Neural-Embedded Discrete Choice Models

by

Yafei Han

B.E., B.A. Peking University (2006)

M.S., M.C.P. Massachusetts Institute of Technology (2015)

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2019

Signature of Author .....
Department of Civil and Environmental Engineering
Aug 16, 2019

Certified by .....
P. Christopher Zegras
Professor of Mobility and Urban Planning
Thesis Supervisor

Certified by .....
Francisco C. Pereira
Professor, Technical University of Denmark
Thesis Co-Supervisor

Certified by .....
Moshe E. Ben-Akiva
Edmund K. Turner Professor of Civil and Environmental Engineering
Thesis Co-Supervisor

Accepted by .....
Colette L. Heald
Professor of Civil and Environmental Engineering
Chair, Graduate Program Committee

Neural-Embedded Discrete Choice Models

by

Yafei Han

Submitted to the Department of Civil and Environmental Engineering
on Aug 16, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Transportation

Abstract

This dissertation is motivated by the possible value of integrating theory-based discrete choice models (DCM) and data-driven neural networks. How to benefit from the strengths of both is the overarching question. I propose hybrid structures and strategies to flexibly represent taste heterogeneity, reduce potential biases, and improve predictability while keeping model interpretability. Also, I utilize neural networks' training machinery to speed up and scale up the estimation of Latent Class Choice Models (LCCMs).

First, I **embed neural networks in DCMs** to enable flexible representations of taste heterogeneity and enhance prediction accuracy. I propose two neural-embedded choice models: *TasteNet-MNL* and *nonlinear-LCCM*. Both models provide a flexible specification of **taste** as a function of individual characteristics. TasteNet-MNL extends the Multinomial Logit Model (MNL). A feed-forward neural network (TasteNet) is utilized to *predict* taste parameters as a nonlinear function of individual characteristics. Taste parameters generated by TasteNet are further fed into a parametric logit model to formulate choice probabilities. I demonstrate the effectiveness of this integrated model in capturing nonlinearity in tastes without *a priori* knowledge. Using synthetic data, TasteNet-MNL is able to recover the underlying utility specification and predict more accurately than some misspecified MNLs and continuous mixed logit models. TasteNet-MNL also provides interpretations close to the ground truth. In an application to a public dataset (Swissmetro), TasteNet-MNL achieves the best out-of-sample prediction accuracy and discovers a broader spectrum of taste variation than the benchmark MNLs with linear utility specifications.

Nonlinear-LCCM enriches the class membership model of a typical LCCM. I represent an LCCM by a neural network and add hidden layers with nonlinear transformations to its class membership model. The nonlinearity introduced by the neural network provides a flexible approximation of the mixing distribution for both *systematic* and *random taste heterogeneity*. I apply this method to model Swissmetro mode choice. The nonlinear-LCCM outperforms an LCCM with a linear class membership model with respect to the out-of-sample prediction accuracy. Nonlinear-LCCM also provides interpretable taste parameters for each latent class.

Second, I **embed DCMs in neural networks** to (partially) maintain model interpretability. I propose two general strategies: **imposing special structure** and **parameter constraints** to incorporate expert knowledge and improve interpretability.

Both TasteNet-MNL and nonlinear-LCCM can be seen as special neural networks with DCMs embedded. In TasteNet-MNL, the downstream choice model defines the meaning of each output unit of the TasteNet, through the utility definition. As for nonlinear-LCCM, the class-specific choice models define meanings of the latent classes. In addition to the special structure, I impose parameter constraints according to prior knowledge in two ways: using a transformation layer in the neural network (TasteNet-MNL); and adding a violation penalty to the objective function (nonlinear-LCCM). Using these strategies, I show both TasteNet-MNL and nonlinear-LCCM can achieve realistic behavioral/economic indicators, such as values of time, demand elasticities and choice probabilities at the disaggregate model level and at the individual level.

Third, I demonstrate the benefits of using the neural network training machinery to estimate LCCMs, especially for models with a large number of classes/parameters, estimated on a large dataset, or under certain challenging scenarios such as highly unbalanced classes and relatively small sample size(s) in the minority latent class(es). I contrast neural network estimation by Adam, a stochastic gradient descent algorithm, with the standard maximum likelihood by quasi-Newton routine, and with Expectation-Maximization (EM). Synthetic data results show similar accuracy and estimation time when the dataset is relatively small and the model has a small number of classes. Traditional estimation approaches scale poorly to a large dataset and a model with a large number of classes/parameters. They tend to encounter the small class vanishing problem. Estimation based on stochastic gradient descent is shown to be 70 times faster on a large synthetic dataset, and can achieve more stable and accurate estimates for latent classes with relatively small membership.

This thesis takes an initial step towards developing a framework to combine theory-based and data-driven methods for discrete choice modeling. I highlight the strengths and weaknesses of econometric DCMs and neural networks, and explore several ways to take advantage of both: DCMs' rigorous theory and domain knowledge; and neural networks' function approximation capability and many techniques developed to scale up estimation for big data and high-dimensional optimization. The end goal is to empower flexible model specification, scale up estimation, and at the same time maintain behavioral interpretability to a satisfactory degree, such that the hybrid approaches can support scenario analysis and discover new knowledge from behavior data with increasing granularity and complexity.

