\mathbf{X}$s are known (or assumed) to influence outcomes by definition. As such, we need an algorithm that only splits on a subset of state variables ($\mathbf{Q}$), but considers all the state variables ($\{\mathbf{X}, \mathbf{Q}\}$) to estimate the choice probabilities and state transitions.

We design a recursive partitioning method that addresses all three of these problems. To address the first problem, we separate the discretization problem from the estimation procedure and suggest an objective function that is fully non-parametric. Thus, we do not need to solve the dynamic programming problem at each potential split; as a result, the evaluation of the objective function at each potential split is computationally cheap. To address the second problem, we split our objective function into two sub-objectives, whose relative importance can be learned from the data using model selection/hyperparameter tuning. To address the third problem, we customize the splitting procedure such that it only splits on $\mathbf{Q}$ and not $\mathbf{X}$. Finally, once we have a discretization, we can simply use the learned discretization ($\hat{\mathcal{P}}$) in addition to $\mathbf{X}$ as the set of state variables in conjunction with any standard estimation method, e.g., Nested Fixed Point(NXFP), Two-step methods, MPEC.

Our dimensionality reduction method has several desirable properties that make it convenient to use and applicable to many settings. First, notice that we separate the estimation and discretization tasks and define a general discretization criterion that does not depend on the parametric assumptions of the estimation step. This property makes the algorithm robust to the parametric assumption of the estimation step. Furthermore, the discretization algorithm does not impose any limitation on the estimation method one can use in the second stage, which has advantages. In general, with high-dimensional state spaces, researchers often employ two-step methods that preclude the ability to perform full counterfactuals. However, in this case, researchers can also use full-solution methods such as nested fixed-point that allow for a full range of counterfactuals.[1] Finally, our discretization algorithm has all the desirable properties of standard recursive partitioning-based algorithms, such

[1]Our approach also works for finite horizon settings. In this case, we would simply use backward induction to evaluate the value function on learned discretization ($\hat{P}$) and the state variables $\mathbf{X}$.

as linear time complexity in the dimensionality of $\mathbf{Q}$, robustness to the scale of the $\mathbf{Q}$ variables and to the presence of irrelevant variables in $\mathbf{Q}$. Furthermore, the algorithm can be implemented in a parallelized way, making its execution fast and scalable.

We present two sets of simulations to highlight the empirical performance of our approach. In both simulation studies, we use a high-dimensional version of the standard Rust bus engine problem as the setting. In the first, we allow for a 10-dimensional set of $\mathbf{Q}$ variables and consider 12 different scenarios for how $\mathbf{Q}$ affects state transitions and flow utilities, and how $\mathbf{Q}$ transitions from one state to another. Across all the cases, we show that using our recursive partitioning approach to first partition the state space can help with parameter recovery and bias reduction compared to ignoring the high dimensional state variables. In the second set of simulations, we use a similar setting but focus on the role of the hyperparameter that captures the relative importance of choice probabilities and state transitions in designing partitions. We show that using state transition information helps improve the quality of the partitioning, though the extent of improvement varies with the informativeness of state transition data. Thus, we show that it is important to tune and learn this hyperparameter in a data-driven fashion.

2 Related Literature

First, our paper relates to the literature on the estimation of dynamic discrete choice models in high-dimensional settings. Many different methods have been proposed, including the two-step estimator (Hotz and Miller, 1993), where the expected value function can be specified as an analytical function of the Conditional Choice Probability (CCP) of certain decisions (under certain assumptions). Arcidiacono and Ellickson (2011) show that estimation can be further simplified in many settings by exploiting the "finite-dependence property", by specifying the expected value functions as a function of terminal or renewal decisions. Chernozhukov et al. (2022) build on the two-step approach of Hotz and Miller (1993) and allow the researcher to estimate the CCPs and flow utilities flexibly using non- or semi-parametric methods and correct for the bias from the first-stage estimation in the second step. While these two-step methods can make estimation feasible in high-dimensional settings, they nevertheless have a few limitations. First, the researcher still needs to estimate non-parametric CCPs at different combinations of the state space, which is not feasible to do accurately in really high-dimensional settings with finite data. Second, in order to conduct counterfactual analysis, researchers usually need to solve the full model at least once using NFXP, which may again not be feasible in very high-dimensional settings. Finally, CCP methods do not provide insight into whether certain variables affect the flow utilities/state transitions, and still require the researcher to make assumptions if/how these variables affect outcomes.

Other methods to simplify estimation in high-dimensional settings include state space reduction

by relying on inclusive value sufficiency (Gowrisankaran and Rysman, 2012), or approximation of the value function using parametric policy iteration (Benitez-Silva et al., 2000) or sieve value function iteration (Arcidiacono et al., 2012). Typically, these methods work well when there exists a set of basis functions that provide a good approximation of the value function (Rust, 2000; Powell, 2007). A more recent body of literature has tried to use the advances in machine learning and methods such as neural networks to solve dynamic discrete choice problems. For example, Norets (2012) uses an artificial neural network to estimate the value function, taking state variables and parameters of interest as input. Su and Judd (2012) proposes yet another approach to solve the curse of dimensionality problem in dynamic discrete choice modeling, called MPEC. They formulate the dynamic discrete choice problem as a constrained maximization problem where the likelihood function is the maximization objective, and the bellman equations are the constraints. Nevertheless, the MPEC algorithm is still not applicable in high-dimensional settings without proper discretization of the state space because it requires us to estimate the value function at each point in the state space. Finally, all of these methods require the researcher to specify the functional form of the flow utility function and do not necessarily help with variable selection and/or state-space reduction when we have no a priori theory on which variables affect the problem and how.

Our algorithm adds to this literature by offering a method to break the curse of dimensionality through data-driven discretization of the state space. We show that state-space discretization and parameter estimation are two separate tasks. Researchers can thus use our recursive partitioning algorithm to reduce the dimensionality of a high-dimensional state space $\mathbf{Q}$ to a one-dimensional categorical variable $\mathcal{P}$ in the first stage, and then use $\mathcal{P}$ in addition to other independent variables $\mathbf{X}$ for parameter estimation in the second stage. Note that in the second step, we can use any of the existing estimation algorithms, e.g., NXFP, two-step.

Our paper also contributes to the literature on estimation using recursive partitioning. Breiman et al. (1984) proposed the original Classification and Regression Trees (CART) algorithm, for prediction using recursive partitioning. Ensemble methods such as random forests combine several trees to produce better performance than a single tree (Breiman, 2001). While recursive partitioning has traditionally been used for prediction tasks, recently, there has been interest in using it for causal estimation of heterogeneous treatment effects using methods such as causal tress and Generalized Random Forests (Athey and Imbens, 2016; Athey et al., 2019). Our algorithm adds to this literature by proposing a novel use of recursive partitioning to estimate dynamic discrete choice models by formulating a non-parametric objective function that incorporates both choice probabilities and state transitions to reduce the dimensionality of the state-space.

Finally, our paper adds to the nascent but growing literature that combines structural models

with machine learning. Jiang et al. (2020) develop a choice model for high-dimensional settings. Singh et al. (2023) exploit the permutation invariance property to propose nonparametric estimators to overcome the curse of dimensionality that is inherent in the non-parametric estimation of choice functions. Wei and Jiang (2022) discuss how neural networks can be used for parameter estimation in structural models. However, all these approaches study static choice models rather than dynamic choice models, which is our focus.

3 Problem Definition

We consider the discrete choice problem from the perspective of a forward-looking single agent, denoted by $i \in \{1 \dots N\}$. In every period t, the agent chooses between $j = 1 \dots J.$ options. i's decision in period t is denoted by d_{it}, and $d_{it} = j$ indicates that the agent i has chosen the option j in period t. The agent's decision is not only a function of her utility in her current state (s_{it}) but also the expectation of her utility in all her future states given the decision d_{it}. We assume that the agent's state s_{it} is composed of three sets of variables.

1. A set of observable low-dimensional state variables $x_{it} \in \mathbf{X}$.
2. A set of observable high-dimensional state variables $q_{it} \in \mathbf{Q}$.
3. Unobservable state variable ϵ_{it}, which is a $J \times 1$ vector each associated with one of the alternatives observed by the agent, but not by the researcher.

$\mathbf{X}$ represents state variables for which we have some a priori theory, i.e., we know the parametric form in which they enter the utility function. The structural parameters associated with these variables form the main estimands of interest. $\mathbf{Q}$ denotes state variables that act as nuisance variables in our estimation exercise – they are not the main variables of interest, and we do not have a theory for if and how they influence users' decisions and state transitions. However, ignoring them can potentially bias the estimates of interest. Note that it is also possible to simply assume that $\mathbf{X}$ is null and simply put all the state variables in $\mathbf{Q}$ and learn how $\mathbf{Q}$ affects utilities and state transitions.

In each period t, agent i derives an instantaneous flow utility $u(s_{it}, d_{it})$, which is a function of her decision d_{it} and her state variables $s_{it} = \{x_{it}, q_{it}, \epsilon_{it}\}$. The per-period utility is additively separable as follows:

$$u(s_{it}, d_{it} = j) = \bar{u}(x_{it}, q_{it}, d_{it} = j; \theta_1) + \epsilon_{itj}, \tag{1}$$

where ϵ_{itj} is the error term associated with j^{th} option at time t, $\bar{u}(x_{it}, q_{it}, d_{it} = j; \theta_1)$ is the deterministic part of the utility from making decision j in state x_{it}, q_{it}, and θ_1 is the set of structural parameters associated with the deterministic part of utility. The state s_{it} transitions into new, but not necessarily different, values in each period following decision d_{it}. We make three standard

assumptions on the state transition process – first-order Markovian, conditional independence, and IID error terms. These assumptions imply that: (i) $\{s_{it}, d_{it}\}$ are sufficient statistics for s_{it+1}, (ii) error terms are independent over time, and (iii) errors in the current period affect states tomorrow only through today's decisions. Thus, we have:

$$\Pr(x_{it+1}, q_{it+1}, \epsilon_{it+1}|x_{it}, q_{it}, \epsilon_{it}, d_{it}) = \Pr(\epsilon_{it+1}) \Pr(x_{it+1}, q_{it+1}|x_{it}, q_{it}, d_{it}) \tag{2}$$

We denote the state transition function $\Pr(x_{it+1}, q_{it+1}|x_{it}, q_{it}, d_{it})$ as $g(x_{it+1}, q_{it+1}|x_{it}, q_{it}, d_{it}; \theta_2)$, where θ_2 captures the parameters associated with state transition.[2]

Each period, the agent takes into account the current period payoff as well as how her decision today will affect the future, with the per-period discount factor given by β. She then chooses d_{it} to sequentially maximize the expected discounted sum of payoffs $\mathbb{E}\left[\sum_{\tau=t}^{\infty} \beta^{\tau} u(s_{i\tau}, d_{i\tau})\right]$. Our goal is to estimate the set of structural parameters $\theta = \{\theta_1, \theta_2\}$ that rationalizes the observed decisions and the states in the data, which are denoted by $\{(x_{i1}, q_{i1}, d_{i1}), \ldots, (x_{it}, q_{it}, d_{it}), \ldots, (x_{iT}, q_{iT}, d_{iT})\}$ for agents $i \in \{1, \ldots, N\}$ for T time periods.

3.1 Challenges

The standard solution is to use a maximum likelihood method and estimate the set of parameters that maximizes the likelihood of observing the data. Given the first-order Markovian and conditional independence assumptions, we can write the likelihood function and estimate the structural parameters as follows:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \log \hat{p}(d_{it}|x_{it}, q_{it}; \theta_1) + \sum_{t=2}^{T} \log \hat{g}(x_{it}, q_{it}|x_{it-1}, q_{it-1}, d_{it-1}; \theta_2) \right) \tag{3}$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}(\theta)$$

where $\hat{p}(.)$ is the predicted choice probabilities.

However, there are three main challenges in estimating a model in this setting:

- **Theory:** First, the researcher may lack theory on how the high-dimensional variables q enter the agents' utility function. For instance, if we consider the example of high-dimensional usage variables in a software subscription decision, we do not have much theoretical guidance on which product usage variables affect users' utility and how. As such, we cannot make parametric assumptions on the effect of $\mathbf{Q}$ on decisions and state transitions, or hand-pick a subset of these

[2]Both the utility function and state transition can also be estimated non-parametrically. In that case, θ_1 and θ_2 would simply denote the utility and state transition at a given combination of state variables.

variables to include in our model.

- **Data:** Second, in a high-dimensional setting, we may not have sufficient data in all areas of the state space to model the flow utility and the transitions to/from a given set of state variables.
- **Estimation:** Finally, estimation of a discrete choice dynamic model in a high dimensional setting is often computationally infeasible and/or costly. Rust (1987)'s nested-fixed point algorithm requires us to calculate the value function at each combination of the state variables at each iteration of the estimation, which is infeasible in a large state space setting. While two-step methods can overcome estimation challenges in large state spaces by avoiding value function iteration, they nevertheless need non-parametric estimates of Conditional Choice Probabilities (CCPs) at each combination of state variables (Hotz and Miller, 1993). This is not possible in a very high-dimensional setting with finite data.

Thus, in a finite data high-dimensional setting where we lack guiding theory, it is not feasible to specify a utility function over q and/or estimate a dynamic discrete choice model using conventional methods. Therefore, we need a data-driven approach that reduces the dimensionality of $\mathbf{Q}$ in an intelligent fashion.

3.2 Dimensionality Reduction using Discretization

Our solution is to recast the problem by mapping $\mathbf{Q}$ to a lower-dimensional space through data-driven discretization such that $\mathbf{\Pi} : \mathbf{Q} \to \mathcal{P}$, where $\mathcal{P} = \{1, \ldots, k\}$. The goal is similar in spirit to that of Classification and Regression Trees (CART) algorithm used for outcome prediction (Breiman et al., 1984) and the Causal Tree algorithm for estimation of conditional average treatment effects (Athey and Imbens, 2016). These algorithms discretize the covariate space to minimize the heterogeneity of the statistic of interest within a partition. For example, the CART algorithm discretizes the covariate space into disjoint partitions such that observations in the same partition have similar outcome values/classes. Similarly, the Causal Tree algorithm discretizes the state space such that observations within the same partition have similar treatment effects. While the exact approaches from the static estimation of CART/causal tree cannot be directly translated to dynamic discrete choice models, we build on the notion of "similarity within a partition" to adapt the high-level intuition from these models to our setting. Therefore, our first step is to outline the characteristics of a suitable discretization. In §3.2.1 we formally define the term *perfect discretization* as a discretization where the observations within a partition behave similarly. Then in, §3.2.2, we show how the estimation problem can be reformulated for a lower-dimensional space by exploiting this definition.

