Assisted specification of discrete choice models with piece-wise linear utility functions

Nicolas Salvadé*[1] and Tim Hillel[2]

[1]PhD Student, Department of Civil, Environmental and Geomatic Engineering, University College London, United Kingdom
[2]Lecturer, Department of Civil, Environmental and Geomatic Engineering, University College London, United Kingdom

Short summary

This paper extends RUMBoost, a model that enhances discrete choice models (DCMs) by automatically learning non-linear parameters using gradient boosting, to piece-wise linear functions. The model output is used to specify DCMs with piece-wise linear utility functions. The methodology is demonstrated using the Swissmetro dataset, showing significant differences in behaviour over models with linear utility function specifications.
Keywords: Discrete Choice Models, Ensemble Learning, Machine Learning, Piece-wise Linear Functions, Assisted Specification

1 Introduction

Discrete choice models (DCMs) based on the Random Utility theory have been the standard for more than 50 years in modelling human behaviour. Their advantages are straightforward:

1. interpretability;
2. behavioural consistency; and
3. robustness and ease of estimation.

However, these advantages are also closely related to their disadvantages. In fact, they rely on a rather inflexible *linear-in-parameter* utility function. While non-linearities are possible, they must be introduced by the modeller. This time-consuming and cumbersome task fails to guarantee optimal transformations due to the size of the search space. Several studies have attempted to alleviate this issue by providing assisted specification for an MNL (Hillel et al., 2019), for a mixed-logit model (Paz et al., 2019), for a portfolio model (Hernandez et al., 2023), and for more general combinations of models (Ortelli et al., 2021). While most of these studies consider variable interaction and non-linearities, they rely on either pre-specified model specifications or observation of non-linearities and variable interactions by the modeller.
In order to automatically learn non-linearities, RUMBoost, a model using gradient boosting to automatically impute non-linear parameters while retaining some key features of DCMs, has been developed (Salvadé & Hillel, 2025). In this work, it is shown that the predictive performance of DCMs can be greatly enhanced thanks to these non-linear parameters. However, the resulting utility functions are piece-wise constants with either zero or infinite gradients, making them unsuitable for computing behavioural indicators such as the *Value of Time* (VoT) or elasticities. Therefore, a computationally expensive smoothing algorithm based on monotonic cubic splines is used to obtain utility functions with defined gradients.
In this paper, we present RUMBoost with piece-wise linear functions. We change the paradigm of gradient boosting by performing it in the parameter space instead of the functional space. By assuming that the utility functions are linear in parameters, this change allows the trees to output coefficients instead of utility values effectively. Therefore, we directly obtain piece-wise utility functions by aggregating all the coefficients in an ensemble and multiplying them back with the variable values. We extend the work of Shi et al. (2018) by incorporating the output of the ensembles in a Rectified Linear Unit (ReLU) function, ensuring the algorithm can guarantee monotonicity. Indeed, it is sufficient to constrain the parameter signs to be positive (or negative) to guarantee positive (or negative) monotonicity.

When informing policies, model interpretability is key to ensuring a fair process. DCMs have the advantage over RUMBoost in providing formal significance tests on the parameters, allowing confidence in the model parameters. Therefore, in this work, we focus on demonstrating how we can use the output of RUMBoost with piece-wise linear functions to *automatically* specify a DCM with piece-wise linear utility functions. We write a Python script to convert the output of RUMBoost into a Biogeme model specification, and we focus on the behavioural implications arising from the piece-wise linear utility functions compared to a DCM with linear utility functions. We apply the methodology to the Swissmetro dataset, a mode choice dataset, and show that we can effectively *automatically* specify a DCM with piece-wise linear utility functions from the output of a RUMBoost model.
The rest of the paper is structured as follows: Section 2 introduces the methods used, Section 3 provides the result of the Swissmetro case study and Section 4 concludes the paper.

2 Methodology

We present a brief overview of DCMs based on random utility, piece-wise linear gradient boosting, and how it is implemented within the RUMBoost framework. We redirect the reader to Ben-Akiva & Lerman (1985); Train (2009); Friedman (2001); Chen & Guestrin (2016); Salvadé & Hillel (2025) for complete explanations on DCMs, gradient boosting and RUMBoost.