The modeling approaches and analysis in this dissertation have several limitations. Future studies can enrich the modeling frameworks and test them under various empirical settings. First, the neural-embedded choice models proposed in this dissertation focus on modeling *taste* heterogeneity. Future research can extend the models to incorporate nonlinear effects in alternative *attributes*. Second, current hybrid models have limited capacity to represent *random* heterogeneity. Future work can extend TasteNet-MNL to incorporate random taste variations; and enrich the nonlinear-LCCM to account for within-class taste heterogeneity. How to embed neu-

ral networks in continuous mixed logit models is an intriguing question. Thirdly, the variety of synthetic data scenarios and the DCMs selected for model comparison is limited. Future studies can test nonlinearity scenarios more thoroughly; and compare TasteNet-MNL and nonlinear-LCCM with other DCM structures, such as various forms of mixed logit and semi-/non-parametric DCMs. Lastly, neural-embedded choice models can complement DCMs, and potentially be integrated into trip-based model systems (e.g., a four-step model) and activity-based model systems to better understand the implications of model uncertainty for transportation planning and policy decisions.

Thesis Supervisor: P. Christopher Zegras
Title: Professor of Mobility and Urban Planning

Thesis Co-Supervisor: Francisco C. Pereira
Title: Professor, Technical University of Denmark

Thesis Co-Supervisor: Moshe E. Ben-Akiva
Title: Edmund K. Turner Professor of Civil and Environmental Engineering

Acknowledgments

I am indebted to my thesis committee that provided the guidance, challenge and encouragement I needed to complete this work. I am grateful to my advisor Prof. Christopher Zegras for his mentorship throughout my Master and PhD studies at MIT. He encouraged and supported me to pursue the dual degree in Master of Science in Transportation, where I found my interest in studying travel behaviors and built up quantitative skills. Without his gracious support, I would not be able to pursue a Ph.D. in transportation. He provided many research opportunities and gave me the freedom to pursue my interest. Without the interdisciplinary learning experience supported by him, the ideas in this thesis would not be born.

I am thankful to Prof. Francisco Pereira, for his enthusiastic support from the beginning of my thesis proposal. He provided timely and helpful feedback at each step. His expertise in both machine learning and transportation helped me see from both sides. He has always been encouraging and ready to help when I faced difficulties.

I am grateful to Prof. Moshe Ben-Akiva. He challenged me the most. As one of the founding fathers of discrete choice models, his deep knowledge and insights have significantly shaped the development of this work. I am thankful to his tough questions, honest opinions, and extraordinary standards.

I want to thank many friends and peers at MIT: Yunke Xiang, Yinjin Lee, Mazen Salah Danaf, Yifei Xie, Youssef Medhat Aboutaleb, Xiang Song, Michael Dowd, Shan Jiang, Jingsi Shaw, and Natalia M Barbour for your help and the memorable time together. My gratitudes also go to my collaborators Jimi Oke, Carlos M. Lima Azevedo, and Fang Zhao.

I want to thank CEE graduate administrators Kiley Clapper, Max Martelli, Katherine Rosa, and Eunice Kim for your awesome job and great patience.

I am indebted to my parents and grandparents, for their boundless patience, support and love.

Most of all, I am overwhelmed by the grace of God and the love I received from the family of God. This dissertation is a fruit of so many people's labor and love. I

am indebted to:

- Brother Ce Liu and sister Irene Liu for helping me with my proposal and the difficulty at every step. Thank you for letting me come over countless times, even when you were going through a difficult time. Many ideas in this work are inspired by you.

- Sister Wei Wang for your critical suggestions on some of the experiments. Thank you for spending much time discussing with me over the phone; and reviewing and providing feedback to my defense presentation.

- My house sisters: Kana san, Woojung unni, Shirley, Wei, Ruilin, Dan, Xiaohui and Sike. Thank you much for preparing a feast for me every day during the last two weeks. I had the best food ever in my life. Thank you for bearing with me, sharing my stress and carrying me with so much food and love. Thank you also for taking care of many things so that I could focus on my dissertation. I want to especially thank Kana san and Woojung unni, for praying for me since the beginning of my dissertation last Fall, and along every step of the progress.

- Kana san for your deep massage that relieved much of my pain. Thank you for always asking and listening. I am so blessed to have a "big" sister with a big heart like you to lean upon.

- Pastor Joseph and Hyejin smn for your spiritual guidance and carrying me with your prayers. You have always led me to the truth and reminded me of my identity.

- All ISM sisters and staff for your earnest prayers that carried me through the defense.

My deepest gratitude and praise is owed to God. My life cannot be explained without Him. I thank Him for bringing me to Boston and for the encounter with Graduate ABSK bible study group. I thank Pastor Paul and Dr. Rebekah Kim Jdsn who founded this family of God, where I could study the Bible and know the Lord. This work will be always the reminder of the love I owed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation & Objectives

Discrete choice models (DCM) provide a powerful econometric framework to analyze and predict choice behavior. The majority of DCMs are Random Utility Models (RUM), derived under the utility-maximization decision rule (McFadden, 1973). Because of DCMs' foundation in economic and behavioral theory, they provide insights about individual behaviors: explaining why/how decision-makers choose among a set of alternatives. A DCM provides important economic indicators (e.g. elasticity, willingness-to-pay), and can answer "what-if" questions. Since the late 1970s, DCMs have been the predominant approach for travel demand forecasting and scenario analysis, applied almost everywhere to support transportation planning and decision-making.