3.2.1 Properties of a Good Discretization

We now discuss some basic ideas that a good discretization should capture. First, some variables in $\mathbf{Q}$ may be irrelevant to our estimation procedure, i.e., they have no effect on flow utilities or state transitions. A good data-driven discretization should be able to neglect these variables and thus be robust to irrelevant variables. Second, our discretization approach has to be entirely non-parametric since we do not have any theory on how variables in $\mathbf{Q}$ affect agents' utilities and state transitions. Finally, the discretization should be generalizable, i.e., it should be valid outside the training data. Formally, we define the term *perfect discretization* as follows:

Definition 1. A discretization $\mathbf{\Pi}^* : \mathbf{Q} \to \mathcal{P}$ is perfect if all the points in the same partition have the same choice probabilities and incoming and outgoing transition probabilities. That is, for any two points q, q' in a partition $\pi \in \mathcal{P}$, we have:

$$\forall x, x' \in \mathbf{X}, q'' \in \mathbf{Q}, j \in \mathbf{J} : \begin{cases} \Pr(j|x,q) = \Pr(j|x,q') & \text{(4a)} \\ \Pr(x',q''|x,q,j) = \Pr(x',q''|x,q',j) & \text{(4b)} \\ \Pr(x,q|x',q'',j) = \Pr(x,q'|x',q'',j) & \text{(4c)} \end{cases}$$

The first equality ensures that the decision probabilities are similar for data points within a given partition $\pi \in \mathcal{P}$. The second equality asserts the equality of transition probabilities *from* any two points q, q' within the same partition $\pi \in \mathcal{P}$. Finally, the last equality implies that the transition probabilities *to* any two points within a partition π are equal. Together, these three equalities imply that all the observations within the same $\pi \in \mathcal{P}$ are similar from both modeling and estimation perspective. Therefore, we do not need to model the heterogeneity within the partition π. Instead, modeling agents' behavior at the level of $\mathcal{P}$ is sufficient.

3.2.2 Formulation of DDC in a discretized space

We now use the definition of perfect discretization and translate the DDC estimation from the $\mathbf{Q}$-space to the $\mathcal{P}$-space. Because observations within each partition in a perfect discretization behave similarly, we can project the problem from the $\mathbf{Q}$-space to the $\mathcal{P}$-space. That is, $\mathbf{\Pi}^*$ is a sufficient statistic for the estimation of state transition and decision probabilities. We can therefore write the probabilities of choices and state transitions in the $\mathcal{P}$-space as follows:

$$\forall x, x' \in \mathbf{X}, q'' \in \mathbf{Q}, j \in \mathbf{J} : \begin{cases} \Pr(j|x,q) = \Pr(j|x,\mathbf{\Pi}^*(q)) & \text{(5a)} \\ \Pr(x',q''|x,q,j) = \Pr(x',q''|x,\mathbf{\Pi}^*(q),j) & \text{(5b)} \\ \Pr(x,q|x',q'',j) = \dfrac{\Pr(x,\mathbf{\Pi}^*(q)|x',q'',j)}{N(x,\mathbf{\Pi}^*(q))} & \text{(5c)} \end{cases}$$

where $N(x, \mathbf{\Pi}^*(q))$ is the number of observations in state $\{x, \mathbf{\Pi}^*(q)\}$.

These three equalities are the counterparts of the relationships shown in Equation (4) in the $\mathcal{P}$-space. According to the first equality in Equation (5), if the choice probabilities for the observations within a partition $\pi \in \mathcal{P}$ are the same, then $\{\mathbf{X}, \mathcal{P}\}$ is a sufficient statistic to capture the choice probabilities. The same argument is true for outgoing state transitions. If the probabilities of transition to other states from all points in a partition are similar, we can use the partition instead of points to specify transition (as shown in the second relationship in Equation (5)). Finally, when the probability of transition into all the points within a partition is similar, the probability of moving to a specific point is equal to the probability of transitioning into the partition divided by the number of observations within that partition. That is:

$$\begin{aligned}\Pr(x, \mathbf{\Pi}^*(q)|x', q', j) &= \sum_{q'' \in \mathbf{\Pi}^*(q)} \Pr(x, q''|x', q', j) \\ &= N(x, \mathbf{\Pi}^*(q)) \Pr(x, q|x', q', j),\end{aligned}$$

which gives us the third relationship in Equation (5).

We now use the relationships in Equation (5) to reformulate the log-likelihood in Equation (3). Given a perfect partitioning $\mathbf{\Pi}^*$, the log-likelihood can be written as:

$$\begin{aligned}\mathcal{L}(\theta, \mathbf{\Pi}^*) = &\sum_{i=1}^{N} \sum_{t=1}^{T} \log p(d_{it}|x_{it}, \mathbf{\Pi}^*(q_{it}); \theta_1, \mathbf{\Pi}^*) \\ &+ \sum_{i=1}^{N} \sum_{t=2}^{T} \log \frac{g(x_{it}, \mathbf{\Pi}^*(q_{it})|x_{it-1}, \mathbf{\Pi}^*(q_{it-1}), d_{it-1}; \theta_2, \mathbf{\Pi}^*)}{N(x_{it}, \mathbf{\Pi}^*(q_{it}); \mathbf{\Pi}^*)}\end{aligned} \tag{6}$$

Note that the second term in the log-likelihood is obtained by combining the second and third terms in Equation (5) as: $\Pr(x', q''|x, q, j) = \Pr(x', q''|x, \mathbf{\Pi}^*(q), j) = \frac{\Pr(x', \mathbf{\Pi}^*(q'')|x', \mathbf{\Pi}^*(q), j)}{N(x, \mathbf{\Pi}^*(q''))}$.

Next, we can formulate the utility function in the $\mathbf{\Pi}^*(q)$-space as follows:

$$u(s_{it}, d_{it}) = \bar{u}(x_{it}, \mathbf{\Pi}^*(q_{it}), d_{it}; \theta_1, \mathbf{\Pi}^*) + \epsilon_{itj} \tag{7}$$

Finally, we can also write the value function and the choice-specific value function in terms of the discretized state space. Conceptually, once we have a perfect discretization $\mathbf{\Pi}^*$, we can treat $\mathbf{\Pi}^*(q)$ as a categorical variable in addition to $\mathbf{X}$, and ignore q. As such, all the methods available for the estimation of dynamic discrete choice models (e.g., nested fixed point, two-step estimators) are directly applicable here, with $\{\mathbf{X}, \hat{\mathcal{P}}\}$ as the state space. Thus, all the consistency and efficiency

properties of the estimator used would directly translate to this setting, i.e., are valid conditional on the state variables, $\{\mathbf{X}, \hat{\mathcal{P}}\}$.

4 Our Discretization Approach

We now present our discretization algorithm. We first explain the recursive partitioning method for a general objective function in §4.1. Next, in §4.2, we present a recursive partitioning algorithm for the dynamic discrete choice model outlined above and discuss its properties. Finally, in §4.3, we discuss the practical aspects of the algorithm such as model selection and hyper-parameter tuning.

4.1 Discretization using Recursive Partitioning

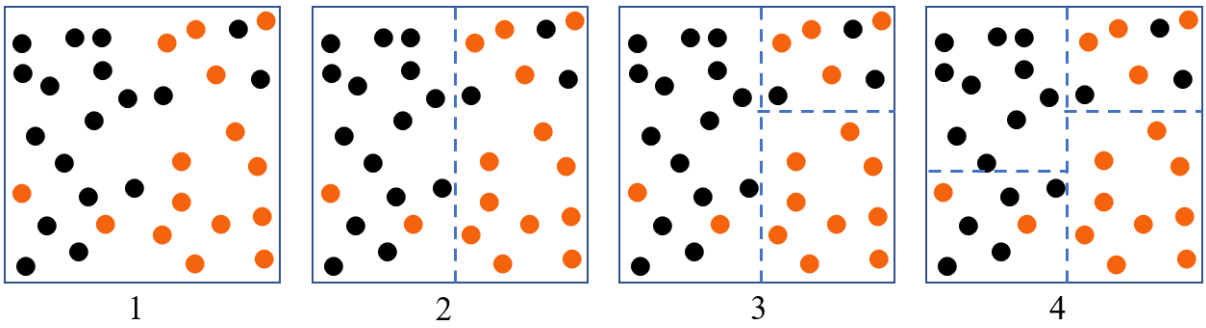


Figure 1: An example of recursive partitioning for a classification task with two explanatory variables and two outcome classes (denoted by orange and black dots).

Recursive partitioning is a meta-algorithm for partitioning a covariate space into disjoint partitions to maximize an objective function. In each iteration, the algorithm uses an objective function as the criterion for selecting the next split among all the potential candidate splits. A split divides a partition into two along one of the variables in the covariate space. The following pseudo-code presents a general recursive partitioning algorithm, where the goal is to maximize the objective function $\mathcal{F}(\mathbf{\Pi})$.

- Inputs: Objective function $\mathcal{F}(\mathbf{\Pi})$ that takes a partitioning $\mathbf{\Pi}$ as input and outputs a score.
- Initialize $\mathbf{\Pi}_0$ as one partition equal to the full covariate space.
- Do the following until the stopping criterion is met, or $\mathcal{F}(\mathbf{\Pi}_r) = \mathcal{F}(\mathbf{\Pi}_{r-1})$
  - For every $\pi \in \mathbf{\Pi}_r$, every $q \in \mathbf{Q}$, and every value $v \in \text{range}(q)$ in π

    $$\Delta(\pi, q, v) = \mathcal{F}(\mathbf{\Pi}_r + \{\pi, q, v\}) - \mathcal{F}(\mathbf{\Pi}_r)$$

  - $\{\pi', q^*, v^*\} = \text{argmax}_{\{\pi,q,v\}} \Delta(\pi, q, v)$

– $\mathbf{\Pi}_{r+1} = \mathbf{\Pi}_r + \{\pi', q^*, v^*\}$

Figure 1 shows three iterations of recursive partitioning applied to a classification task. The partitioning $\mathbf{\Pi}$ maps the two-dimensional covariate space into four different partitions.

4.2 Recursive Partitioning for Dynamic Discrete Choice Models

The ultimate goal of our discretization exercise is to estimate θ by maximizing the likelihood function in Equation (6). To do so, we first need to find a perfect discretization, i.e., a discretization that satisfies Equation (4). Thus, an intermediate goal is to find the partitioning $\mathbf{\Pi}^*$. The key question then becomes what should be the objective function for the recursive partitioning algorithm that helps us achieve the two goals discussed above. A naive approach would be to simply use the log-likelihood shown in Equation (6). However, this is problematic because of three reasons.

- First, this likelihood is a function of both $\mathbf{\Pi}^*$ and θ. Thus, a naive implementation of the recursive partitioning algorithm would require us to estimate the optimal θ in every iteration for a given $\mathbf{\Pi}_r$. However, estimating θ is computationally expensive because we need to calculate the discounted future utility (or expected value function) associated with a given choice to estimate the primitives of agents' utility functions fully. Doing this at every *potential* split in each iteration of the algorithm is computationally expensive (and infeasible when the $\mathbf{Q}$-space is large).
- Second, the likelihood function contains two sets of outcomes: (i) choice probabilities and (ii) state transition probabilities. Any data-driven approach to discretize the state space must account for both of these outcomes. This makes the splitting process more complex than usual recursive partitioning algorithms such as CART/causal tree, where there is only one set of estimands to be considered.
- Third, unlike a standard recursive partitioning algorithm, where we split on all the state variables, here we only split on $\mathbf{Q}$ since $\mathbf{X}$s are known (or assumed) to influence outcomes by definition. As such, we need an algorithm that only splits on a subset of state variables ($\mathbf{Q}$), but considers all the state variables ($\{\mathbf{X}, \mathbf{Q}\}$) to estimate the choice probabilities and state transitions; see Figure 2 for an example.

We design a recursive partitioning method that addresses all these three problems. To address the first problem, we separate the discretization problem from estimation and suggest an objective function that is fully non-parametric, i.e., is not dependent on θ. Thus, its calculation is computationally cheap. To address the second problem, we split our objective function into two sub-objectives, whose relative importance can be learned from the data using model selection. Finally, to address the third problem, we customize the splitting procedure such that it only splits on $\mathbf{Q}$ and not $\mathbf{X}$. We

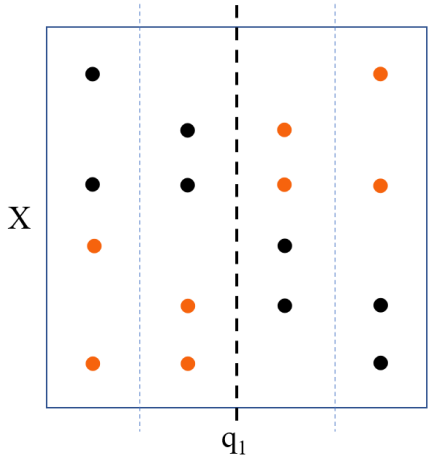


Figure 2: The dashed lines are candidate splits. Orange and black dots are observations associated with decisions 1 and 2 respectively. Note that agents' choices are a function of both X and q_1.

discuss these ideas in detail below.

4.2.1 Nonparametric Objective Function

As we discussed in §3.2.2, we can estimate the primitives of agents' utilities and state transitions by maximizing the log-likelihood shown in Equation (6). This likelihood is a function of both the primitive parameters (θ) and the discretization $\mathbf{\Pi}^*$. Nevertheless, we do not need to solve the maximization problem on both dimensions simultaneously. Conceptually, the discretization goal is to separate $\mathbf{Q}$ into buckets such that observations within the same bucket behave similarly. A key insight here is that we can achieve this objective without estimating θ, by simply using the non-parametric estimates of choice probabilities and state transitions observed in the data. That is, we can re-formulate the objective function from Equation (6) in non-parametric terms. Our proposed objective function is thus a weighted sum of the nonparametric equivalents of the first and second components of Equation (6) as shown below.

$$\mathcal{F}(\mathbf{\Pi}) = \mathcal{F}_{dc}(\mathbf{\Pi}) + \lambda\, \mathcal{F}_{tr}(\mathbf{\Pi}) \tag{8}$$

where $\mathbf{\Pi}$ is a partitioning function, and $\mathcal{F}_{dc}(\mathbf{\Pi})$ and $\mathcal{F}_{tr}(\mathbf{\Pi})$ are:

$$\begin{aligned}\mathcal{F}_{dc}(\mathbf{\Pi}) &= \sum_{i=1}^{N}\sum_{t=1}^{T} \log \frac{N(x_{it}, \mathbf{\Pi}(q_{it}), d_{it}; \mathbf{\Pi})}{N(x_{it}, \mathbf{\Pi}(q_{it}); \mathbf{\Pi})} \\ &= \sum_{x\in\mathbf{X}}\sum_{\pi\in\mathbf{\Pi}}\sum_{j\in\mathbf{J}} N(x, \pi, j; \mathbf{\Pi}) \log \frac{N(x, \pi, j; \mathbf{\Pi})}{N(x, \pi; \mathbf{\Pi})}\end{aligned} \tag{9}$$

$$\mathcal{F}_{tr}(\mathbf{\Pi}) = \sum_{i=1}^{N}\sum_{t=2}^{T} \log \frac{N(x_{it}, \mathbf{\Pi}(q_{it}), x_{it-1}, \mathbf{\Pi}(q_{it-1}), d_{it-1}; \mathbf{\Pi})}{N(x_{it-1}, \mathbf{\Pi}(q_{it-1}), d_{it-1}; \mathbf{\Pi}) N(x_{it}, \mathbf{\Pi}(q_{it}))}$$
$$= \sum_{x\in\mathbf{X}}\sum_{\pi\in\mathbf{\Pi}}\sum_{j\in\mathbf{J}}\sum_{x'\in\mathbf{X}}\sum_{\pi'\in\mathbf{\Pi}} N(x, \pi, x', \pi', j; \mathbf{\Pi}) \log \frac{N(x, \pi, x', \pi', j; \mathbf{\Pi})}{N(x, \pi; \mathbf{\Pi}) N(x', \pi', j; \mathbf{\Pi})} \tag{10}$$

The function $N(.)$ counts the number of observations for a given condition. For example, $N(x, \pi, j)$ is the number of observations where the agent chose decision j in state $\{x, \pi\}$ and $N(x, \pi, x', \pi', j)$ is the number of observations that chose decision j in state $\{x', \pi'\}$, and transitioned to $\{x, \pi\}$.