Discrete choice models

Discrete choice models based on random utility theory assume that individuals choose the alternative that maximises their utility, a latent measure of the attractivity of an alternative. The utility is usually defined as follows:

$$U_{in} = V_{in} + \varepsilon_{in} \tag{1}$$

where V_{in} is the observable utility and ε_{in} the error term capturing unobserved variables for alternative i and individual n. A widely used representation of V_{in} in DCMs is a manually specified linear-in-parameters function of the variables, such that:

$$V_{in} = \beta_{i,0} + \sum_{k=1}^{K} \beta_{i,k} x_{ikn} \tag{2}$$

where $K+1$ parameters associated with K variables x_{ikn} have to be estimated from the data. Here, $\beta_{i,0}$ is an *alternative specific constant* (ASC). For simplicity, we consider here the MNL model, where the error term is assumed to be i.i.d, following an extreme value distribution of type 1 (Gumbel). For the MNL, in a classification problem with J classes, the probabilities are given by the multi-class logistic function:

$$P_{in} = \frac{e^{V_{in}}}{\sum_j^J e^{V_{jn}}} \tag{3}$$

Gradient boosting

Gradient boosting algorithms make use of an ensemble of regression trees to partition data with split points and impute a leaf value (constant) for each partition. At each iteration m, for a classification task, a new decision tree is added to the ensemble to minimise the loss function directly:

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{l=1}^{L} \ell(y_{in}, \hat{y}_{in}) \tag{4}$$

where:

- N is the number of individuals;
- L is the number of alternatives;
- ℓ is the loss function for each individual and alternative, usually the cross-entropy loss (akin to log-likelihood function) for a classification task;

- y_{in} is 1 if individual n chooses alternative i; and
- $\hat{y}_{in} = \sum_m f_{i,m}(x_{in})$ is the prediction of the model at current iteration m for individual n to choose alternative i. These predictions are then turned into probabilities with the softmax function (akin to a multi-class logistic function) for a classification task. We interpret $\hat{y}_{in}$ as the deterministic utility V_{in}.

When adding a new tree, the optimal leaf values can be computed by taking the second-order Taylor expansion of the loss function:

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{i=1}^{J} \ell(y_{in}, \hat{y}_{in,m-1}) + g_{in} f_{i,m}(x_{in}) + \frac{1}{2} h_{in} f_{i,m}(x_{in})^2 \tag{5}$$

where $g_{in} = \partial \ell(y_{in}, \hat{y}_{in,m-1}) / \partial \hat{y}_{in,m-1}$, $h_{in} = \partial^2 \ell(y_{in}, \hat{y}_{in,m-1}) / \partial^2 \hat{y}_{in,m-1}$ are the first and second derivatives of the loss function with respect to the prediction for observation n and alternative i. In traditional gradient boosting, the predictive function $f_{i,m}(x_{in})$ is composed of L terminal nodes arising from the partitions of the data points x_{in} from the split points of the decision tree. Note that these terminal nodes are non-overlapping, which means that the data points will fall into one and only one terminal node, such that:

$$f_{i,m}(x_{in}) = \sum_{l=1}^{L} \gamma_{l,im} \mathbf{1}(x_{in} \in l) \tag{6}$$

where $\mathbf{1}(x_{in} \in l)$ is 1 if x_{in} falls in terminal node l, 0 otherwise. To find the optimal leaf value $\gamma_{l,im}$ of a terminal node l, an alternative i, and an iteration m, we set the first derivative of Equation 5 with respect to the leaf value to zero. After rearranging the terms, we obtain:

$$\gamma_{l,im} = -\frac{(\sum_{n \in l} g_{in})}{(\sum_{n \in l} h_{in})} \tag{7}$$

Essentially, gradient boosting mimics a *quasi-Newton* method with a diagonal approximation of the Hessian.