Machine learning (ML) is often viewed as a data-driven approach that exploits the rich information in large raw data. In contrast to DCMs, ML methods usually require less a priori theories and assumptions. Its primary focus is prediction accuracy rather than interpretability. Since the 2010s, research in Deep Neural Networks (DNN) has achieved remarkable breakthroughs in various domains, including computer vision, natural language understanding, speech recognition and the biomedical field (e.g. Krizhevsky et al. (2012); Hinton et al. (2012); Sutskever et al. (2014)). Supported by big data and increased computational capacity, DNNs have surpassed a variety of

traditional ML methods by a large margin in many complicated tasks, such as image classification, object recognition, language generation and translation. However, the potential of neural networks has not been fully exploited in the context of discrete choice.

A theory-driven DCM requires assumptions about the model structure and specification. If the prior assumptions are close to the truth, a DCM would be the most precise and efficient description of behaviors. When the underlying relationships are complex with nonlinear effects, difficulties in model specification arise. We may lack a good *a priori* knowledge of the true utility form. A parametric function, even with nonlinear terms added, may still be too restricted to reflect the complex nonlinearity. The consequences of model misspecification can be severe. Biased parameter estimates can result in low prediction accuracy and misleading interpretations for policy decisions. Uncertainty in model structure and specification has been a persistent concern for model developers and users. Given the limitations in manual specification, could we utilize a data-driven neural network to unravel the complexity in data and relax *a priori* assumptions? Can we bring the strengths of neural networks to DCMs, in ways that enhance model flexibility, reduce potential bias, and improve predictability?

The strength of a neural network lies in its powerful function approximation ability. Current neural network applications to discrete choice problems focus on prediction. However, the lack of interpretability is a fundamental challenge for this method to be useful for long-term demand forecasts and scenario analysis. A straightforward adoption of a feed-forward neural network for choice prediction suffers from large estimation variance, and can generate unrealistic economic indicators, such as for value of time and elasticities. It cannot provide credible answers to "what-if" questions at the *disaggregated* model level or consumer level. How might we make neural networks produce meaningful and interpretable results that can be used for transportation policy analysis and planning?

The recent success of neural networks is enabled by progress in optimization methods for deep learning. A DNN can include millions of parameters estimated on a

gigantic dataset. Scalability of estimation is a necessity for training DNNs. For complex discrete choice models, such as the latent class choice model (LCCM), estimation is still challenging when there is a large number of latent classes and/or parameters, and on a large dataset. The difficulty in LCCM optimization - non-concave objective function with a lot of local optima, saddle points and flat regions in a high-dimensional parameter space - resembles the challenges in neural network optimization. Neural networks are almost exclusively trained with stochastic gradient descent (SGD) algorithms, well known for scalability to large data. Many enhanced SGD algorithms have been developed and shown to be highly effective for training DNNs. Could we utilize the training machinery of a neural network to estimate an LCCM? Could we make LCCM estimation scalable to large datasets and high-dimensional problems, and more robust under some challenging scenarios?

The objective of this dissertation is to create synergies between theory-driven DCMs and data-driven neural networks, in order to benefit from both approaches. Specifically, I propose to:

1. embed neural networks in DCMs to **improve model flexibility and predictability**;

2. embed DCMs in neural networks to **keep interpretability**; and

3. utilize neural networks for DCM estimation to **improve estimation speed and scalability** with large data and/or complex problem.

1.2 Thesis Contributions

1. Embed neural networks in discrete choice models for flexible specification of taste heterogeneity

I propose two neural-embedded choice models: *TasteNet-MNL* that extends Multinomial Logit Model (MNL); and *nonlinear-LCCM* that extends the Latent Class Choice Model (LCCM). Both models provide a flexible specification of *taste* as a function of individual characteristics. TasteNet-MNL captures *systematic* taste variation by modeling taste parameters directly as nonlinear functions of individual character-

istics by a neural network (TasteNet). Nonlinear-LCCM represents taste by discrete latent classes. A neural network allows for nonlinearity in modeling class membership, which captures both *systematic* and *random* taste variation.

A key idea, as demonstrated by the two cases, is to assign the complex or less known part of the model specification to a neural network (e.g. heterogeneous taste, class membership assignment), and keep the well-understood part (e.g. trade-offs between alternative attributes) parametric.

a) *TasteNet-MNL*

TasteNet, a feed-forward neural network is employed to capture *systematic taste heterogeneity*. It models taste parameters as a flexible function of individual characteristics. Taste parameters generated by TasteNet are further fed into a parametric logit model to formulate choice probabilities. I demonstrate the effectiveness of this integrated model in capturing nonlinear relationships between socioeconomic characteristics and tastes without *a priori* knowledge. On synthetic data, TasteNet-MNL is able to recover the underlying utility specification, while exemplary MNLs and continuous mixed logit models with *misspecified* systematic utility produce large parameter bias and result in lower prediction accuracy. Misspecified MNLs also have large errors in estimated economic indicators (e.g. value of time, and elasticity), while TasteNet-MNL is able to give accurate interpretations. I apply TasteNet-MNL to model mode choice using a publicly available dataset (Swissmetro data) and compare it with several benchmark MNLs. TasteNet-MNL achieves the best out-of-sample prediction accuracy and discovers a broader spectrum of taste variation than the benchmark MNLs with linear utility specifications.

b) *Nonlinear-LCCM*

An LCCM is represented as a neural network. Its class membership model is replaced by a multi-layer perceptron network. The nonlinear class membership model allows for a more flexible mixing distribution, which captures both *systematic* and *random taste heterogeneity*. I apply this extension to model mode choice using Swissmetro data. The nonlinear-LCCM significantly improves out-of-sample prediction accuracy compared to the LCCM with linear utilities in its logit class membership

model.