The weighting parameter λ is a multiplier that specifies the relative importance of the state transition likelihood in comparison to decision likelihood in our algorithm. To make this parameter more intuitive and generalizable across different datasets, we decompose it as follows:

$$\lambda = \lambda_{adj} \times \lambda_{rel}, \quad \text{where } \lambda_{adj} = \frac{\mathcal{F}_{dc}(\mathbf{\Pi}_0)}{\mathcal{F}_{tr}(\mathbf{\Pi}_0)}. \tag{11}$$

Here, $\mathbf{\Pi}_0$ is the full covariate space of $\mathbf{Q}$ as one partition (i.e., the baseline case that ignores $\mathbf{Q}$). λ_{rel} is a hyperparameter that captures the relative importance of state transition and decision likelihoods in our objective function and should be learned from the data using cross-validation. For example, $\lambda_{rel} = 2$ implies that the recursive partitioning algorithm values one percentage lift in $\mathcal{F}_{tr}$ twice as much a one percentage lift in $\mathcal{F}_{dc}$ when selecting the next split. The optimal λ_{rel} can vary with the application. In settings where there is more information in the state transition, the optimal λ_{rel} will be higher whereas a smaller λ_{rel} is better when the choice probabilities are more informative. Therefore, it is important to choose the right value of λ_{rel} to prevent over-fitting and learn a good discretization.

A couple of additional points of note regarding our proposed objective function. First, the first lines in both Equations (9) and (10) are aggregating observations over individuals and time whereas the second lines are aggregating over all possible decisions and state transitions weighted by their occurrence. While the two representations are equivalent, we use the latter one in the rest of the paper since it is more convenient. Second, the objective function in Equation (8) is the non-parametric version the log-likelihood in Equation (6) (with the hyperparameter λ added in for data-driven optimization). The terms $\frac{N(x_{it}, \mathbf{\Pi}(q_{it}), d_{it}; \mathbf{\Pi})}{N(x_{it}, \mathbf{\Pi}(q_{it}); \mathbf{\Pi})}$, and $\frac{N(x_{it}, \mathbf{\Pi}(q_{it}), x_{it-1}, \mathbf{\Pi}(q_{it-1}), d_{it-1}; \mathbf{\Pi})}{N(x_{it-1}, \mathbf{\Pi}(q_{it-1}), d_{it-1}; \mathbf{\Pi})}$ are the non-parametric counterparts of $\Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}); \theta_1, \mathbf{\Pi})$ and $\Pr(x_{it}, \mathbf{\Pi}(q_{it})|x_{it-1}, \mathbf{\Pi}(q_{it-1}), d_{it-1}; \theta_2, \mathbf{\Pi})$, respectively. Therefore, maximizing $\mathcal{F}(\mathbf{\Pi})$ is equivalent to maximizing the original likelihood function.

4.2.2 Algorithm Properties

We now establish two key properties of the recursive partitioning algorithm proposed here. First, as shown in Web Appendix A.1, the non-parametric log-likelihood shown in Equation (6) is non-decreasing at each iteration of the algorithm. Formally, we have:

$$\mathcal{L}(\theta_r^*, \mathbf{\Pi}_r) \leq \mathcal{L}(\theta_{r+1}^*, \mathbf{\Pi}_{r+1})$$
$$\text{where} \begin{cases} \theta_r^* = \text{argmax}_\theta \, \mathcal{L}(\theta, \mathbf{\Pi}_r) \\ \theta_{r+1}^* = \text{argmax}_\theta \, \mathcal{L}(\theta, \mathbf{\Pi}_{r+1}) \end{cases}$$

Second, as shown in Web AppendixA.2, for $\lambda > 0$, the final discretization $\mathbf{\Pi}^* = \text{argmax}_{\mathbf{\Pi}} \mathcal{F}(\mathbf{\Pi})$ converges to a perfect discretization (under certain conditions). Together, these two properties ensure that: (a) the algorithm will increase the likelihood at each step and converge, and (b) upon convergence, it will achieve a perfect discretization. Of course, in practice, we may choose a λ that does some bias-variance trade-off to avoid over-fitting and stop before we reach the perfect discretization to ensure that the discretization generalizes outside the training data.

Further, our proposed algorithm shares the desirable properties of other recursive partitioning-based algorithms. First, it has linear time complexity with respect to the dimensionality of $\mathbf{Q}$ (Sani et al., 2018) – if the number of dimensions in $\mathbf{Q}$ doubles, the algorithm's runtime at most doubles. Second, the proposed algorithm is robust to the scale of the state variables and only depends on the ordinality of the state variable. As such, any changes in the scale of variables in $\mathbf{Q}$ do not change the estimated partition. Finally, since the algorithm splits on a variable only if it increases the log-likelihood shown in Equation (8), it is robust to the presence of irrelevant state variables. Together, these properties allow the researcher to include all potential observable variables that might affect the agents' decisions in $\mathbf{Q}$ without significantly increasing the compute cost. This is valuable in the current data-abundant era, where firms have massive amounts of user-level data but lack theoretical insight into the effect (if any) of these variables on users' decisions. Indeed, one of the advantages of the method is that it allows the researcher to make post-hoc inference or theory discovery in a data-rich environment.

Nevertheless, like other recursive partitioning algorithms, our algorithm does not guarantee consistency. This is because recursive partitioning algorithms are inherently unable to capture certain data patterns and are greedy by design; see Assumption 2 in Web Appendix§A and Biau et al. (2008) for further details. Thus, it is not feasible for us to provide a formal consistency proof for the algorithm. Nevertheless, as we show in §5, the algorithm performs well in a wide range of simulations.

4.3 Hyperparameter Optimization and Model Selection

Like other machine learning approaches, our recursive partitioning algorithm also needs to address the bias-variance trade-off. If we discretize $\mathbf{Q}$ into too many small partitions, the set of partitions (and the corresponding estimates of choice and state transition probabilities) will not generalize beyond the training data. Thus, we need a set of hyperparameters that constrain or penalize model complexity. We can then tune these hyperparameters using a validation procedure.[3]

4.3.1 Set of Hyperparameters

We start with two hyperparameters that are commonly used in other recursive partitioning-based models: *minimum number of observations* and *minimum lift*.[4] The former stops the algorithm from making very small partitions by ruling out splits (at any given iteration) that produce partitions with observations fewer than the *minimum number of observations*. The latter prevents over-fitting by stopping the partitioning process if the next split does not increase the objective function by the *minimum lift*, which is defined as $\frac{\mathcal{F}(\mathbf{\Pi}_{r+1})-\mathcal{F}(\mathbf{\Pi}_r)}{\mathcal{F}(\mathbf{\Pi}_r)}$. In addition to these two standard hyperparameters, we also include another one: *maximum number of partitions*, which stops the recursive partitioning after the algorithm has reached the *maximum number of partitions*. This hyper-parameter not only controls over-fitting, but can also help with identification concerns because it allows the researchers to restrict the number of partitions, and hence the number of estimands. Together, these three hyperparameters ensure that the algorithm does not over-fit and produces a generalizable partition that is valid out-of-sample.

In addition to these three hyper-parameters that constrain partitioning, we have another key hyperparameter, λ_{rel}, that shapes the direction of partitioning. As discussed earlier, λ_{rel} captures the relative importance of the two parts of the objective function when selecting the next split. If λ_{rel} is set close to zero, the discretization procedure prioritizes splits that explain the choice probabilities. As λ_{rel} increases, the recursive partitioning tends to choose splits that explain the variation in the state transition. λ_{rel} can be either tuned using a validation procedure or set manually by the researched based on their intuition or the requirements of the problem at hand. We present some simulations on the importance of picking the right value of λ_{rel} in §5.3.

4.3.2 Score function

Let η denote the set of all hyperparameters. To pick the right η for a given application setting, we need a measure of performance at a given η. Then, we can consider different values of η and

[3] See Hastie et al. (2009) for a detailed discussion of the pros-cons of different validation procedures.

[4] See the hyperparameters in Random Forest (Breiman, 2001), XGBoost (Chen and Guestrin, 2016), and Generalized Random Forest (Athey et al., 2019).

select the one that maximizes out-of-sample performance (on the validation data). The model's performance is measured by calculating our objective function on a validation set using the training set's estimated values. Formally, our score function for a given set of hyperparameters is:

$$score(\eta) = \frac{1}{1+\lambda_{rel}} \left(\sum_{x \in \mathbf{X}} \sum_{\pi \in \mathbf{\Pi}^*} \sum_{j \in \mathbf{J}} N^{val}(x, \pi, j; \mathbf{\Pi}^*) \log \frac{N^{trn}(x, \pi, j; \mathbf{\Pi}^*)}{N^{trn}(x, \pi; \mathbf{\Pi}^*)} \right. \tag{12}$$

$$\left. + \lambda_{rel} \lambda_{adj}^{val} \sum_{x \in \mathbf{X}} \sum_{\pi \in \mathbf{\Pi}^*} \sum_{j \in \mathbf{J}} \sum_{x' \in \mathbf{X}} \sum_{\pi' \in \mathbf{\Pi}^*} N^{val}(x, \pi, x', \pi', j; \mathbf{\Pi}^*) \log \frac{N^{trn}(x, \pi, x', \pi', j; \mathbf{\Pi}^*)}{N^{trn}(x, \pi; \mathbf{\Pi}^*) N^{trn}(x', \pi', j; \mathbf{\Pi}^*)} \right)$$

where $\mathbf{\Pi}^* = \text{argmax}_{\mathbf{\Pi}} \mathcal{F}(\mathbf{\Pi}; \eta)$ and $N^{trn}(.)$ and $N^{val}(.)$ are counting functions within the training and validation data, respectively. Notice that the main differences between Equations (8) and (12) is that here the model is *learned* on the training data, but evaluated on the validation data. As such, the weight terms ($N(\cdot)$) and λ_{adj} are calculated on the validation dataset, while the probability terms are based on the training set. In addition, one minor difference is that this score is normalized by $\frac{1}{1+\lambda_{rel}}$. Without this normalization, any hyperparameter optimization procedure will tend to select larger λ_{rel} since a larger λ_{rel} leads to a higher score.

In the validation procedure, we select the hyperparameter values that have the highest $score$ on the validation dataset. Note that having a separate validation set is not necessary. We can also use other hyper-parameter optimization techniques such as cross-validation (and this option is available in the accompanying software package).

4.3.3 Zero Probability Outcomes in Training Data

A final implementation issue that we need to address is the presence of choices and state transitions in the validation set that have not been seen in the training set.[5] For example, suppose that we have no observations where the agent chooses action j in state $\{x, \pi\}$ in the training data, but there exists such an observation in the validation data. Then, the score in Equation (12) becomes negative infinity. This problem makes the hyperparameter optimization very sensitive to outliers.

To solve this problem, we turn to the Natural Language Processing (NLP) literature, which also deals with high-dimensional data and has studied this problem extensively. Several smoothing techniques have been proposed to resolve this issue in NLP models (Chen and Goodman, 1999). One of the simplest smoothing methods that is used in practice and can be applied to our setting is additive smoothing (Johnson, 1932), which assumes that we have seen all possible observations (i.e., choices and state-transitions) at least δ times, where usually $0 \leq \delta \leq 1$. This smoothing technique

[5]These kind of zero-probability occurrences are common in a finite sample setting as the dimensionality of the data increases.

removes the possibility of zero probabilities

A typical value for δ in the NLP literature is 1; however, the optimal value for δ depends on the data-generating process. A low value for δ penalizes the model more heavily for potential outliers. It also prevents the recursive partitioning step from creating small partitions, since it increases the probability of having observations in the validation set that are not observed in the training set. On the other hand, a high value for δ may help with reducing overfitting by adding more noise to the data. In our simulations, we set δ to 10^{-5}, which is a relatively small value. However, the researcher can use a different number depending on their data and application setting.

4.4 Summary of Estimation Approach

In summary, the estimation approach consists of two steps. In the first step, we employ the recursive partitioning algorithm described in §4.1–§4.3 to reduce the dimensionality of the high-dimensional state variables $\mathbf{Q}$ to lower-dimensional $\mathcal{P}$-space. In the second step, we use $\mathbf{X}$ and the estimated $\hat{\mathcal{P}}$ as the set of observed state variables and estimate the dynamic discrete choice model. Note that at this stage, we can use any estimation procedure (including NXFP and two-step methods), and all the properties of these estimators are valid conditional of $\mathbf{X}$ and $\hat{\mathcal{P}}$.

5 Monte Carlo Experiments

We now present two simulation studies to illustrate the performance of our algorithm. We use the canonical bus engine replacement problem introduced by Rust (1987) as the setting for our experiments. Rust's framework has been widely used as a benchmark to compare the performance of newly proposed estimators/algorithms for single-agent dynamic discrete choice models (Hotz et al., 1994; Aguirregabiria and Mira, 2002; Arcidiacono and Miller, 2011). We start by describing the original bus engine replacement problem and our high-dimensional extension of it in §5.1. In our first set of experiments in §5.2, we document the extent to which our algorithm is able to recover parameters of interest and compare its performance to a benchmark case where the high-dimensional state variables are ignored. In the second set of experiments in §5.3, we examine the importance of optimizing the key hyperparameter, λ_{rel}.

5.1 Engine replacement problem

In Rust's model, a single agent (Harold Zurcher) chooses whether to replace the engine of a bus or continue maintaining it in each period. The maintenance cost is linearly increasing in the mileage of the engine, while replacement constitutes a one-time lump-some cost. The intertemporal trade-off is as follows – by replacing the bus engine, he pays a high replacement cost today and has lower maintenance costs in the future. If he chooses to not replace the bus engine, he avoids the

replacement cost but will continue to pay a higher maintenance cost in the future.

We start with the basic version of the model without the high-dimensional state variables. Here, the per-period utility function of two choices is given by:

$$\begin{aligned} u(x_{it}, d_{it} = 0) &= -c_m x_{it} + \epsilon_{i0t} \\ u(x_{it}, d_{it} = 1) &= -c_r + \epsilon_{i1t}, \end{aligned} \tag{13}$$

where x_{it} is the mileage of bus i at time t, c_m is the per-mile maintenance cost, c_r is the cost of replacing the engine, and $\{\epsilon_{i0t}, \epsilon_{i1t}\}$ are the error terms associated with the two choices. Next, we assume that the mileage increases by one unit in each period [6]. Formally:

$$\begin{aligned} &\text{if } \; d_{it} = 0, \;\; \text{then } \; x_{it+1} = x_{it} + 1 \;\; \text{if } \; x_{it} < 20, \;\; \text{else } \; x_{it+1} = x_{it} \\ &\text{if } \; d_{it} = 1, \;\; \text{then } \; x_{it+1} = 1 \end{aligned} \tag{14}$$

The maximum mileage is capped at 20, i.e., after the mileage hits 19, it continues to stay there.