Gradient boosting with linear trees

We change the paradigm of gradient boosting and assume that the terminal nodes output coefficients of linear models. That is, the model's prediction $\hat{y}_{in}$ is equal to a parameter and no longer to a deterministic utility. The fundamental difference of our approach with Shi et al. (2018) is that we want to enforce continuity at each split point. In their approach, the authors fit independent linear models $\beta x + \alpha$ on the left and right sides of the split point. Instead, we consider that each non-binary variable follows a continuous piece-wise linear specification of the following form:

$$x_{kl} = \begin{cases} 0 & \text{if } t < a \\ t - a & \text{if } a \leq t < a + b \\ b & \text{otherwise} \end{cases} \qquad x_{kl} = \max(0, \min(t - a, b)) \tag{8}$$

And the deterministic utility becomes:

$$V_{in} = \beta_{i,0} + \sum_{k=1}^{K} \sum_{l=1}^{L_k} \beta_{i,kl} x_{ikln} \tag{9}$$

where L_k is the number of discontinuities for variable k. Equation 5 can be used again to quickly find the optimal parameter values, with the difference that the first and second derivatives need to be multiplied by the data points. Indeed, from the chain rule, we have, for the first derivative:

$$g'_{in} = \frac{\partial \ell(y_{in}, \hat{y}_{in,m-1})}{\partial \hat{y}_{ink,m-1}} = \frac{\partial \ell(y_{in}, V_{in,m-1})}{\partial V_{in,m-1}} \cdot \frac{\partial V_{in}}{\partial \hat{y}_{ink,m-1}} = g_{in} x_{ikln} \tag{10}$$

and for the second derivative:

$$
\begin{aligned}
h'_{in} &= \frac{\partial^2 \ell(y_{in}, \hat{y}_{in,m-1})}{\partial^2 \hat{y}_{ink,m-1}} = \frac{\partial}{\partial \hat{y}_{ink,m-1}} \left(\frac{\partial \ell(y_{in}, V_{in,m-1})}{\partial V_{in,m-1}} \cdot \frac{\partial V_{in}}{\partial \hat{y}_{ink,m-1}} \right) \\
&= \frac{\partial^2 \ell(y_{in}, V_{in,m-1})}{\partial^2 V_{in,m-1}} \cdot \left(\frac{\partial V_{in}}{\partial \hat{y}_{ink,m-1}} \right)^2 + \frac{\partial \ell(y_{in}, V_{in,m-1})}{\partial V_{in,m-1}} \cdot \frac{\partial^2 V_{in}}{\partial^2 \hat{y}_{ink,m-1}} \\
&= h_{in} x_{ikln}^2
\end{aligned} \tag{11}
$$

The optimal leaf values can be computed in the same fashion as in Equation 7:

$$
\gamma_{l,im} = -\frac{(\sum_{n \in l} g'_{in})}{(\sum_{n \in l} h'_{in})} \tag{12}
$$

From the additive nature of the gradient boosting algorithm, the parameter of a piece-wise linear interval is the sum of the leaf values from all trees whose partitions include the interval. When a new tree is added to the ensemble, the split points are added to the list of discontinuities of that variable and the parameter values from all intervals are updated. Positive (or negative) monotonicity can be easily enforced by applying a ReLU function on the ensemble output to keep the parameters above (or below) zero.

RUMBoost and assisted specification of DCMs

RUMBoost uses gradient boosting to learn non-linear parameters directly from the data automatically. More specifically, each parameter in the traditional utility specification is replaced by an ensemble of regression trees. In the context of gradient boosting with linear trees, there is an exact equivalence between a RUMBoost model and a DCM based on random utility with piece-wise linear model specification. That is, we can use the automatically learnt split points or discontinuities from the trained RUMBoost model to inform the model specification of the DCM. The parameter values from RUMBoost are used to provide an initial solution to the DCM, effectively giving a warm start and reducing the optimisation time. Furthermore, DCMs generally use optimisation algorithms that use the full Hessian matrix, providing more guarantees to obtain an optimal solution than the quasi-Newton gradient boosting method. In addition, with DCMs, we can get formal significance testing of the parameters, helping to gain confidence in the learnt non-linearities of RUMBoost for policy implications.
We write a script that converts a trained RUMBoost model into a model specification from the popular choice modelling software Biogeme (Bierlaire, 2023) to be used directly within RUMBoost, therefore allowing modellers to replicate any MNL model in RUMBoost easily.