2. Embed discrete choice models in neural networks to keep interpretability

To enhance model interpretability, I propose two general strategies: imposing special structure and parameter constraints. With these strategies, we can inject expert knowledge about behaviors into a neural network in order to constrain its flexibility and guide it to generate meaningful results.

Both TasteNet-MNL and nonlinear-LCCM can be seen as neural networks with choice models embedded in them. In the case of TasteNet-MNL, the downstream choice model defines the meaning of each output unit of the TasteNet, through the utility definition. The TasteNet must learn to generate taste parameters that can maximize the log-likelihood of the choice model. As for nonlinear-LCCM, the class-specific choice models define meanings of the latent classes, which are essentially one hidden layer in the entire LCCM network.

In addition to the special structure, I impose parameter constraints according to prior knowledge in two ways: using a transformation layer in the neural network (TasteNet-MNL); and adding a violation penalty to the objective function (nonlinear-LCCM).

Using these two strategies, I improve model interpretability compared to a direct employment of a neural network. I show both TasteNet-MNL and nonlinear-LCCM are able to obtain realistic behavioral/economic indicators, such as values of time, choice elasticities and probability at the disaggregate model level and at the individual level. I compare the interpretations given by neural-embedded choice models and their correspondent DCMs on synthetic data and Swissmetro data.

3. Estimate LCCM with neural networks to speed up estimation and improve scalability to large data/complex problems

I demonstrate the benefits of using the training machinery of neural networks to estimate LCCMs, especially for models with a large number of classes/parameters, estimated on a large dataset, or under certain challenging scenarios such as highly unbalanced classes and relatively a small sample size in the minority latent class. I

contrast neural network estimation by Adam, a SGD algorithm, with the standard maximum likelihood by quasi-Newton routine, and with Expectation-Maximization (EM). Synthetic data results show similar accuracy and estimation time when the data size is relatively small and the model has a small number of classes. Traditional estimation approaches scale poorly to a large dataset and a model with a large number of classes/parameters. They tend to encounter the small class vanishing problem; estimation based on stochastic gradient descent is shown to be 70 times faster on a large synthetic dataset, and can achieve more stable and accurate estimates for latent classes with relatively small membership.

This thesis takes an initial step towards developing a framework to combine theory-based and data-driven methods for discrete choice modeling. I highlight the strengths and weaknesses of econometric DCMs and neural networks. The main idea is to take advantage of both: DCMs' rigorous theoretical foundations and rich domain knowledge accumulated over decades; and neural network's function approximation capability and many techniques developed to scale up estimation for big data and high-dimensional optimization. The end goal is to enable flexible model specification, scale up estimation, and maintain interpretability so that the hybrid model can be used to support scenario analysis and discover new knowledge about behaviors from data.

1.3 Outline

The following chapters are organized as follows.

Chapter 2 provides a review of DCM and current neural network applications to discrete choice problems.

Chapter 3 describes the model structure of TasteNet-MNL.

Chapter 4 presents Monte-Carlo experiments for TasteNet-MNL; and compares TasteNet-MNL with benchmarks.

Chapter 5 shows an application of TasteNet-MNL to model mode choice using Swissmetro data.

Chapter 6 describes the formulation of the LCCM as a neural network

Chapter 7 compares LCCM estimation procedures on synthetic data

Chapter 8 describes the nonlinear-LCCM structure and its application to Swissmetro data.

Chapter 9 concludes and discusses the limitations and future work.

Chapter 2

Literature Review

2.1 Discrete Choice Model Framework: A Theory-driven Approach

2.1.1 Random Utility Models and Extensions

Random utility models (RUMs) have been extensively applied in the field of transportation. RUMs are based on random utility maximization decision theory (McFadden, 1973). For detailed discussions of the micro-economic and psychological foundation of RUMs, readers can refer to McFadden (1973), Manski (1977), and Ben-Akiva and Lerman (1985).

Random utility maximization theory postulates that a decision-maker chooses an alternative from a choice set that maximizes his/her utility. The utility of alternative i to decision-maker n is a random variable U_{in} expressed as a sum of the observable (systematic) part V_{in} and the unobserved (random) component ϵ_{in}.

$$U_{in} = V_{in} + \epsilon_{in} \tag{2.1}$$

RUMs are constructed according to the joint distribution of the random error components in the utilities. Initially, most applications used the Multinomial Logit (MNL) model, which assumes that ϵ_{in} are independently and identically distributed

Gumbel across alternatives and individuals with scale 1 and location 0. MNL is limited by its "independence of irrelevant alternatives" (IIA) assumption, which leads to counter-intuitive substitution patterns across alternatives. Another limitation of MNL is the absence of random taste variation.

Researchers have long focused on relaxing the assumptions of traditional RUMs, and characterizing behaviors with flexible model structure and specification. Generalized Extreme Value (GEV) models relax the IIA assumption and allow for flexible substitution patterns. The well known examples of GEV are the nested logit (NL) model (Ben-Akiva, 1973; Williams, 1977; McFadden, 1978) and cross-nested logit (CNL) model. An overview of the GEV structures can be found in Ben-Akiva and Bierlaire (2003) and Train (2009).

Flexible model structures have been developed to capture **random** taste heterogeneity. The most popular class of such models is Mixed Logit, including the Latent Class Choice Model (LCCM) and the continuous mixed logit model. Mixed logit can approximate any random utility model (McFadden and Train, 2000). It relaxes the assumptions in MNL by allowing for random taste variation, unrestricted substitution patterns, and correlations among unobserved factors (Train, 2009).