We now extend the problem to incorporate a set of high-dimensional state variables $\mathbf{Q}$ that can affect the utility function and state transition. $\mathbf{Q}$ can include all the potential variables that can affect utilities and state transitions. For example, the mileage accrued may vary depending on the bus route, weather of the day, etc. Similarly, the replacement costs may vary by bus brands and/or economic conditions. A priori, it can be hard to identify which of these state variables and their combinations matter. Our method allows us to include all potential variables in the utility and state transition, and identify the partitions that matter.

In our simulations, we expand the utility function and state transition to include $\mathbf{Q}$ as follows:

$$\begin{aligned} u(x_{it}, \mathbf{\Pi}^*(q_{it}), d_{it} = 0) &= c_m x_{it} + \epsilon_{i0t} \\ u(x_{it}, \mathbf{\Pi}^*(q_{it}), d_{it} = 1) &= f_{dc}(\mathbf{\Pi}^*(q_{it})) + \epsilon_{i1t} \end{aligned} \tag{15}$$

$$\begin{aligned} &\text{if } \; d_{it} = 0, \;\; \text{then } \; x_{it+1} = x_{it} + f_{tr}(\mathbf{\Pi}^*(q_{it})) \;\; \text{if } \; x_{it} < 20, \;\; \text{else } \; x_{it+1} = x_{it} \\ &\text{if } \; d_{it} = 1, \;\; \text{then } \; x_{it+1} = f_{tr}(\mathbf{\Pi}^*(q_{it})) \end{aligned} \tag{16}$$

where q_{it} is the high-dimensional state variable for bus i at time t, and $f_{dc}(\mathbf{\Pi}^*(q_{it}))$ and $f_{tr}(\mathbf{\Pi}^*(q_{it}))$ are functions that specify the effect of q_{it} on the replacement cost and state transition, respectively. Note that we use $\mathbf{\Pi}^*(q_{it})$ instead of q_{it} since $\mathbf{\Pi}^*(q_{it})$ is a perfect discretization (and conveys the same information) as q_{it}. Finally, we set $c_m = -0.2$ in both our simulation studies.[7]

[6] This choice is just for the sake of increasing simplicity. However, our framework can easily handle stochastic state transitions as well.

[7] Note that the specific functional form is chosen for convenience, and the algorithm works even with fully non-parametric

5.2 First simulation study

In the first set of experiments, our goal is to demonstrate the algorithm's performance and document the extent of bias when we do not account for $\mathbf{Q}$.

5.2.1 Data generating process

In this simulation study, $\mathbf{Q}$ consists 10 variable: $q^1, \ldots, q^{10}$, where $q^i \in \{0, 1, \ldots, 9\}$. However, only the first two variables affect the data-generating process, and the rest of them are irrelevant. In principle, our algorithm is robust to the inclusion of irrelevant state variables. Therefore, the extra eight variables in $\mathbf{Q}$ allow us to examine if this is indeed the case in practice. In our simulations, q^1 and q^2 partition $\mathbf{Q}$ into four regions $\{\pi_1, \pi_2, \pi_3, \pi_4\}$ such that all the observations within a partition have the same choice and state transition probabilities. The partitions are given by:

$$\mathbf{\Pi}^*(q) = \begin{cases} \pi_1, & \text{if } q^1 < 5 \text{ and } q^2 < 5 \\ \pi_2, & \text{if } q^1 < 5 \text{ and } q^2 \geq 5 \\ \pi_3, & \text{if } q^1 \geq 5 \text{ and } q^2 < 5 \\ \pi_4, & \text{if } q^1 \geq 5 \text{ and } q^2 \geq 5 \end{cases}$$

The mileage transitions are as described in Equation (16). We also need to define the state transition in the $\mathbf{Q}$ space. Note that only the state transitions between πs matter, i.e., conditional on the partition π, the exact q is not informative of utilities or state transitions random. We consider three different state transition models in our simulations:

- No transition: The agent remains within the same partition in each period: $\mathbf{\Pi}^*(q_{it}) = \mathbf{\Pi}^*(q_{it+1})$.
- Random transition: Agents' transitions in the $\mathbf{Q}$-space are completely random. In each period, an agent randomly transitions from one partition to another such that: $\mathbf{\Pi}^*(q_{it}) \perp \mathbf{\Pi}^*(q_{it+1})$.
- Sparse transition: After each period, the agent remains in the same partition with probability 0.5 and moves to the next partition with probability 0.5. We choose the order of partitions as follows: π_2 is after π_1, π_3 is after π_2, π_4 is after π_3, and π_1 is after π_4.

Next, we consider two different options for f_{tr} and f_{dc} in Equations (15) and (16) each, as follows:

$$f_{tr}([\pi_1, \pi_2, \pi_3, \pi_4]) = \begin{cases} [0, 1, 2, 3] \text{ in the dissimilar mileage transition case} \\ [1, 1, 1, 1] \text{ in the similar mileage transition case} \end{cases} \tag{17}$$

utilities within a partition and non-parametric state transitions across partitions.

$$f_{dc}([\pi_1, \pi_2, \pi_3, \pi_4]) = \begin{cases} [-7, -6, -5, -4] \text{ in the dissimilar replacement costs case} \\ [-5, -5, -5, -5] \text{ in the similar replacement costs case} \end{cases} \tag{18}$$

Together, this gives us $3 \times 2 \times 2 = 12$ possible scenarios for the data-generating process. For example, one possible data-generating process could be {No transition, Similar mileage transition, Dissimilar replacement cost}, where $f_{tr}([\pi_1, \pi_2, \pi_3, \pi_4]) = [1, 1, 1, 1]$ and $f_{dc}([\pi_1, \pi_2, \pi_3, \pi_4]) = [-7, -6, -5, -4]$. In this case, $\mathbf{Q}$ does not affect mileage transitions but only the flow utilities. On the other hand, if the data-generating process is {Random transition, Dissimilar mileage transition, Similar replacement cost}, then $\mathbf{Q}$ affects state transitions but not flow utilities. By considering all such possible scenarios, we are able to explore how the algorithm performs as the data-generating process changes.

We now simulate data and recover the structural parameters for each of these cases using the approach outlined in §4.4. In the first step, we use our recursive partitioning procedure to generate the partitioning (as described in §4). While doing so, we set the hyperparameters as follows: *minimum lift* $= 10^{-10}$, *minimum number of observations* $= 1$, and $\lambda_{rel} = 1$. Further, for each case, we run the algorithm (and then perform the estimation) for four different values of *maximum partitions*: $1, 2, 4,$ or 6. The single partition case is equivalent to ignoring the high-dimensional state variable $\mathbf{Q}$. Setting *maximum partitions* to 2 and 6 allows us to examine how our algorithm performs when we allow for under-discretization and over-discretization, respectively. Then, once we have a partition, we use the Nested Fixed Point algorithm by Rust (1987) to estimate the structural parameters for each case.

5.2.2 Results

For each of the 12 data generation processes, we run 100 simulations. In each simulation, we generate data for 400 buses for 100 time periods. For each simulated dataset, we first use our recursive partitioning algorithm to discretize the high-dimensional state space $\mathbf{Q}$ and obtain $\hat{\mathcal{P}}$. While doing so, we consider four different versions of the recursive partitioning algorithm, where each version allows for a different value of *the number of partitions* – $\{1, 2, 4, 6\}$. Then, as discussed in §4.4, we simply treat the set of estimated partitions in $\hat{\mathbf{\Pi}}^*(q)$, i.e., $\hat{\mathcal{P}}$ as an extra categorical variable in addition to $\mathbf{X}$ at the estimation stage, and estimate the structural utility parameters. The parameter estimates of c_m from these simulations are presented in Table 1. Next, in Table 2, we focus only on two versions of the recursive partitioning model – (1) where we allow for four partitions of $\mathbf{Q}$, and (2) one where we neglect $\mathbf{Q}$, which is similar to treating all the data as being in one partition, and present the replacement cost estimates.

As we can see from both tables, neglecting $\mathbf{Q}$ often leads to biased estimates; see column 5

Case No.	Effect on Replacement Cost	Effect on Mileage Transition	Transition in $\Pi^*(\mathbf{Q})$ Sapce	Number of Allowed Partitions			
				1	**2**	**4**	**6**
Case 1	Dissimilar	Dissimilar	No transition	-0.135 (-0.144, -0.127)	-0.173 (-0.18, -0.167)	-0.194 (-0.204, -0.187)	-0.193 (-0.204, -0.186)
Case 2	Dissimilar	Dissimilar	Random transition	-0.149 (-0.154, -0.142)	-0.186 (-0.191, -0.18)	-0.2 (-0.206, -0.193)	-0.199 (-0.206, -0.192)
Case 3	Dissimilar	Dissimilar	Sparse transition	-0.123 (-0.129, -0.118)	-0.182 (-0.198, -0.167)	-0.199 (-0.207, -0.191)	-0.199 (-0.207, -0.191)
Case 4	Dissimilar	Similar	No transition	-0.165 (-0.172, -0.158)	-0.191 (-0.2, -0.183)	-0.199 (-0.207, -0.19)	-0.198 (-0.207, -0.19)
Case 5	Dissimilar	Similar	Random transition	-0.171 (-0.177, -0.162)	-0.193 (-0.201, -0.184)	-0.2 (-0.209, -0.192)	-0.199 (-0.209, -0.192)
Case 6	Dissimilar	Similar	Sparse transition	-0.17 (-0.176, -0.164)	-0.096 (-0.119, -0.067)	-0.201 (-0.211, -0.189)	-0.201 (-0.211, -0.188)
Case 7	Similar	Dissimilar	No transition	-0.178 (-0.188, -0.17)	-0.193 (-0.201, -0.187)	-0.199 (-0.208, -0.192)	-0.199 (-0.208, -0.192)
Case 8	Similar	Dissimilar	Random transition	-0.185 (-0.194, -0.179)	-0.197 (-0.205, -0.189)	-0.2 (-0.208, -0.192)	-0.2 (-0.207, -0.191)
Case 9	Similar	Dissimilar	Sparse transition	-0.174 (-0.182, -0.168)	-0.236 (-0.247, -0.224)	-0.2 (-0.209, -0.193)	-0.2 (-0.209, -0.193)
Case 10	Similar	Similar	No transition	-0.2 (-0.205, -0.192)	-0.199 (-0.205, -0.192)	-0.199 (-0.204, -0.189)	-0.198 (-0.204, -0.188)
Case 11	Similar	Similar	Random transition	-0.2 (-0.207, -0.193)	-0.199 (-0.207, -0.193)	-0.199 (-0.207, -0.192)	-0.198 (-0.206, -0.191)
Case 12	Similar	Similar	Sparse transition	-0.199 (-0.209, -0.192)	-0.199 (-0.208, -0.192)	-0.198 (-0.208, -0.19)	-0.198 (-0.207, -0.19)

Table 1: The estimated mileage maintenance cost, c_m, and the 96% bootstrap confidence interval around the mean in each of the 12 Monte Carlo simulations. Each cell is a result of 100 rounds of simulation for 4 buses. Note that the case where the number of partitions is set to 1 is equivalent to completely neglecting $\mathbf{Q}$.

Case No.	Effect on Replacement Cost	Effect on Mileage Transition	Transition in $\Pi^*(\mathbf{Q})$ Sapce	Incorporating discretized $\mathbf{Q}$				Neglecting $\mathbf{Q}$
				$f_{dc}(\pi_1)$	$f_{dc}(\pi_2)$	$f_{dc}(\pi_3)$	$f_{dc}(\pi_4)$	c_r
Case 1	Dissimilar	Dissimilar	No transition	-6.812	-5.845	-4.843	-4.435	-5.408
				(-7.115, -6.589)	(-6.121, -5.635)	(-5.084, -4.642)	(-4.724, -3.908)	(-5.76, -5.148)
Case 2	Dissimilar	Dissimilar	Random transition	-7.006	-6.008	-4.998	-3.995	-4.962
				(-7.179, -6.834)	(-6.132, -5.853)	(-5.165, -4.831)	(-4.13, -3.852)	(-5.099, -4.846)
Case 3	Dissimilar	Dissimilar	Sparse transition	-7.015	-5.987	-4.984	-3.984	-4.886
				(-7.352, -6.773)	(-6.201, -5.804)	(-5.139, -4.846)	(-4.222, -3.773)	(-5.046, -4.766)
Case 4	Dissimilar	Similar	No transition	-6.975	-5.964	-4.977	-4.0	-4.701
				(-7.252, -6.69)	(-6.162, -5.74)	(-5.142, -4.82)	(-4.13, -3.86)	(-4.906, -4.548)
Case 5	Dissimilar	Similar	Random transition	-7.025	-5.996	-4.997	-3.995	-4.661
				(-7.327, -6.775)	(-6.211, -5.83)	(-5.183, -4.829)	(-4.165, -3.842)	(-4.769, -4.507)
Case 6	Dissimilar	Similar	Sparse transition	-7.061	-6.028	-5.021	-4.012	-4.731
				(-7.394, -6.769)	(-6.253, -5.757)	(-5.244, -4.813)	(-4.221, -3.803)	(-4.858, -4.618)
Case 7	Similar	Dissimilar	No transition	-5.064	-5.019	-4.977	-4.925	-5.323
				(-5.218, -4.901)	(-5.163, -4.857)	(-5.138, -4.811)	(-5.083, -4.732)	(-5.524, -5.115)
Case 8	Similar	Dissimilar	Random transition	-5.041	-5.01	-4.986	-4.957	-5.039
				(-5.191, -4.905)	(-5.151, -4.883)	(-5.14, -4.856)	(-5.115, -4.84)	(-5.221, -4.9)
Case 9	Similar	Dissimilar	Sparse transition	-5.059	-5.021	-4.994	-4.956	-5.045
				(-5.237, -4.905)	(-5.196, -4.854)	(-5.149, -4.847)	(-5.112, -4.812)	(-5.219, -4.864)
Case 10	Similar	Similar	No transition	-5.043	-4.998	-4.963	-4.911	-4.994
				(-5.196, -4.886)	(-5.131, -4.861)	(-5.086, -4.823)	(-5.039, -4.779)	(-5.11, -4.869)
Case 11	Similar	Similar	Random transition	-5.033	-4.994	-4.975	-4.936	-5.0
				(-5.25, -4.874)	(-5.162, -4.851)	(-5.12, -4.831)	(-5.099, -4.772)	(-5.143, -4.863)
Case 12	Similar	Similar	Sparse transition	-5.005	-4.982	-4.964	-4.939	-4.989
				(-5.166, -4.842)	(-5.118, -4.817)	(-5.102, -4.813)	(-5.094, -4.808)	(-5.129, -4.827)

Table 2: The estimated replacement cost and their 96% bootstrap confidence interval around the mean in each of the 12 Monte Carlo simulations. Each row is a result of 100 rounds of simulation.

in Table 1 and the last column in Table 2. This is true even when q_{it} does not directly affect the flow utility and there is no serial correlation in q over time, e.g., Case 8 in Table 1. In this case, the bias arises because $\mathbf{Q}$ affects the state transitions through f_{tr} (which in turn affects the expected future value function). Thus, when we neglect $\mathbf{Q}$ in the estimation procedure, it is captured in the error term, which in turn violates the conditional independence assumption that ϵ_{it+1} and x_{it+1} are independent of ϵ_{it}. Further, the bias seems to be worse when $\mathbf{Q}$ affects both flow utility and state transition, especially when $\mathbf{Q}$ itself has state dependence in how it transitions into the next period (e.g., case 3). Also, as expected, in cases where $\mathbf{Q}$ does not affect the utility function or the state transition (cases 10, 11, and 12 in both tables), neglecting $\mathbf{Q}$ does not bias the parameter estimates. This is because $\mathbf{Q}$ is an irrelevant state variable in these cases.