3 Results and discussion

We demonstrate our methodology on the Swissmetro dataset (Bierlaire et al., 2001), a stated preference dataset where respondents are asked to choose between Swissmetro (a new underground mag-lev transportation mode in Switzerland), car or train. We keep observations where the car alternative is available and where choice is known. After this preprocessing, we have 9036 observations from 1004 individuals. We find the optimal number of trees for the RUMBoost model with cross-validation, training the model for 67 iterations before using it for assisted specification[1]. We apply negative monotonicity constraints on the headway, travel time and cost (i.e. the parameters associated with these variables can only be negative). The model specification used for the RUMBoost and MNL models is summarised in Table 1.
The discontinuities learnt from the trained RUMBoost model are summarised in Table 2. Note that the number of split points is much lower than the number of trees in the RUMBoost model since decision trees can split on the same split points several times.
We observe that for the train travel time, there are discontinuities around 1hr, 3hrs and 4 hrs, for Swissmetro travel time around 1hr 30min and for car travel time around 1hr 45min, 2hrs and 2hrs 30min. For the train and Swissmetro costs, the discontinuities arise around 140 CHF, and for the car, around 80 CHF. Finally, there is a discontinuity for a headway of 80 minutes for the train and

[1] As this is ongoing research, we choose the optimal number of trees based on prediction performance. Future research will focus on applying multi-objective optimisation to couple the performance and behavioural interpretability of the model.

Table 1: Variables from the Swissmetro dataset used. With assisted specification, travel time, headway and cost are piece-wise linear. For the model estimation, the socio-demographic characteristics and generic attributes are normalised to 0 for the driving alternative. Purpose_X and Age_X are dummy variables where one category (return and other trips for Purpose_X, age above 39 for Age_X) is normalised to 0.

	Train	**Swissmetro**	**Driving**
Alternative-specific attributes			
Constant	✓		✓
Travel time	✓	✓	✓
Headway	✓	✓	
Cost	✓	✓	✓
socio-demographic characteristics and generic attributes			
First class traveller	✓		
Age_1: $x \leq 24$	✓	✓	✓
Age_2 : $24 < x \leq 39$	✓	✓	✓
Gender	✓	✓	✓
Purpose_1: commuting	✓	✓	✓
Purpose_2: shopping	✓	✓	✓
Purpose_3: business	✓	✓	✓
Purpose_4: leisure	✓	✓	✓

Table 2: Discontinuities learnt from RUMBoost, and used to specify the piece-wise linear MNL. Travel times and headways discontinuities are in minutes. Cost discontinuities are in CHF.

	Discontinuities
Train travel time	[52, 62.3, 68.5, 181.9, 230.3]
Train cost	[129, 149.5, 155.5]
Train headway	[80]
Swissmetro travel time	[91.4]
Swissmetro cost	[141]
Swissmetro headway	[22.5]
Car travel time	[108.2, 124.6, 149.1]
Car cost	[83.8]

22.5 minutes for Swissmetro. The parameter estimates of the piece-wise linear MNL (PWL-MNL) and the MNL are in Table 3. Note that all variables have been normalised in the range of 0 to 1 for the DCM estimation.

For the MNL, one parameter is not significant, and for the PWL-MNL, seven parameters are not significant. For the travel time, headway and cost, an upper bound (0) is applied to the parameters following the negative monotonicity constraint of RUMBoost. We use a likelihood ratio test to find the preferred model, with the null hypothesis H_0 being that the baseline MNL model is the true model. The degree of freedom is 12 (30 free parameters in PWL-MNL, 18 free parameters in MNL), which gives:

$$-2(\mathcal{L}(\hat{\beta}_{MNL}) - \mathcal{L}(\hat{\beta}_{PWL-MNL})) = 105.52 > 21.03 = \chi^2(0.05, 12) \tag{13}$$

Therefore, we reject the null hypothesis with a 95% level of confidence and prefer the PWL-MNL model.