A hybrid choice model (HCM) framework integrates a variety of models and methods that extend the traditional RUM (Ben-Akiva et al., 2002). HCM integrates modular components, such as latent variable models to account for latent attitudes and perceptions, flexible structures such as GEV and Mixed Logit, and combines revealed preference (RP) and stated preference (SP) data. HCM is able to accommodate a decision protocol that is not random utility maximization (Ben-Akiva et al., 2002).

As the neural-embedded choice models in my thesis focus on modeling taste heterogeneity, I provide a more detailed review of major developments for modeling taste heterogeneity.

2.1.2 Taste Heterogeneity

The treatment of heterogeneity across decision makers is one of the key research topics in choice modelling. Heterogeneity is generally defined as the difference in consumers'

preferences for the same set of product or service attributes (Ben-Akiva et al., 1997). Heterogeneity exists in various aspects of modeling discrete choice behaviors: such as decision protocols, choice sets, tastes/sensitivity to attributes of the alternatives (Ben-Akiva et al., 1997), and information processing strategies (Hess, 2014).In RUMs, taste heterogeneity is represented by different sets of taste parameters and/or the random component of utility (Ben-Akiva et al., 1997).

Systematic/Random Taste Heterogeneity

There are two kinds of taste heterogeneity: systematic heterogeneity and random heterogeneity. **Systematic heterogeneity** is the taste variation that can be explained by observed characteristics of individuals (e.g. income, age) and choice contexts (e.g. trip purpose, weather condition). In a DCM, systematic taste heterogeneity is represented by different sets of taste parameters for choice-makers with different characteristics (Bhat, 2000).

Taste variation can also be **random**. The random effect is captured by the error component of the utility. There are different sources that account for the random variation. First, there can be inherent uncertainty/stochasticity in consumer choice. The same consumer's taste can vary across repeated choice experiments (*intra-person*), although his/her characteristics do not change. Second, unobserved characteristics (individual or contextual) or latent constructs, such as perceptions and attitudes may explain the taste variation across individuals. As these factors are not directly observed, they are captured by the random error component of the utility. Thirdly, errors from a misspecified systematic utility can be a source of random heterogeneity.

Inter-person/Intra-person Heterogeneity

Taste heterogeneity not only occurs at the inter-person level, but also at the intra-person level (Ben-Akiva et al., 2019). There has been a growing interest in modeling both *inter-person* and *intra-person* taste heterogeneity with more flexible structures, such as stacking multiple layers of mixtures (Walker and Li, 2007; Hess and Rose, 2009; Hess and Train, 2011; Becker et al., 2018; Ben-Akiva et al., 2019), with some

for inter-person and others for intra-person.

Models for Taste Heterogeneity

Systematic taste heterogeneity is usually accommodated by including interactions between characteristics and attributes of alternatives in the utility function. Non-parametric methods are also developed to improve the flexibility in systematic utility specification.

For random heterogeneity, the most popular approaches are the *latent class choice model* (LCCM) and the *random coefficient model*. Since both methods' choice probabilities are mixtures of logit probabilities, they are both called **Mixed Logit**: LCCM is a *discrete* mixed logit model; while random coefficient model is a *continuous* mixed logit model. However, in practice, mixed logit often refers to the continuous mixed logit. For the sake of clarity, we use **LCCM** for discrete mixed logit, and **continuous mixed logit** for the random coefficient model.

How to better represent systematic and random heterogeneity has been a thriving research area for many years, with a variety of flexible model structures developed. In the following sections, I provide a review of the mainstream methods for representing taste heterogeneity.

2.1.3 Modeling Random Taste Heterogeneity

Mixed logit models are distinguished by their choice probabilities: an integral or summation of standard logit probabilities over a probability density function of parameters (Train, 2009). In the continuous mixed logit case (Eqn. 2.2), the mixing distribution $f(\beta)$ is continuous, and takes the form of a standard distribution, such as normal or log-normal. $P_{ni}(y|x;\beta)$ is the choice probability, usually logit probability, conditional on the coefficient β. In the LCCM formulation, the mixing distribution is discrete (Eqn. 2.3). The probability of a consumer belonging to a discrete mixture is modeled by the class membership model ($Q(s|z,\gamma)$), which is often a logit probability. The class-specific choice probability is $P_{nis}(y|x;\beta_s)$.

$$P_{ni}(y|x;\Omega) = \int P_{ni}(y|x;\beta) f(\beta|\Omega) d\beta \tag{2.2}$$

$$P_{ni}(y|x) = \sum_{s} P_{nis}(y|x;\beta_s) Q(s|z;\gamma) \tag{2.3}$$

Mixed logit is a flexible structure that can approximate any random utility model (McFadden and Train, 2000). It relaxes the assumptions of MNL by allowing for random taste variation, unrestricted substitution patterns and correlation in unobserved factors over time (Train, 2009). However, finding the mixing distribution is a challenge.

2.1.4.1 Continuous Mixed Logit

The continuous mixed logit model represents taste heterogeneity by a continuous distribution of taste parameters ($f(\beta)$). The first mixed logit applications date back to Boyd and Mellman (1980) and Cardell and Dunbar (1980). The first applications to individual choice are by Train et al. (1987) and Ben-Akiva et al. (1993). For in-depth descriptions of the model structure, readers can refer to McFadden and Train (2000), Hensher and Greene (2003), and Train (2009).