Finally, columns 6 and 8 in Table 1 highlight the impact of under- and over-discretization. As expected, over-discretization does not introduce any bias in the estimation procedure, since it does not violate any estimation assumption. However, it does lead to marginally higher variance (compared to column 7) since we estimate more parameters when we over-discretize. However, under-discretization does lead to bias; though, by and large, the bias seems to be lower than completely ignoring $\mathbf{Q}$.

In sum, these simulations show that our recursive partitioning approach can help reduce the dimensionality of the dynamic discrete choice problem and help with correcting for bias and recovering true primitive parameters.

5.3 Second Simulation Study

We now conduct a series of numerical experiments to examine the importance of hyperparameter tuning. Specifically, we focus on the choice of λ_{rel}, which is unique to our dynamic setting and captures the relative importance of the choice probabilities and state transitions in designing the partition. Recall that when λ_{rel} is large, the recursive partitioning algorithm puts a higher weight on state transitions (compared to choice probabilities) and vice versa. Tuning this hyperparameter should therefore allow us to weight the state transition data less when state transitions are noisy and/or less informative compared to choice probabilities (since higher λ_{rel} in such cases would lead the algorithm to pick up noise as a signal in its discretization).

In general, we expect the choice of λ_{rel} to be important when: (1) the dimensionality of $\mathbf{Q}$ is high since it is much easier for the algorithm to find noisy patterns in such cases (because there are a lot more variables for a potential split), and (2) when the number of observations is relatively small, i.e., finite samples, because there may not be enough data in each possible partition to pick up patterns accurately. In the rest of this section, we therefore explore the sensitivity of the partitioning procedure to the choice of λ_{rel} for different data-generating processes and data sizes.

5.3.1 Data generating process

The data-generating process in these simulations is similar to those in the previous study, with small differences. We give a brief summary below and present details in Appendices §B and §B.

1. First, the dimensionality of $\mathbf{Q}$ is 30, but only the first 10 variables affect the data-generating process, and the rest are irrelevant. We randomly discretize $\mathbf{Q}$ into 15 partitions using the process explained in Web Appendix§B. We use this process to generate 100 different discretizations, i.e., 100 cases each with a different discretization of $\mathbf{Q}$.

2. We then allow the replacement cost in each partition in each case to be a function of the partition in $\mathbf{\Pi}^*$ as shown in Web Appendix§B.

Next, for each of the 100 discretizations, we vary the mileage transition model, the state transition model in $\mathbf{Q}$, and the amount of data available in two ways each, which gives us eight different data-generating scenarios for each of the discretizations. These are discussed below.

3. We consider two cases for f_{tr}: (i) $f_{tr}(\pi) = 1$ for all 15 partitions (Similar mileage transition case), and (ii) $f_{tr}(\pi)$ is a random number from the set $\{0, 1, 2, 3\}$ (Dissimilar mileage transition case).

4. We consider two types of state transition models for $\mathbf{Q}$:

   - Random transition: Agents' transitions in the $\mathbf{Q}$-space are completely random. In each period, an agent randomly transitions from one partition to another such that: $\mathbf{\Pi}^*(q_{it}) \perp \mathbf{\Pi}^*(q_{it+1})$.
   - Sparse transition: After each period, the agent remains in the same partition with probability $1/3$ and moves to one of the next two partitions with the same probability, and the ordinality/ordering of partitions is randomly chosen in each simulation.

   Variation across these two dimensions allows us to vary the amount of information available in the state transition. For example, there is less information in the state transition in the case where the transitions across partitions in $\mathbf{\Pi}^*$ random and $f_{tr}(\pi) = 1$.

5. We also vary the amount of data available for the analysis by considering two scenarios for the number of observations: (i) 100 buses in 100 periods and (ii) 100 buses in 400 periods.

For all these scenarios, we examine the impact of different values of λ_{rel} by learning the partition from the training data and evaluating it on a validation dataset (that is separately generated and is the same size as the training data) using the score function (as shown in Equation (12)). Specifically,

Case No.	Number of Periods	Transition in $\mathbf{\Pi}^*(Q)$ space	$\mathbf{\Pi}^*(Q)$ affects mileage transition	Value of λ_{rel}						
				0	0.2	0.5	1	2	5	100
Case 1	100	Sparse	Yes	-3722	-3481	-3274	-3121	-2977	-2864	**-2766**
Case 2		Sparse	No	-3762	-3536	-3407	-3323	-3245	**-3200**	-3228
Case 3		Random	Yes	-3743	-3762	**-3730**	-3768	-3831	-3913	-4062
Case 4		Random	No	**-3698**	-3781	-3867	-3959	-4063	-4192	-4373
Case 5	400	Sparse	Yes	-13287	-12969	-12606	-12239	-11864	-11519	**-11217**
Case 6		Sparse	No	-13815	-13650	-13469	-13264	-13059	**-12909**	-12987
Case 7		Random	Yes	**-13518**	-13620	-13693	-13770	-13863	-13974	-14187
Case 8		Random	No	**-13613**	-13921	-14231	-14542	-14856	-15177	-15556

Table 3: The calculated score for different values of λ_{rel} in different data-generating processes. A bigger λ_{rel} is better when there is more information in the state transition data. Scores are averaged over the 100 different partitioning schemes and shown for the validation data.

we consider the following values of $\lambda_{rel} \in \{0, 0.2, 0.5, 1, 2, 5, 100\}$. During this analysis, we fix all the other hyper-parameters as follows: *minimum number of observations* = 1, *minimum lift* = 10^{-10}, and *total number of partitions* = 15 (which is the number of true partitions in the data). Fixing all the other hyper-parameters allows us to focus on the role of λ_{rel}.

5.3.2 Results

In total, we run 100 (partitions) $\times 8$ (scenarios)$\times 7$ (values for λ_{rel}) $= 5600$ simulations and report the score on the validation data in Table 3. The bold numbers in each row denote the best average score (on the validation data) across all the 100 partitions for a given scenario. We find that the optimal value of λ_{rel} varies across data-generating processes. As expected, the optimal value for λ_{rel} is the largest when the state transition is most informative (i.e. cases 1 and 5, where the transition in $\mathbf{\Pi}^*(\mathbf{Q})$ is Sparse (as opposed to Random), and the partition affects the mileage transition). Similarly, the optimal value λ_{rel} is small (or zero) in cases 4 and 8, where the transition in $\mathbf{\Pi}^*(\mathbf{Q})$ is Random and $\mathbf{\Pi}^*(\mathbf{Q})$ does not affect mileage transition (thus not informative for finding the discretization).

In summary, these simulations demonstrate that using the state transition information in the recursive partitioning algorithm helps improve the quality of the partitions, though the extent to which this data helps varies with the data-generating process (and depends on the level of signal available in the state transitions). Thus, it is important to tune λ_{rel} in a data-driven fashion to learn how much to weigh state transition data (relative to choice data).

Please note that we provide two GitHub repos for the methods in the paper: (1) https://github.com/ebzgr/RePaD – consists of a set of codes/package for implementing the approach on other datasets, and (2) https://github.com/ebzgr/RePaD-Paper – contains all the codes for the simulation results from this

section.

6 Extensions, Limitations, and Conclusion

The ideas presented in the paper can be generalized and applied to other settings. We now discuss two natural ways to improve the performance of the method and extend it to other cases.

Cross-fitting: The idea of cross-fitting has been used in Double Machine Learning approaches to avoid overfitting bias (Chernozhukov et al., 2018, 2022). Essentially, cross-fitting requires the researcher to fit the first-stage models on one dataset and then perform the estimation on a different dataset. In our setting, this can be done efficiently by splitting the data into two parts and using split-1 to learn the partitioning and performing the estimation on split-2 (taking the partition from split-1 as given), and vice versa. This separates the process of partitioning and estimation further, thereby reducing potential bias from overfitting/regularization during the first-stage partitioning.

Dynamic Games: The method can be easily extended to dynamic games, e.g., entry models, oligopolistic models of competition and investment (Bajari et al., 2007; Aguirregabiria et al., 2021). In this case, in the recursive-partitioning algorithm, the state transition probabilities of an entity/agent would be conditioned on both their own actions as well as the actions of the other agents in the system. Then, once we have the partitioning, we can use a standard two-step estimator with the estimated partitions as another state variable just as we did in the case of the single agent model (with the appropriate assumptions on the uniqueness of the equilibrium observed in the data).

Limitations: Nevertheless, there are a series of limitations with our method, many of which are shared by other tree-based approaches. First, if $\mathbf{Q}$ enters the state transitions and decisions in a continuous fashion (instead of as partitions/categories), then no empirical partitioning will ever be perfect. This would be similar to using CART to capture a regression model where the explanatory variables enter regression in a continuous fashion, i.e., we can keep partitioning the data further and further without reaching a perfect discretization. In practice, this is not a big issue, since the regularization procedure will stop the partitioning when the gains from partitioning are not sufficiently high to continue. However, in theory, there is likely to be some bias from the fact that we have not correctly accounted for the true partitions and have under-discretized $\mathbf{Q}$. Second, even though CART and other recursive partitioning methods are extremely popular for prediction and classification tasks and have many good properties, they lack some standard consistency properties. In particular, it is well-known that some data patterns cannot be captured by recursive partitioning even if the number of observations goes to infinity; see Web Appendix§A and Biau et al. (2008) for details. Thus, there are no theoretical guarantees that this procedure will reach the true underlying partition for certain types of data-generating processes. Finally, from a practical perspective, we might not have enough observations to fully recover the accurate discretization

when the dimensionality of $\mathbf{Q}$ is high, especially when we have a limited number of observations.

In practice, we can take a few steps to ensure that the partitioning algorithm is producing stable and reasonable, and not unduly affected by these limitations. First, it is important to use hyper-parameter optimization to ensure that the discretization generated by our algorithm is generalizable to the data-generating process, not just the training sample. We can check the performance of estimated parameters in both training and validation sets and use the extent of the difference as a measure of the quality of our partitioning. Another suggestion is to check the sensitivity of the discretization and estimated parameters to the number of observations. For example, we can discretize $\mathbf{Q}$ and estimate the model with 80% of the data, and check how close the discretization and estimated parameters are to the case where we use all the data. This could be a simple heuristic to test the robustness of the results.

In sum, our proposed algorithm allows researchers to reduce the estimation bias associated with ignoring relevant state variables, improve the fit of their models, and draw post-hoc inferences in a high-dimensional setting without theoretical guidance, all at relatively low compute costs. Further, because the recursive partitioning approach is modular, it does not require the user to develop or learn new estimation procedures. Indeed, it can be easily combined with any estimation procedure (e.g., nested fixed point, two-step) as a first-step approach. As such, we expect it to be easily applicable to a wide variety of settings.

Competing Interests Declaration

Author(s) have no competing interests to declare.

References

V. Aguirregabiria. *Empirical industrial organization: models, methods, and applications*. 2021.

V. Aguirregabiria and P. Mira. Swapping the nested fixed point algorithm: A class of estimators for discrete markov decision models. *Econometrica*, 70(4):1519–1543, 2002.

V. Aguirregabiria and P. Mira. Dynamic discrete choice structural models: A survey. *Journal of Econometrics*, 156(1):38–67, 2010.

V. Aguirregabiria, A. Collard-Wexler, and S. P. Ryan. Dynamic games in empirical industrial organization. In *Handbook of Industrial Organization*, volume 4, pages 225–343. Elsevier, 2021.

P. Arcidiacono and P. B. Ellickson. Practical methods for estimation of dynamic discrete choice models. *Annu. Rev. Econ.*, 3(1):363–394, 2011.

P. Arcidiacono and R. A. Miller. Conditional choice probability estimation of dynamic discrete choice models with unobserved heterogeneity. *Econometrica*, 79(6):1823–1867, 2011.

P. Arcidiacono, P. Bayer, F. A. Bugni, and J. James. Approximating high-dimensional dynamic

models: Sieve value function iteration. Technical report, National Bureau of Economic Research, 2012.

S. Athey and G. Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016.

S. Athey, J. Tibshirani, S. Wager, et al. Generalized random forests. *The Annals of Statistics*, 47(2): 1148–1178, 2019.

P. Bajari, C. L. Benkard, and J. Levin. Estimating dynamic models of imperfect competition. *Econometrica*, 75(5):1331–1370, 2007.

H. Benitez-Silva, G. Hall, G. J. Hitsch, G. Pauletto, and J. Rust. A comparison of discrete and parametric approximation methods for continuous-state dynamic programming problems. *manuscript, Yale University*, 2000.

G. Biau, L. Devroye, and G. Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(Sep):2015–2033, 2008.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

V. Chernozhukov, D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins. Double/debiased machine learning for treatment and structural parameters: Double/debiased machine learning. *The Econometrics Journal*, 21(1), 2018.

V. Chernozhukov, J. C. Escanciano, H. Ichimura, W. K. Newey, and J. M. Robins. Locally robust semiparametric estimation. *Econometrica*, 90(4):1501–1535, 2022.

T. M. Cover and J. A. Thomas. Entropy, relative entropy and mutual information. *Elements of information theory*, 2:1–55, 1991.

P. B. Ellickson, S. Misra, and H. S. Nair. Repositioning dynamics and pricing strategy. *Journal of Marketing Research*, 49(6):750–772, 2012.

T. Erdem, S. Imai, and M. P. Keane. Brand and quantity choice dynamics under price uncertainty. *Quantitative Marketing and economics*, 1:5–64, 2003.

S. Esteban and M. Shum. Durable-goods oligopoly with secondary markets: the case of automobiles. *The RAND Journal of Economics*, 38(2):332–354, 2007.

G. Gowrisankaran and M. Rysman. Dynamics of consumer demand for new durable goods. *Journal of political Economy*, 120(6):1173–1219, 2012.

W. R. Hartmann. Intertemporal effects of consumption and their implications for demand elasticity estimates. *Quantitative Marketing and Economics*, 4:325–349, 2006.

T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learnin. *Cited on*, page 33, 2009.