Some differences arise in the piece-wise linear parameters and can be visualised in Figure 1a, 1b and 1c. Overall, the piece-wise linear model highlights how the perception of travel times is dampened for longer trips. The same behaviour arises for train and car costs, as well as train headway, where after some point, an increment of these variables impacts the utility function less strongly than for low values of the variable. Finally, the car cost exhibits a linear behaviour, and the Swiss metro

headway a concave behaviour.

4 Conclusions

In this paper, we show how we can automatise a piece-wise linear model specification from the output of the RUMBoost model. This can effectively reveal the population's non-linear behaviours. Further works include validating the methodology with different datasets, verifying the non-linearities with the literature, and analysing the potential predictive improvements of the assisted specified MNL over a linearly specified MNL.

References

Ben-Akiva, M. E., & Lerman, S. R. (1985). *Discrete choice analysis: theory and application to travel demand*. Cambridge, Mass: MIT Press.

Bierlaire, M. (2023). A short introduction to biogeme. *Transport and Mobility Laboratory, ENAC, EPFL*.

Bierlaire, M., Axhausen, K., & Abay, G. (2001). The acceptance of modal innovation: The case of swissmetro. In *Swiss transport research conference.*

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

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Hernandez, J. I., van Cranenburgh, S., Chorus, C., & Mouter, N. (2023). Data-driven assisted model specification for complex choice experiments data: Association rules learning and random forests for participatory value evaluation experiments. *Journal of choice modelling*, *46*, 100397.

Hillel, T., Bierlaire, M., Elshafie, M., & Jin, Y. (2019). Weak teachers: Assisted specification of discrete choice models using ensemble learning..

Ortelli, N., Hillel, T., Pereira, F. C., de Lapparent, M., & Bierlaire, M. (2021). Assisted specification of discrete choice models. *Journal of choice modelling*, *39*, 100285.

Paz, A., Arteaga, C., & Cobos, C. (2019). Specification of mixed logit models assisted by an optimization framework. *Journal of choice modelling*, *30*, 50–60.

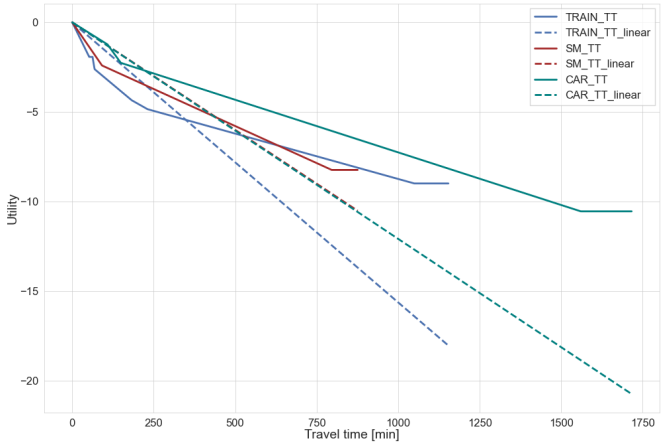
Salvadé, N., & Hillel, T. (2025). Rumboost: Gradient boosted random utility models. *Transportation Research Part C: Emerging Technologies*, *170*, 104897.

Shi, Y., Li, J., & Li, Z. (2018). Gradient boosting with piece-wise linear regression trees. *arXiv preprint arXiv:1802.05640*.

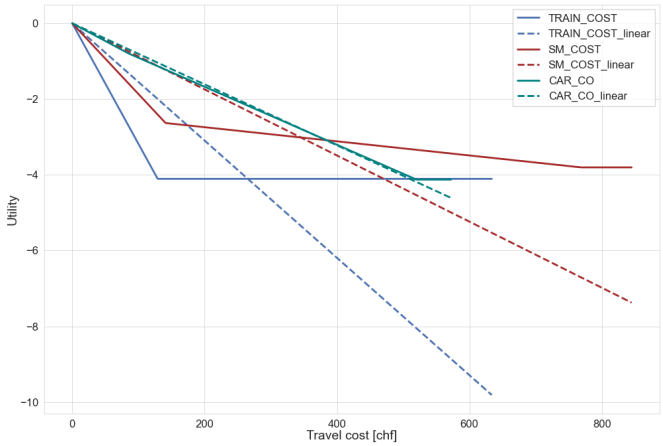
Train, K. E. (2009). *Discrete choice methods with simulation*. Cambridge university press.