Continuous mixed logit requires *a priori* assumptions about both the systematic part of the utility and the distribution form of the random taste parameters. The majority of applications use normal or log-normal distribution for β (Revelt and Train, 1998). Other distribution forms include triangular and uniform distributions (Hensher and Greene, 2003; Train, 2001), Rayleigh distribution (Siikamaki, 2001), and truncated normal (Revelt, 1999).

Individual characteristics enter the utility function as alternative specific variables or interacted with alternative attributes. For example, cost divided by income allows the value of cost to decline as income increases (Train, 2009). Individual characteristics can also enter the mixing distribution $f(\beta)$. For example, the variance of β can depend on individual characteristics (Bhat, 1998, 2000; Greene et al., 2006).

An equivalent formulation of continuous logit model is the error components logit

(Walker et al., 2007), which has a different interpretation. It captures correlations and heteroscedasticity between utilities of alternatives.

2.1.4.2 Latent Class Choice Model

A latent class choice model represents taste heterogeneity using discrete latent constructs. The latent classes can represent different decision protocols; choice sets; and population segments with varying tastes or sensitivities to attributes of the alternatives (Gopinath, 1995). Unlike the continuous mixed logit, LCCM does not need *a priori* assumptions about the distribution of the random coefficients. The class membership model captures the joint distribution of random coefficients (Hess, 2014).

LCCM dates back to Kamakura and Russell (1989), with important developments by Gupta and Chintagunta (1994), Swait (1994), and Gopinath (1995). The initial model's class membership probability is the same across individuals (Kamakura and Russell, 1989). Later models assign individuals probabilistically to latent classes conditional on individual characteristics (Gupta and Chintagunta, 1994; Gopinath, 1995).

2.1.4.3 Comparisons of LCCM and Continuous Mixed Logit

Studies have analyzed the difference between continuous mixed logit and LCCM from a theoretical perspective (Greene and Hensher, 2003; Hess et al., 2009; Hess, 2014). Researchers have also compared the two based on synthetic and real data application (Andrews et al., 2002; Provencher and Bishop, 2004; Shen, 2009; Keane and Wasi, 2013). Empirical comparisons show mixed results regarding model fit and generalization performance, e.g. Andrews et al. (2002); Provencher and Bishop (2004); Shen (2009), and Keane and Wasi (2013).

From a theoretical point of view, LCCM has several advantages compared to the continuous mixed logit. First, LCCM makes fewer assumptions about the distribution form (Hensher and Greene, 2003; Hess et al., 2009); in the continuous mixed logit model, the distributional assumptions made during model specification can have significant impacts on model results, such as the signs of parameters (see discussions in

Hess et al. (2005)).

Second, the majority of mixed logit models assume no correlation between randomly distributed parameters, with very few exceptions, e.g. Walker (2001). In LCCM, the correlation between taste coefficients is an inherent characteristic of the model structure. Moreover, the correlation varies across individuals as a function of individual characteristics used in the class membership model. In the case of continuous mixed logit, the correlation is constant across respondents without more sophisticated specification (Hess et al., 2009).

Also, in continuous mixed logit, it is less common in practice to explain the unobserved heterogeneity by socio-demographic characteristics, although it is possible (Greene et al., 2006). In contrast, LCCM models this relationship naturally in the class membership model. In continuous mixed logit, the relationship between socio-demographic characteristics and elasticities are not always easily determined; while in LCCM, the elasticities depend directly on the class membership probabilities and thus are a function of individual characteristics (Hess et al., 2009). Also, the intuitive interpretation of variations across classes makes LCCM convenient to analyze the distribution of welfare associated with policy changes (Provencher et al., 2002; Greene and Hensher, 2003).

One shortcoming of LCCM is that the assumption of within-class homogeneity can be too restrictive (Andrews et al., 2002). The latent specification may oversimplify taste variations in the population, especially when a small number of classes is estimated, or if the underlying distribution is continuous within latent classes (Allenby and Rossi, 1998; Wedel et al., 1999). For example, Allenby and Rossi (1998) find that the extent of heterogeneity is greater than that measured by latent class.

2.1.4.4 Combining Discrete and Continuous Representation of Heterogeneity

To allow for more flexibility in the distribution of random parameters, models have been developed to combine discrete and continuous mixtures. Lenk and DeSarbo (2000) model random coefficients in a generalized linear model as a finite mixture of

multivariate normal distribution. They develop a hierarchical Bayes procedure to estimate the model. Walker and Li (2007) bring continuous variation into a latent class model through error component terms that capture the correlation across alternatives and across choices by the same decision maker. Hess and Rose (2009) combine latent class and continuous mixed logit to account for intra-person taste variation across repeated choices. Following the work of Lenk and DeSarbo (2000), Bujosa et al. (2010) and Greene and Hensher (2013) propose a latent class mixed logit model structure: latent classes are decided by socioeconomic characteristics; within each latent class, taste parameters are randomly distributed among individuals (within-class heterogeneity).

Hess and Train (2011) develop a model with double layers of continuous mixed logit to represent inter- and intra-person heterogeneity. Becker et al. (2018) implement a Hierarchical Bayes estimator for logit mixtures with inter- and intra-level heterogeneity, which improves the estimation speed and is useful for online parameter updates.

2.1.4.5 Non-parametric Mixing Distribution

Although McFadden and Train (2000) have shown that a mixed logit model can approximate any random utility model to any degree of accuracy, finding such a mixing distribution is a main challenge. Parametric distributions have pre-defined distribution forms with a fixed number of parameters. Each distribution has some limitations (Hess et al., 2005; Train, 2008). They are also restricted by their functional forms in the shape of distributions they can represent (Vij and Krueger, 2017). In practice, researchers need to test different distributions. The most appropriate distribution is determined by goodness-of-fit and behavioral interpretation (Vij and Krueger, 2017).