I. Hendel and A. Nevo. Measuring the implications of sales and consumer inventory behavior. *Econometrica*, 74(6):1637–1673, 2006.

G. J. Hitsch. An empirical model of optimal dynamic product launch and exit under demand uncertainty. *Marketing Science*, 25(1):25–50, 2006.

V. J. Hotz and R. A. Miller. Conditional choice probabilities and the estimation of dynamic models. *The Review of Economic Studies*, 60(3):497–529, 1993.

V. J. Hotz, R. A. Miller, S. Sanders, and J. Smith. A simulation estimator for dynamic models of discrete choice. *The Review of Economic Studies*, 61(2):265–289, 1994.

Z. Z. Jiang, J. Li, and D. Zhang. A high-dimensional choice model for online retailing. *Jun and Zhang, Dennis, A High-Dimensional Choice Model for Online Retailing (September 6, 2020)*, 2020.

W. E. Johnson. Probability: The deductive and inductive problems. *Mind*, 41(164):409–423, 1932.

M. P. Keane and K. I. Wolpin. The career decisions of young men. *Journal of political Economy*, 105(3):473–522, 1997.

J. B. Kim, P. Albuquerque, and B. J. Bronnenberg. Online demand under limited consumer search. *Marketing science*, 29(6):1001–1023, 2010.

H. Li. Intertemporal price discrimination with complementary products: E-books and e-readers. *Management Science*, 65(6):2665–2694, 2019.

O. Melnikov. Demand for differentiated durable products: The case of the us computer printer market. *Economic Inquiry*, 51(2):1277–1298, 2013.

H. Nair. Intertemporal price discrimination with forward-looking consumers: Application to the us market for console video-games. *Quantitative Marketing and Economics*, 5:239–292, 2007.

A. Norets. Estimation of dynamic discrete choice models using artificial neural network approximations. *Econometric Reviews*, 31(1):84–106, 2012.

W. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics. Wiley, 2007. ISBN 9780470182956. URL `https://books.google.com/books?id=WWWDkd65TdYC`.

J. Rust. Optimal replacement of gmc bus engines: An empirical model of harold zurcher. *Econo-*

metrica: Journal of the Econometric Society, pages 999–1033, 1987.

J. Rust. Parametric policy iteration: An efficient algorithm for solving multidimensional dp problems? Technical report, Mimeo, Yale University, 2000.

H. M. Sani, C. Lei, and D. Neagu. Computational complexity analysis of decision tree algorithms. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 191–197. Springer, 2018.

A. Singh, Y. Liu, and H. Yoganarasimhan. Choice models and permutation invariance. *arXiv preprint arXiv:2307.07090*, 2023.

C. Smammut and G. I. Webb, editors. *Bellman Equation*, pages 97–97. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_71. URL `https://doi.org/10.1007/978-0-387-30164-8_71`.

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*, 1:371–407, 2003.

C.-L. Su and K. L. Judd. Constrained optimization approaches to estimation of structural models. *Econometrica*, 80(5):2213–2230, 2012.

E. Y. Wang. The impact of soda taxes on consumer welfare: implications of storability and taste heterogeneity. *The RAND Journal of Economics*, 46(2):409–441, 2015.

Y. Wei and Z. Jiang. Estimating parameters of structural models using neural networks. *USC Marshall School of Business Research Paper*, 2022.

H. Yoganarasimhan. The value of reputation in an online freelance marketplace. *Marketing Science*, 32(6):860–891, 2013.

Web Appendix

A Proof of Convergence

In this section, we provide proof that our proposed algorithm converges to a perfect discretization. We start with a brief explanation of the structure of the proof.

The convergence proof consists of two steps. First, in §A.1 we prove that the likelihood of observing data given partitioning $\mathbf{\Pi}_r$ (at iteration r) and the optimal parameter value $\theta_r^* = \text{argmax}_\theta \mathcal{L}(\theta, \mathbf{\Pi}_r)$ is increasing at each iteration of the algorithm. Therefore, each additional iteration does not reduce the likelihood of observing the data, i.e., $\mathcal{L}(\theta_r^*, \mathbf{\Pi}_r) \leq \mathcal{L}(\theta_{r+1}^*, \mathbf{\Pi}_{r+1})$. Then in §A.2, we prove that $\mathcal{L}(\theta_r^*, \mathbf{\Pi}_r) = \mathcal{L}(\theta_{r+1}^*, \mathbf{\Pi}_{r+1})$ if and only if $\mathbf{\Pi}_r$ is a perfect discretization, proving that the proposed algorithm stops if and only if it reaches a perfect discretization. Before proceeding, we provide some definitions, assumptions, and lemmas that are used in the proof.

Definition 2. We split the likelihood function into two parts: the decision part and the state transition part, denoted by $\mathcal{L}_{dc}(\theta, \mathbf{\Pi})$ and $\mathcal{L}_{st}(\mathbf{\Pi})$ respectively, such that $\mathcal{L}(\theta, \mathbf{\Pi}) = \mathcal{L}_{dc}(\theta, \mathbf{\Pi}) + \lambda \mathcal{L}_{st}(\mathbf{\Pi})$. Note that the decision part of the likelihood is a function of the utility function parameters, θ.

Definition 3. Discretization $\mathbf{\Pi}$ is a **parent** of discretization $\mathbf{\Pi}'$ if for every $\pi' \in \mathbf{\Pi}'$, there is a partition $\pi \in \mathbf{\Pi}$ such that π' is completely within π. In other words, discretization $\mathbf{\Pi}$ is a *parent* of discretization $\mathbf{\Pi}'$ if one can generate discretization $\mathbf{\Pi}'$ by further splitting discretization $\mathbf{\Pi}$.

Definition 4. The expected value function and expected choice-specific value function are defined as follows:

$$\bar{V}(x, \pi; \theta, \mathbf{\Pi}) = \log \sum_{j \in \mathbf{J}} \exp v(x, \pi, j; \theta, \mathbf{\Pi}) \tag{A1}$$

$$v(x, \pi, j; \theta, \mathbf{\Pi}) = \bar{u}(x, \pi, j; \theta, \mathbf{\Pi}) + \beta \sum_{x' \in \mathbf{X}} \sum_{\pi' \in \mathbf{\Pi}} \bar{V}(x', \pi'; \theta, \mathbf{\Pi}) g(x', \pi' | x, \pi, j; \theta, \mathbf{\Pi}). \tag{A2}$$

where β is the discount factor usually set by the researcher. Assuming that error terms are drawn from Type 1 Extreme Value distribution, the predicted choice probabilities can be calculated as:

$$\hat{p}(j | x, \pi; \theta, \mathbf{\Pi}) = \frac{\exp v(x, \pi, j; \theta, \mathbf{\Pi})}{\exp \bar{V}(x, \pi; \theta, \mathbf{\Pi})}. \tag{A3}$$

Assumption 1. We assume a fully non-parametric form for the flow utility and state transition function to avoid dependence on parametric functions for the proof and keep the analysis general,

i.e., $\bar{u}(x, \pi, d; \theta, \mathbf{\Pi})$ and $g(x, \pi | x', \pi', j; \mathbf{\Pi})$ are constant for each combination of states, $\{x, \pi\}$, and decision d.

Assumption 2. There is no data pattern that is not discoverable with a single split.

Assumption 2 is common to all algorithms that are based on recursive partitioning. It has been shown that some patterns cannot be captured by recursive partitioning, even when the number of observations goes to infinity (Biau et al., 2008). A simple example of such patterns is presented in Figure A1. Assume observations are uniformly distributed throughout the covariate space. Also, assume that the flow utility in the crosshatched regions is l and it is h in the non-crosshatched regions. Any split would result in two sub-partitions with a similar number of observations in the crosshatched and non-crosshatched regions. The average utility would be equal to $\frac{h+l}{2}$ in the resulting two sub-partitions. Since the split does not increase $\mathcal{L}_{dc}$, the algorithm does not add it to its discretization. Generally, recursive partitioning cannot capture variational patterns that are symmetric in a way that any splits would lead to two sub-partition with a similar average statistic.

To overcome this shortcoming of recursive partitioning algorithms, we assume no data pattern exists that a single split in a discretization cannot partially capture. Then because of the assumption we can draw the following corollary.

Corollary 1. A discretization $\mathbf{\Pi}$ is either perfect, or there exists a split such that the decision probabilities, average incoming transition probabilities, or outgoing transition probabilities on two sub-partitions are not equal. Formally, if discretization $\mathbf{\Pi}$ is not perfect, there exists a split that partitions $\pi_p \in \mathbf{\Pi}$ into π_l and π_r such that for a $\{x, x'\} \in \mathbf{X}$, $\pi' \in \mathbf{\Pi}$ and $j \in \mathbf{J}$ at least one of the following inequalities holds:

$$\begin{aligned}
\Pr(j|x, \pi_l) &\neq \Pr(j|x, \pi_r) \\
\Pr(x', \pi'|x, \pi_l, j) &\neq \Pr(x', \pi'|x, \pi_r, j) \\
\frac{\Pr(x, \pi_l|x', \pi', j)}{N(x, \pi_l)} &\neq \frac{\Pr(x, \pi_r|x', \pi', j)}{N(x, \pi_r)}
\end{aligned}$$

Finally, we need the following lemma for proving that the decision likelihood is increasing in each iteration in §A.1.

Lemma 1. For any given vector $a : \sum_i a_i = 1$, the solution to the following maximization problem

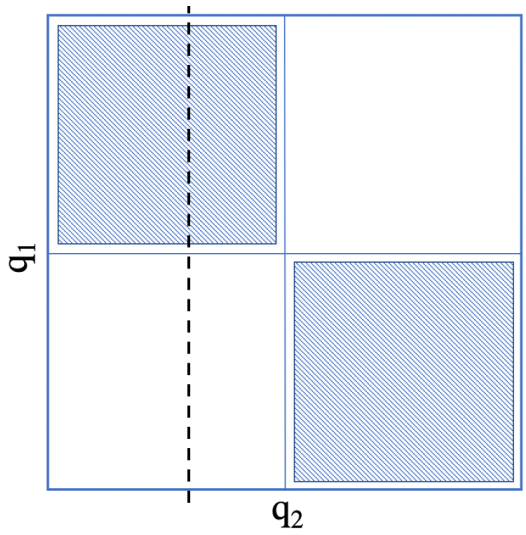


Figure A1: An example of a pattern that cannot be captured by recursive partitioning. Observations in the white and crosshatched region have different statistics. Any split, such as the black dashed line, result in two sub-partitions that have similar average statistics.

is equal to a.

$$\begin{aligned} \max_{b} \quad & f(b) = \sum_i a_i \ln b_i \\ \text{s.t.} \quad & \sum_i b_i = 1 \end{aligned}$$

Proof. It is a constrained optimization problem that can be solved by maximizing the Lagrangian function.

$$\begin{aligned} \mathcal{L}(a, b, \lambda) &= \sum_i a_i \ln b_i - \lambda(\sum_i b_i - 1) \\ \nabla\mathcal{L}(b, \lambda) &= 0 \\ \frac{\partial \mathcal{L}}{\partial b_i} &= \frac{a_i}{b_i} - \lambda = 0 \Rightarrow a_i = \lambda b_i \Rightarrow \sum a_i = \lambda \sum b_i \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= \sum_i b_i - 1 = 0 \Rightarrow \sum_i b_i = 1 \\ &\Rightarrow \lambda = 1 \Rightarrow b_i = a_i \end{aligned}$$

□

A.1 Proof of Likelihood Increase

In this section, we prove that the likelihood function $\mathcal{L}(\theta_r^*, \mathbf{\Pi}_r)$ increases at each iteration of our recursive partitioning algorithm. Formally, we prove

$$\mathcal{L}(\theta_r^*, \mathbf{\Pi}_r) \leq \mathcal{L}(\theta_{r+1}^*, \mathbf{\Pi}_{r+1}) \tag{A4}$$

where $\mathbf{\Pi}_r$ and $\mathbf{\Pi}_{r+1}$ are the discretizations generated by the proposed recursive partitioning in iterations r and $r+1$ respectively, $\theta_r^* = \text{argmax}_\theta \, \mathcal{L}(\theta, \mathbf{\Pi}_r)$, and $\theta_{r+1}^* = \text{argmax}_\theta \, \mathcal{L}(\theta, \mathbf{\Pi}_{r+1})$.

Theorem 1. For any candidate additional split, noted by $\{k, q, z\}$, to a discretization $\mathbf{\Pi}$, where k is a partition in $\mathbf{\Pi}$, q is a feature in Q, and z is a value within the range of possible values for q in k, the following inequalities hold

$$\exists \;\; \theta' : \mathcal{L}_{dc}(\theta^*, \mathbf{\Pi}) \;\; \leq \mathcal{L}_{dc}(\theta', \mathbf{\Pi}') \tag{A5}$$

$$\mathcal{L}_{st}(\mathbf{\Pi}) \;\;\;\; \leq \mathcal{L}_{st}(\mathbf{\Pi}') \tag{A6}$$

where $\mathbf{\Pi}' = \mathbf{\Pi} + \{k, q, z\}$ is the discretization after adding the candidate split, and $\theta^* = \text{argmax}_\theta \, \mathcal{L}(\theta, \mathbf{\Pi})$ is the optimal parameters given discretization $\mathbf{\Pi}$.

We prove Inequalities A5 and A6 separately. First, in Lemma 2, we create a new parameter set θ' for the discretization $\mathbf{\Pi}'$ from θ^* that satisfies a certain inequality. Next, we show that Inequality A5 holds for the new parameter set θ' and $\mathbf{\Pi}'$. Finally, we prove that for any two partitions $\mathbf{\Pi}$ and $\mathbf{\Pi}'$ such that $\mathbf{\Pi}$ is the parent of $\mathbf{\Pi}'$ the inequality A6 holds. Please note that discretization $\mathbf{\Pi}$ is the parent of any discretization that is created by adding additional splits to $\mathbf{\Pi}$.