Table 3: Parameter estimates for the piece-wise linear MNL (PWL-MNL) and a baseline MNL. A parameter significant at the 99% level is indicated by 3 asterisks (***), at the 95% level by 2 asterisks (**) and at the 90% level with 1 asterisk (*).

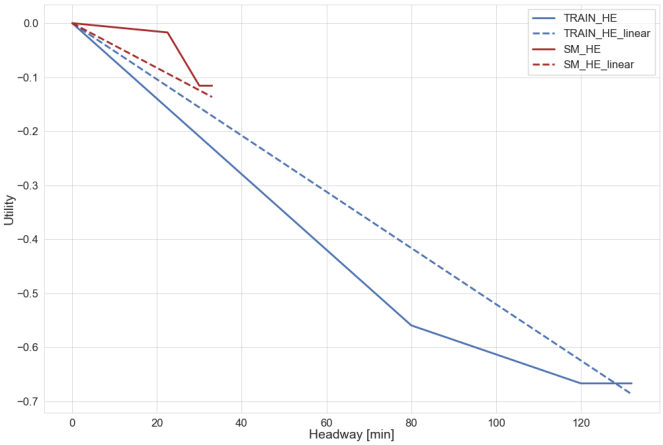
	PWL-MNL	**MNL**
Log Likelihood	-6815.97	-6868.73
ASC_TRAIN	-1.02**	2.93***
ASC_SM	-3.17***	2.64***
B_AGE_1	-0.77***	0.67***
B_AGE_2	0.59***	0.16***
B_CAR_CO	-	-4.19***
B_CAR_CO_0	-4.88***	-
B_CAR_CO_1	-3.98***	-
B_CAR_TT	-	-18.86***
B_CAR_TT_0	-17.80***	-
B_CAR_TT_1	-39.67***	-
B_CAR_TT_2	-39.03***	-
B_CAR_TT_3	-9.16**	-
B_FIRST	-0.15	-0.06
B_MALE	0.61***	-0.26***
B_PURPOSE_1	0.35*	-1.59***
B_PURPOSE_2	0.04	-1.00***
B_PURPOSE_3	-0.06	-2.25***
B_PURPOSE_4	0.41*	-2.98***
B_SM_COST	-	-6.70***
B_SM_COST_0	-14.34**	-
B_SM_COST_1	-1.43**	-
B_SM_HE	-	-0.12**
B_SM_HE_0	-0.02	-
B_SM_HE_1	-0.39	-
B_SM_TT	-	-9.66***
B_SM_TT_0	-21.00***	-
B_SM_TT_1	-6.58***	-
B_TRAIN_COST	-	-8.92***
B_TRAIN_COST_0	-18.33***	-
B_TRAIN_COST_1	0 (fixed)	-
B_TRAIN_COST_2	0 (fixed)	-
B_TRAIN_COST_3	0 (fixed)	-
B_TRAIN_HE	-	-0.62***
B_TRAIN_HE_0	-0.84***	-
B_TRAIN_HE_1	-0.32	-
B_TRAIN_TT	-	-16.40***
B_TRAIN_TT_0	-38.90***	-
B_TRAIN_TT_1	0 (fixed)	-
B_TRAIN_TT_2	-116.09***	-
B_TRAIN_TT_3	-16.01***	-
B_TRAIN_TT_4	-10.87**	-
B_TRAIN_TT_5	-5.31	-



(a) Travel time



(b) Cost



(c) Headway

Figure 1: Differences between piece-wise linear (solid lines) and linear parameters (dashed lines). The parameters are multiplied by their respective variable values.