Nonparametric methods have been developed to offer the flexibility of not being constrained by distributional assumptions (Bajari et al., 2007; Fosgerau and Hess, 2007; Train, 2008). An important property of nonparametric mixing distribution is that the number of parameters increases with sample size and the complexity of the approximating distribution increases with the number of parameters (Train, 2008).

The finite mixture version of LCCM with mass points and frequencies treated as parameters is an example of the nonparametric approach (Train, 2008).

Fosgerau and Hess (2007) propose two nonparametric methods: a discrete mixture of normal distributions with the number of normal distributions rising with sample size; and adding series expansion to a continuous base distribution.

Bajari et al. (2007) use a discrete distribution with *fixed* mass points and estimated frequencies at each point, where the number of points rises with sample size. A major limitation of this method is that model performance varies depending on the predetermined location of the mass points. Possible solutions proposed by Train (2008, 2016) require exogenous determination of the mass point locations, although prior information can be collected from more restrictive frameworks.

Dong and Koppelman (2014) propose an endogenous estimation of the support point location and probability mass. Vij and Krueger (2017) implemments an EM algorithm that allows this approach to work for high-dimensional parameter space and a large number of mass points. The support of the distribution is a grid with equal or unequal intervals between successive points.

The flexibility of nonparametric methods originates from utilizing an increasing number of parameters as sample size grows. The benefit comes with computational challenges (Train, 2008), which often prohibit estimating models with a high degree of complexity (Vij and Krueger, 2017).

2.1.4 Modeling Systematic Taste Heterogeneity

Compared to the majority of research on random heterogeneity, less effort has focused on improving the flexibility of systematic utility specification.

Evidence of nonlinear effect in utility

Nonlinear effects of explanatory variables on utility function have been observed in numerous studies of consumer choice (Schindler et al., 2007). For example, in a cognitive study Monroe (1973) observes a price insensitive region (latitude of acceptance)

and *threshold effects* for prices: utility is affected only if price change exceeds a certain threshold. In an experimental study, Gupta and Cooper (1992) find both this threshold effect and a *saturation effect* that beyond a certain range, attribute change has very little or no effect on the behavior of consumers. Kalyanaram and Little (1994) observe the latitude of acceptance and asymmetric response toward price increase and decrease in a brand choice study. They use a piecewise linear function to model the nonlinear effect.

Nonlinear effect finds theoretical explanations in prospect theory (Kahnemann and Tversky, 1979) and the assimilation-contrast theory. Prospect theory proposes asymmetric effects: consumers are more sensitive to losses than to gains (Kahnemann and Tversky, 1979; Winer, 1986, 1988). Assimilation-contrast theory postulates that consumers are insensitive to price change within the range of acceptable prices covering the reference price (Winer, 1988). Some empirical studies support assimilation-contrast theory (Kalyanaram and Little, 1994). Abe (1998, 1999) find evidence that supports both theories.

Specifications of nonlinear systematic utility

Parametric approach

Parametric functions commonly used for nonlinear effects include higher-order polynomial (e.g. Pedrick and Zufryden (1991)), fixed transformation (e.g., semilog in Krishnamurthi and Raj (1988)), piece-wise linear functions (Ben-Akiva and Lerman, 1985; Kalyanaram and Little, 1994; Wedel and Leeflang, 1998).

These approaches require a priori knowledge of the function form. In the piecewise linear case, the number of linear pieces and the boundaries between them need to be specified a priori or searched extensively (Kneib et al., 2007). The parametric specification of nonlinear effect are most efficient if the function form assumed is true. If the assumed function forms differ from the truth, estimation provides inconsistent solutions (Kneib et al., 2007). In most applications, the selection of the nonlinear transformation is carried out by trial-and-error based on model fit and the researcher's judgment/interpretation. Finding the correct specification is still a time-consuming

task (Abe, 1999).

Nonparametric approach

Hastie and Tibshirani (1986, 1987) introduce the class of *generalized additive model* (GAM) that replaces a linear function with a sum of smooth functions. They use a local scoring algorithm to estimate the smoothing function nonparametrically. They demonstrate the model's ability to uncover nonlinear covariate effects for linear regression and logistic regression.

Abe (1998, 1999) adapts GAM to multinomial response (GAM-MNL). The linear-in-parameter utility function is replaced by a sum of one-dimensional nonparametric functions of the explanatory variables. The simulation results show this method can recover underlying nonlinearity of various shapes. This data-driven approach can reduce subjective influence on the results and serve as an exploratory tool for conventional parametric specification (Abe, 1999). Although GAM does not require an a priori assumption of the functional form, the smoothness of each function needs to be decided. The optimal model needs to be selected by hyper-parameter search. Kneib et al. (2007) provides an extension to GAM-MNL, using penalized splines to model the smooth effects of continuous covariates.

A major limitation of GAM-MNL arises from its additive separability assumption. The smoothing function is usually one-dimensional, and rarely goes beyond two dimensions. The nonlinear terms are added together, which limits its ability to model complex interactions between covariates (Abe, 1999). In addition, the model requires a fairly large amount of data to achieve reliable estimates. Estimation is unstable or fails to converge when covariates are highly correlated.

Researchers have also explored using neural networks to model nonlinear effect in utility functions. A review of this avenue of research is found in section 2.2.2.