Lemma 2. There is a flow utility coefficient set θ' such that for any $x \in \mathbf{X}$, $q \in \mathbf{Q}$ and $j \in \mathbf{J}$, the following equation holds

$$\begin{aligned} \exp v(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}') &= \exp v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) \\ &\quad + \Big[\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q)) \Big] \exp \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi}) \end{aligned} \tag{A7}$$

Proof. Using the equality $\log(a+b) = \log(a) + \log(1 + b/a)$, we write Equation (A7) as follows:

$$\begin{aligned} v(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}') &= v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) + \log \Big(1 + \frac{\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q))}{\frac{\exp v(x, \Pi(q), j; \theta^*, \mathbf{\Pi})}{\exp \bar{V}(x, \Pi(q); \theta^*, \mathbf{\Pi})}} \Big) \\ &= v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) + \log \Big(1 + \frac{\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q))}{\hat{p}(j|x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})} \Big) \end{aligned} \tag{A8}$$

Using the formulation of the choice-specific value function (Equation (A2)), we can write Equation (A8) in terms of the flow utility and expected value functions as follows:

$$\begin{aligned} u(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}') = u(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) \\ + \beta \sum_{x' \in X} \sum_{\pi \in \mathbf{\Pi}} \bar{V}(x', \pi; \theta^*, \mathbf{\Pi}) g(x', \pi | x, \mathbf{\Pi}(q), j) \\ - \beta \sum_{x' \in X} \sum_{\pi \in \mathbf{\Pi}'} \bar{V}(x', \pi; \theta', \mathbf{\Pi}') g(x', \pi | x, \mathbf{\Pi}'(q), j) \\ + \log \Big(1 + \frac{\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q))}{\hat{p}(j|x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})} \Big) \end{aligned} \tag{A9}$$

We assume a non-parametric functional form for the utility function; therefore, we can calculate a set of new utility values for each observable state $\{x, \pi\}$ based on the above equation that satisfies Equation A7. However, the right-hand side of Equation (A9) uses θ', which we are trying to calculate. If we want to use Equation (A9), we need to prove that this equation has a unique answer. Alternatively, we prove that if Equation A7 holds, the value functions in the new partitioning and old partitioning are equal.

$$\begin{aligned} &\bar{V}(x, \mathbf{\Pi}'(q); \theta', \mathbf{\Pi}') \\ &= \log \sum_{j \in \mathbf{J}} \exp v(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}') \\ &= \log \sum_{j \in \mathbf{J}} \Big(\exp v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) + \big[\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q)) \big] \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi}) \Big) \\ &= \log \Big(\sum_{j \in \mathbf{J}} \exp v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) + \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi}) \sum_{j \in \mathbf{J}} \big[\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q)) \big] \Big) \\ &= \log \sum_{j \in \mathbf{J}} \exp v(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) \\ &= \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi}) \end{aligned} \tag{A10}$$

where the equality from line 3^{rd} to line 4^{th} comes from $\sum_{j \in \mathbf{J}} \big[\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q)) \big] = 0$.

We can now write Equation (A9) as:

$$
\begin{aligned}
u(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}') = u(x, \mathbf{\Pi}(q), j; \theta^*, \mathbf{\Pi}) \\
&+ \beta \sum_{x' \in X} \sum_{\pi \in \mathbf{\Pi}} \bar{V}(x', \pi; \theta^*, \mathbf{\Pi}) g(x', \pi | x, \mathbf{\Pi}(q), j) \\
&- \beta \sum_{x' \in X} \sum_{\pi \in \mathbf{\Pi}'} \bar{V}(x', \mathbf{\Pi}(\pi); \theta^*, \mathbf{\Pi}) g(x', \pi | x, \mathbf{\Pi}'(q), j) \\
&+ \log \Big(1 + \frac{\Pr(j|x, \mathbf{\Pi}'(q)) - \Pr(j|x, \mathbf{\Pi}(q))}{\hat{p}(j|x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})} \Big)
\end{aligned} \tag{A11}
$$

where $\mathbf{\Pi}(\pi)$ is the partition in $\mathbf{\Pi}$ that includes all $\pi \in \mathbf{\Pi}'$. Equation A11 is not dependent on θ', and can be used to calculate a set of utility function values (θ') for partitioning $\mathbf{\Pi}'$. We can show that if one substitute $u(x, \mathbf{\Pi}'(q), j; \theta', \mathbf{\Pi}')$ from Equation A11 into the left hand side of Equation A7, then we should see that the Equation holds. Essentially, we are deriving a θ' from the current partition $\mathbf{\Pi}$ and θ^* such that Equation A7 holds. □

Proof of Inequality (A5). We show in Lemma 2 that there is a θ' such that Equality A7 holds, and for any $x \in X$ and $q \in \mathbf{Q}$ we have $\bar{V}(x, \mathbf{\Pi}'(q); \theta', \mathbf{\Pi}') = \bar{V}(x, \mathbf{\Pi}(q); \theta^*, \mathbf{\Pi})$. Since $\bar{V}(.; \theta^*, \mathbf{\Pi})$ is a fixed point solution to the standard contraction mapping problem for the Bellman equation, $\bar{V}(.; \theta', \mathbf{\Pi}')$ generated in Lemma 2 is a solution to the contraction mapping in Bellman equation as well. Additionally, we have

$$
\begin{aligned}
\mathcal{L}_{dc}(\theta', \mathbf{\Pi}') &= \sum_{i=1}^{N} \sum_{t=1}^{T} \log \hat{p}(d_{it} | x_{it}, \mathbf{\Pi}'(q_{it}); \theta', \mathbf{\Pi}') \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T} \log \frac{\exp v(x_{it}, \mathbf{\Pi}(q_{it}), d_{it}; \theta', \mathbf{\Pi}')}{\exp \bar{V}(x_{it}, \mathbf{\Pi}(q_{it}); \theta', \mathbf{\Pi}')} \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T} \log \frac{\exp v(x_{it}, \mathbf{\Pi}(q_{it}), d_{it}; \theta', \mathbf{\Pi}')}{\exp \bar{V}(x_{it}, \mathbf{\Pi}(q_{it}); \theta^*, \mathbf{\Pi})} \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T} \log \Big(\frac{\exp v(x_{it}, \mathbf{\Pi}(q_{it}), d_{it}; \theta^*, \mathbf{\Pi})}{\exp \bar{V}(x_{it}, \mathbf{\Pi}(q_{it}); \theta^*, \mathbf{\Pi})} + \Pr(d_{it} | x_{it}, \mathbf{\Pi}'(q_{it})) - \Pr(d_{it} | x_{it}, \mathbf{\Pi}(q_{it})) \Big)
\end{aligned} \tag{A12}
$$

Please note the equality between line two and line three is derived from Equation (A10). Also, we use Equation (A7) to go from three and line four. Using $\log(a + b) = \log(a) + \log(1 + b/a)$,

we have the following

$$
\begin{aligned}
\mathcal{L}_{dc}(\theta', \mathbf{\Pi}') &= \sum_{i=1}^{N}\sum_{t=1}^{T} \log\Big(\frac{\exp v(x_{it}, \mathbf{\Pi}(q_{it}), d_{it}; \theta^*, \mathbf{\Pi})}{\exp \bar{V}(x_{it}, \mathbf{\Pi}(q_{it}); \theta^*, \mathbf{\Pi})} + \Pr(d_{it}|x_{it}, \mathbf{\Pi}'(q_{it})) - \Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}))\Big) \\
&= \sum_{i=1}^{N}\sum_{t=1}^{T} \log\Big(\frac{\exp v(x_{it}, \mathbf{\Pi}(q_{it}), d_{it}; \theta^*, \mathbf{\Pi})}{\exp \bar{V}(x_{it}, \mathbf{\Pi}(q_{it}); \theta^*, \mathbf{\Pi})}\Big) + \sum_{i=1}^{N}\sum_{t=1}^{T} \log\Big(1 + \frac{\Pr(d_{it}|x_{it}, \mathbf{\Pi}'(q_{it})) - \Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}))}{\frac{\exp v(x_{it}, \mathbf{\Pi}(q_{it}), d_{it}; \theta^*, \mathbf{\Pi})}{\exp \bar{V}(x_{it}, \mathbf{\Pi}(q_{it}); \theta^*, \mathbf{\Pi})}}\Big) \\
&= \mathcal{L}_{dc}(\theta^*, \mathbf{\Pi}) + \sum_{i=1}^{N}\sum_{t=1}^{T} \log\Big(1 + \frac{\Pr(d_{it}|x_{it}, \mathbf{\Pi}'(q_{it})) - \Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}))}{\hat{p}(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}); \theta^*, \mathbf{\Pi})}\Big) \\
\Rightarrow \mathcal{L}_{dc}(\theta', \mathbf{\Pi}') - \mathcal{L}_{dc}(\theta^*, \mathbf{\Pi}) &= \sum_{i=1}^{N}\sum_{t=1}^{T} \log\Big(1 + \frac{\Pr(d_{it}|x_{it}, \mathbf{\Pi}'(q_{it})) - \Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}))}{\hat{p}(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}); \theta^*, \mathbf{\Pi})}\Big)
\end{aligned}
\tag{A13}
$$

We will show that the right-hand side of Equation A13 is positive; thus, proving that the likelihood is increasing. Assuming $NT \to \infty$, and our predicted choice probabilities are consistent, concludes $\hat{p}(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}); \theta^*, \mathbf{\Pi}) = \Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}))$[8]. Replacing the predicted choice probability with its counter-part conditional choice probability in Equation A13 yields

$$
\begin{aligned}
&\sum_{i=1}^{N}\sum_{t=1}^{T} \log\Big(1 + \frac{\Pr(d_{it}|x_{it}, \mathbf{\Pi}'(q_{it})) - \Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}))}{\hat{p}(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}); \theta^*, \mathbf{\Pi})}\Big) \\
&= \sum_{i=1}^{N}\sum_{t=1}^{T} \log\Big(1 + \frac{\Pr(d_{it}|x_{it}, \mathbf{\Pi}'(q_{it})) - \Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}))}{\Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}))}\Big) \\
&= \sum_{i=1}^{N}\sum_{t=1}^{T} \log \frac{\Pr(d_{it}|x_{it}, \mathbf{\Pi}'(q_{it}))}{\Pr(d_{it}|x_{it}, \mathbf{\Pi}(q_{it}))}
\end{aligned}
\tag{A14}
$$

The value inside the summation in Equation (A14) is equal to zero for all the observations, except for those where q_{it} lands in the newly split partition. Let us call the partition before the split $\pi_p \in \mathbf{\Pi}$, and the two resulting partitions $\{\pi_l, \pi_r\} \in \mathbf{\Pi}'$. Also let $N(x, \pi)$ denote the number of observations where $x_{it} = x$ and $\mathbf{\Pi}'(q_{it}) = \pi$, and $N(x, \pi, j)$ denote the number where in addition

[8]We can also conclude this equality since we assume a fully non-parametric form for utility function in Assumption 1.

to the aforementioned conditions $d_{it} = j$. We can write Equation (A14) as follows:

$$\sum_{i=1}^{N}\sum_{t=1}^{T}\log\frac{\Pr(d_{it}|x_{it},\mathbf{\Pi}'(q_{it}))}{\Pr(d_{it}|x_{it},\mathbf{\Pi}(q_{it}))} = \sum_{x\in X}\sum_{\pi\in\{\pi_l,\pi_r\}}\sum_{j\in\mathbf{J}} N(x,\pi,j)\log\big(\Pr(j|x,\pi)-\Pr(j|x,\pi_p)\big)$$
$$= \sum_{x\in X}\sum_{\pi\in\{\pi_l,\pi_r\}} N(x,\pi)\sum_{j\in\mathbf{J}}\big(\Pr(j|x,\pi)\log\Pr(j|x,\pi)-\Pr(j|x,\pi)\log\Pr(j|x,\pi_p)\big)$$

According to Lemma 1, $\sum_{j\in\mathbf{J}}\Pr(j|x,\pi)\log\Pr(j|x,\pi) \geq \sum_{j\in\mathbf{J}}\Pr(j|x,\pi)\log\Pr(j|x,\pi_p)$. This inequality is strict if there is a $j \in \mathbf{J}$ such that $\Pr(j|x,\pi) \neq \Pr(j|x,\pi_p)$ for any $x \in X$ and $\pi \in \{\pi_l,\pi_r\}$. Thus, Equation (A14) is greater than or equal to zero, which proves the Inequality (A5) in Theorem 1. □

Proof of Inequality (A6). First, note that discretization $\mathbf{\Pi}$ is a parent of discretization $\mathbf{\Pi}'$. Based on the definition of parent, for every $\pi \in \mathbf{\Pi}$, there is a set of partitions $\{\pi_i\} \in \mathbf{\Pi}'$ such that $\bigcup_i \pi_i = \pi$. Let us call $\{\pi_i\}$ the child set of π in $\mathbf{\Pi}'$ and denote it by $\mathbf{\Pi}'(\pi)$. First we use the log sum inequality (Cover and Thomas, 1991) to prove that for any $\{\pi,\pi'\} \in \mathbf{\Pi}$, $\{x,x'\} \in \mathbf{X}$, and $j \in \mathbf{J}$ the following inequality holds:

$$\sum_{\pi_i\in\mathbf{\Pi}'(\pi)}\sum_{\pi'_i\in\mathbf{\Pi}'(\pi')} N(x,\pi_i,x',\pi'_i,j)\log\frac{N(x,\pi_i,x',\pi'_i,j)}{N(x,\pi_i)N(x',\pi'_i,j)} \geq N(x,\pi,x',\pi',j)\log\frac{N(x,\pi,x',\pi',j)}{N(x,\pi)N(x',\pi',j)} \tag{A15}$$

We prove this inequality by applying the log sum inequality twice. First for a given $\pi_i \in \mathbf{\Pi}'(\pi)$ the following holds according to log sum inequality[9].

$$\sum_{\pi'_i\in\mathbf{\Pi}'(\pi')} N(x,\pi_i,x',\pi'_i,j)\log\frac{N(x,\pi_i,x',\pi'_i,j)}{N(x',\pi'_i,j)} \geq N(x,\pi_i,x',\pi',j)\log\frac{N(x,\pi_i,x',\pi',j)}{N(x',\pi',j)} \tag{A16}$$

which by subtracting $N(x,\pi_i,x',\pi',j)\log N(x,\pi_i)$ from both sides changes to

$$\sum_{\pi'_i\in\mathbf{\Pi}'(\pi')} N(x,\pi_i,x',\pi'_i,j)\log\frac{N(x,\pi_i,x',\pi'_i,j)}{N(x',\pi'_i,j)N(x,\pi_i)} \geq N(x,\pi_i,x',\pi',j)\log\frac{N(x,\pi_i,x',\pi',j)}{N(x',\pi',j)N(x,\pi_i)}. \tag{A17}$$

[9]We have $\sum_{\pi'_i\in\mathbf{\Pi}'(\pi')} N(x,\pi_i,x',\pi'_i,j) = N(x,\pi_i,x',\pi',j)$, and $\sum_{\pi'_i\in\mathbf{\Pi}'(\pi')} N(x',\pi'_i,j) = N(x',\pi',j)$,

Similarly, according to log sum inequality we have

$$\sum_{\pi_i \in \mathbf{\Pi}'(\pi)} N(x, \pi_i, x', \pi', j) \log \frac{N(x, \pi_i, x', \pi', j)}{N(x, \pi_i)} \geq N(x, \pi, x', \pi', j) \log \frac{N(x, \pi, x', \pi', j)}{N(x, \pi)} \quad \text{(A18)}$$

which by subtracting $N(x, \pi, x', \pi', j) \log N(x', \pi', j)$ from both sides changes to

$$\sum_{\pi_i \in \mathbf{\Pi}'(\pi)} N(x, \pi_i, x', \pi', j) \log \frac{N(x, \pi_i, x', \pi', j)}{N(x, \pi_i) N(x', \pi', j)} \geq N(x, \pi, x', \pi', j) \log \frac{N(x, \pi, x', \pi', j)}{N(x, \pi) N(x', \pi', j)}. \quad \text{(A19)}$$

By merging inequalities A17 and A19 we have

$$\begin{aligned}
&\sum_{\pi_i \in \mathbf{\Pi}'(\pi)} \sum_{\pi'_i \in \mathbf{\Pi}'(\pi')} N(x, \pi_i, x', \pi'_i, j) \log \frac{N(x, \pi_i, x', \pi'_i, j)}{N(x, \pi_i) N(x', \pi'_i, j)} \\
&\qquad \geq \sum_{\pi_i \in \mathbf{\Pi}'(\pi)} N(x, \pi_i, x', \pi', j) \log \frac{N(x, \pi_i, x', \pi', j)}{N(x, \pi_i) N(x', \pi', j)} \\
&\qquad \geq N(x, \pi, x', \pi', j) \log \frac{N(x, \pi, x', \pi', j)}{N(x, \pi) N(x', \pi', j)}
\end{aligned}$$

which concludes inequality A15. We can prove the lemma by summing this inequality over all $\{\pi, \pi'\} \in \mathbf{\Pi}$, $\{x, x'\} \in \mathbf{X}$ and $j \in \mathbf{J}$. □

We proved both inequalities in the theorem 1. Next we prove that the equality happens if and only if our algorithm reaches a perfect discretization.

A.2 The proof of convergence to perfect discretization

In this section we prove that our proposed algorithm stops once it reaches a perfect discretization.

Theorem 2. The recursive partitioning algorithm discussed in §4 stops once it find a perfect discretization, i.e., $\mathcal{F}(\mathbf{\Pi}_r) = \mathcal{F}(\mathbf{\Pi}_{r+1})$ if and only if $\mathbf{\Pi}_r$ is a perfect discretization.

We prove this theorem by proving separate lemmas for decision and state transition probabilities. Let us denote the decision and transition part of $\mathcal{F}(\mathbf{\Pi})$ by $\mathcal{F}_{dc}(\mathbf{\Pi})$ and $\mathcal{F}_{tr}(\mathbf{\Pi})$ respectively.

Lemma 3. For any candidate additional split $\{k, q, z\}$ to discretization $\mathbf{\Pi}$, that splits $\pi_p \in \mathbf{\Pi}$ into $\{\pi_l, \pi_r\} \in \mathbf{\Pi}'$, we have $\mathcal{F}_{dc}(\mathbf{\Pi}) = \mathcal{F}_{dc}(\mathbf{\Pi}')$ if and only if for all $x \in \mathbf{X}$ and $j \in \mathbf{J}$ the decision probabilities in π_l and π_r are similar, i.e., $\Pr(j|x, \pi_l) = \Pr(j|x, \pi_r)$.

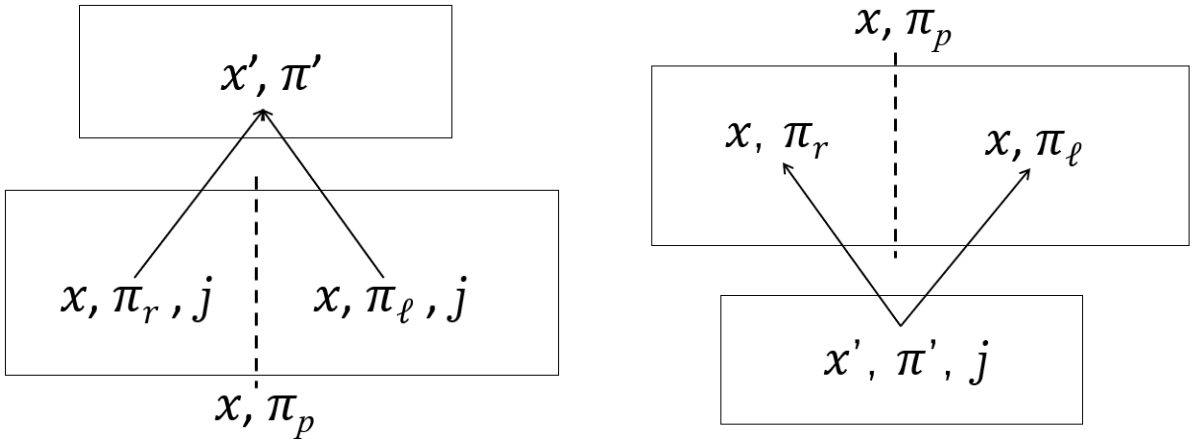
Proof. We have

$$\mathcal{F}_{dc}(\mathbf{\Pi}') - \mathcal{F}_{dc}(\mathbf{\Pi}) = N(x, \pi_l, j; \mathbf{\Pi}) \log \frac{N(x, \pi_l, j; \mathbf{\Pi})}{N(x, \pi_l; \mathbf{\Pi})} + N(x, \pi_r, j; \mathbf{\Pi}) \log \frac{N(x, \pi_r, j; \mathbf{\Pi})}{N(x, \pi_r; \mathbf{\Pi})} - N(x, \pi_p, j; \mathbf{\Pi}) \log \frac{N(x, \pi_p, j; \mathbf{\Pi})}{N(x, \pi_p; \mathbf{\Pi})}$$

According to log sum inequality the right-hand side of this equality is equal to zero if and only if $\frac{N(x,\pi_r,j;\mathbf{\Pi})}{N(x,\pi_r;\mathbf{\Pi})} = \frac{N(x,\pi_l,j;\mathbf{\Pi})}{N(x,\pi_l;\mathbf{\Pi})}$, which concludes $\Pr(j|x, \pi_l) = \Pr(j|x, \pi_r)$. □

Lemma 4. For any candidate additional split $\{k, q, z\}$ to discretization $\mathbf{\Pi}$, that splits $\pi_p \in \mathbf{\Pi}$ into $\{\pi_l, \pi_r\} \in \mathbf{\Pi}'$, we have $\mathcal{F}_{tr}(\mathbf{\Pi}) = \mathcal{F}_{tr}(\mathbf{\Pi}')$ if and only if for all $\{x, x'\} \in \mathbf{X}$, $\pi' \in \mathbf{\Pi}'$, and $j \in \mathbf{J}$ the following equations hold

$$\Pr(x', \pi'|x, \pi_l, j) = \Pr(x', \pi'|x, \pi_r, j)$$
$$\frac{\Pr(x, \pi_l|x', \pi', j)}{N(x, \pi_l)} = \frac{\Pr(x, \pi_r|x', \pi', j)}{N(x, \pi_r)}$$



(a) Transition from the new sub-partitions

(b) Transition to the new sub-partitions

Figure A2: The change in likelihood from transition-to and transition-from perspective.

Proof. We proved in Web AppendixA.1 that likelihood, $\mathcal{L}(\theta_t)$, increases with any additional split. We assumed a completely non-parametric form for the state transition part of the likelihood function. Therefore the state transition of the likelihood function and recursive partitioning objective function are the same, i.e., $\mathcal{L}_{tr}(\mathbf{\Pi}) = \mathcal{F}_{tr}(\mathbf{\Pi})$. Here we prove that the increment in likelihood, and consequently in $\mathcal{F}(\theta_t)$, is equal to zero if and only if Lemma's equations hold. The split changes the likelihood by changing the transition-to and average transition-from probabilities presented in figure

A2. We can calculate the changes in likelihood with respect to each of these changes separately. First, according to (a) in figure A2 we have

$$\mathcal{F}(\mathbf{\Pi}') - \mathcal{F}(\mathbf{\Pi}) = \sum_{x,x' \in \mathbf{X}} \sum_{\pi' \in \mathbf{\Pi}'} \sum_{j \in \mathbf{J}} N(x', \pi') \Bigg(\frac{N(x', \pi', x, \pi_r, j)}{N(x', \pi')} \log \frac{N(x', \pi', x, \pi_r, j)}{N(x', \pi') N(x, \pi_r, j)} + \frac{N(x', \pi', x, \pi_l, j)}{N(x', \pi')} \log \frac{N(x', \pi', x, \pi_l, j)}{N(x', \pi') N(x, \pi_l, j)} - \frac{N(x', \pi', x, \pi_p, j)}{N(x', \pi')} \log \frac{N(x', \pi', x, \pi_p, j)}{N(x', \pi') N(x, \pi_p, j)} \Bigg)$$

According to log sum inequality the term in the parentheses is greater or equal to zero. The left-hand side is equal to zero if and only if $\frac{N(x',\pi',x,\pi_l,j)}{N(x,\pi_l,j)} = \frac{N(x',\pi',x,\pi_r,j)}{N(x,\pi_r,j)} = \frac{N(x',\pi',x,\pi_p,j)}{N(x,\pi_p,j)}$, for every $\{x, x'\} \in \mathbf{X}$, $j \in \mathbf{J}$ and $\pi' \in \mathbf{\Pi}'$. This concludes the first equation of the lemma.

We can conclude the second equation similarly with (b) in figure A2 as the following

$$\mathcal{F}(\mathbf{\Pi}') - \mathcal{F}(\mathbf{\Pi}) = \sum_{x,x' \in \mathbf{X}} \sum_{\pi' \in \mathbf{\Pi}'} \sum_{j \in \mathbf{J}} N(x', \pi', j) \Bigg(\frac{N(x, \pi_r, x', \pi', j)}{N(x', \pi', j)} \log \frac{N(x, \pi_r, x', \pi', j)}{N(x, \pi_r) N(x', \pi', j)} + \frac{N(x, \pi_l, x', \pi', j)}{N(x', \pi', j)} \log \frac{N(x, \pi_l, x', \pi', j)}{N(x, \pi_l) N(x', \pi', j)} - \frac{N(x, \pi_p, x', \pi', j)}{N(x', \pi', j)} \log \frac{N(x, \pi_p, x', \pi', j)}{N(x, \pi_p) N(x', \pi', j)} \Bigg)$$

Again, according to log sum inequality the term in the parentheses is greater or equal to zero. The left-hand side is equal to zero if and only if $\frac{N(x',\pi',x,\pi_l,j)}{N(x,\pi_l)N(x',\pi',j)} = \frac{N(x',\pi',x,\pi_r,j)}{N(x,\pi_r)N(x',\pi',j)} = \frac{N(x',\pi',x,\pi_p,j)}{N(x,\pi_p)N(x',\pi',j)}$, for every $\{x, x'\} \in \mathbf{X}$, $j \in \mathbf{J}$ and $\pi' \in \mathbf{\Pi}'$. This concludes the second equation of the lemma. □

Proof for theorem 2. Now we prove theorem 2 using lemmas 4 and 3. Assume that our algorithm stops at iteration r. Given that the algorithm stops once no split increase the $\mathcal{F}$ function, for any additional split that generates candidate partition $\mathbf{\Pi}'$, we have $\mathcal{F}(\mathbf{\Pi}') = \mathcal{F}(\mathbf{\Pi}_r)$. Therefore, according to lemmas 4 and 3 for any additional split to $\mathbf{\Pi}_r$ the following equalities hold for all

$\{x, x'\} \in \mathbf{X}$, $\pi' \in \mathbf{\Pi}_r$ and $j \in \mathbf{J}$.

$$\begin{aligned}
\Pr(d|x, \pi_l) &= \Pr(d|x, \pi_r) \\
\Pr(x', \pi'|x, \pi_l, j) &= \Pr(x', \pi'|x, \pi_r, j) \\
\frac{\Pr(x, \pi_l|x', \pi', j)}{N(x, \pi_l)} &= \frac{\Pr(x, \pi_r|x', \pi', j)}{N(x, \pi_r)}
\end{aligned}$$

which concludes that $\mathbf{\Pi}_r$ is a perfect discretization according to corollary 1. □

B Random discretization generator algorithm

This section explains the random discretization generator algorithm that we used in the second simulation study. The intuition for this algorithm is very similar to recursive partitioning – in each step, we randomly select one of the partitions and split it into two partitions along one of the first 10 variables in $\mathbf{Q}$. Please note that we only used the first 10 variables in $\mathbf{Q}$ for partitioning, and the following 20 variables are added as irrelevant variables to show the robustness of the algorithm to irrelevant variables. In addition, to prevent very small or very large partitions, the random partition selection is weighted by the size of the partition: big partitions are more likely to be selected than smaller partitions. Formally, the random partitioning algorithm is as follows.

- Initialize $\mathbf{\Pi}_0$ as one partition equal to the full covariate space.
- Do the following for 15 rounds.
  - Step 1: Randomly select a partition from $\mathbf{\Pi}_r$ weighted by $(\frac{1}{\text{partition's total splits}})^2$
  - Step 2: Randomly select a variable from the first 10 variables
  - Step 3: Split the selected partition into two from the midpoint along the selected variable. If there is no exact midpoint to split, then pick the smaller of the two integers closest to the midpoint as the splitting point. Go back to Step 1 and repeat this round if the split is not possible[10].

The total number of splits for each partition is the total of times their parents have selected to be split. If the total split for a partition is 5, it means that it takes five splits from $\mathbf{\Pi}_0$ to get to this

[10]Recall that the variables in $\mathbf{Q}$ only take integer values from 0 to 10. Therefore, if the selected variable in Step 2, say q_1, starts at 9 and ends at 10 in the chosen partition, then further splits along q_1 within this partition are not possible.

partition. Note that the total number of splits of a partition is negatively correlated with the size of the partition.

We also vary the replacement cost for each partition to add a decision variation to the model that is not caused by state transition. The replacement cost of a partition is calculated based on the range of its first 10 variables as the following.

$$f_{dc}(\pi) = 5 - \frac{\sum_{i=1}^{10}(v_i^{min}(\pi) + v_i^{max}(\pi))}{10} \tag{A20}$$

where $v_i^{min}(\pi)$ and $v_i^{max}(\pi)$ are the minimum and maximum range of i^{th} variable in partition π. We choose this formulation for the replacement cost to create a good balance between decision variation caused by state transition or replacement cost. Figure A3 depicts the distribution of total split and replacement costs in the 1500 generated partitions across all the 100 generated discretizations in the second simulation study.

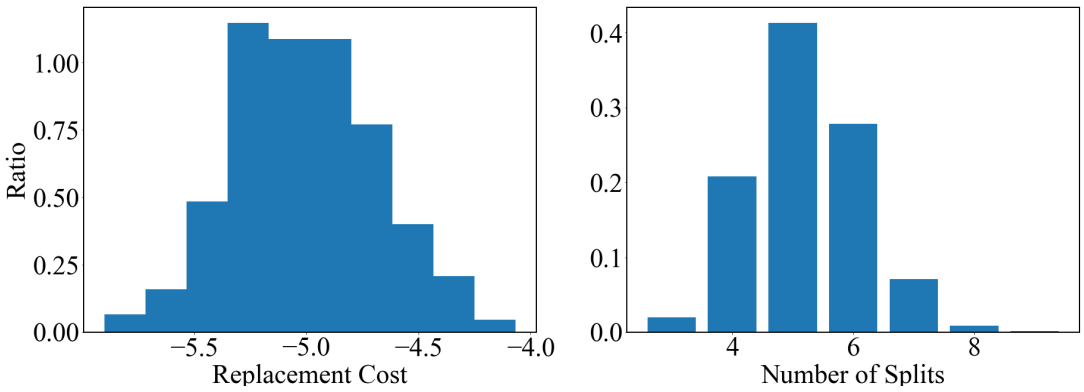


Figure A3: The histogram of generated partitions' calculated replacement cost, and number of splits in the 100 rounds of partitioning generation.