2.1.5 Challenge I: Model Uncertainty

It is widely acknowledged that the validity of a theory-driven model depends on making the correct assumptions about model structure and specification. Model uncertainty remains a persistent concern. Misspecified systematic utility and random

error distribution can lead to biases in parameter estimates, lower prediction accuracy, and invalid interpretation.

Determining what is deterministic and what is random

As Ben-Akiva et al. (2002) point out, it is often difficult to decide where to build up the complexity in the model: in the systematic part or the error component of the utility. Empirical studies constantly face the challenge of deciding which coefficients should be allowed to vary across individuals (Hess, 2014). When the systematic effect is ignored or misspecified, it goes into the error term, and can be mistaken as random taste variation. Compared to the vast majority of research on modeling random heterogeneity, systematic utility receives less attention.

Most applications use relatively simple systematic utility (e.g. linear-in-parameter functions) plus a complex error structure. To the best of my knowledge, it is rare in practice to diagnose whether systematic utility is under-specified. As Ben-Akiva et al. (2002) points out: "a good error is a zero error" and "it is desirable to expand on the systematic term thereby reducing the disturbance term". We need flexible functions that can capture the systematic effect to the maximum extent so that no systematic effect is left to the error term.

Assumptions about the systematic utility

As mentioned in section 2.1.4, there is empirical evidence of nonlinearity in utility. Various studies show the consequences of misspecifying systematic utility (Torres et al., 2011; Bentz and Merunka, 2000; van der Pol et al., 2014). Although various nonlinear functions (e.g. polynomial term, log transform, piecewise linear) can be used, they require the correct assumption about the functional form. If the assumption is wrong, we still end up with misspecification.

Nonparametric methods such as GAM, can automatically learn nonlinear effects. But its flexibility is restricted by the additive separability assumption. It is useful for learning nonlinear effect for *individual covariate* (e.g. attribute), but not suitable for modeling the *interactions* among covariates (taste heterogeneity).

Distributional assumptions about the error components

The continuous mixed logit model requires assumptions about the error term dis-

tribution. The choice of statistical distribution and assumptions about the correlation structure for random parameters are key challenges for analysts (Hess, 2014). The vast majority of applications rely on independent Normal distributions, which may be not appropriate, as discussed by Hess et al. (2005). Train (2008) and Hess et al. (2005) give detailed discussions about the pros and cons of different parametric distributions. Train (2009) highlights the limitations of the mixing distribution that captures variance and correlations in unobserved factors: "there is a natural limit on how much one can learn about things that are not seen".

The distributional assumptions made during model specification can have significant impacts on model results and interpretation (Hess et al., 2005, 2009). For example, when mixed logit yields a proportion of individuals with positive signs for time coefficients, it is unclear whether it reflects the presence of such values in the data, or an artifact of the distribution form chosen (Hess et al., 2005). Interpretation of the results may depend on the selected distribution. Usually the parametric distribution is selected by trial-and-error according to model fit and somewhat subjective judgment.

A data-driven approach can complement theory-based models to detect misspecifications and differentiate systematic and random effects. It can serve as an **exploratory tool** to reveal the complexity that is not well-understood and/or hard to specify manually. For scenario analysis, a data-driven method can provide a second opinion and be compared with theory-based models. It could help analysts evaluate the uncertainty due to model specification.

2.1.6 Challenge II: Estimation of Complex Structures (LCCM)

Mixed Logit model estimation for large data and high-dimensional parameter space is still challenging. The flexibility of continuous mixed logit comes with significant computational costs as the choice probabilities are given by integrals. An increasing number of studies use Bayesian estimation with Markov Chain Monte Carlo (MCMC) (Train, 2009; Becker et al., 2018). Its application to large-scale problems is still limited. Much of the work to date has been on datasets with limited sample sizes

and small sets of random parameters (Hess, 2014).

LCCM Estimation: Methods and Challenges

LCCM estimation can be challenging because the log-likelihood function is not globally concave. Early LCCM estimation was done by direct maximum likelihood estimation (DML) with Newton or quasi-Newton optimization algorithms (Kamakura and Russell, 1989; Gupta and Chintagunta, 1994). This method has been found to be computationally unstable and often fails to converge (Bhat, 1997). This is partly because Newton and quasi-Newton algorithms can get stuck at some point far from the global optimal, since they rely on quadratic approximations that may be valid only near the local optimal. Therefore, the starting values are essential for such algorithms to converge. In an empirical study, Bhat (1997) shows that DML with a quasi-Newton algorithm DFP (Davidon–Fletcher–Powell) frequently fails to converge for an LCCM with 2 or 3 latent classes, and never converges for 4 latent classes.

An alternative to DML is the Expectation-Maximization (EM) algorithm developed by Dempster et al. (1977). EM is an iterative procedure to maximize likelihood function. Each iteration of the algorithm consists of an expectation (E) step followed by a maximization (M) step. EM is convenient if under the "complete data" scenario, maximum likelihood estimation is easy to perform. In the LCCM case, incomplete data is the latent class label and the complete data distribution belongs to the exponential family, which is easy to maximize. A fundamental property of EM is that it monotonically increases log-likelihood (Dempster et al., 1977). However, the global optimum is not guaranteed, and even a local optimum requires further assumptions (see Gopinath (1995) for details).

EM has been widely applied to estimate models with latent variables or missing data. EM was first introduced to estimate latent class logit models by Gopinath (1995)'s dissertation (page 295-299). Bhat (1997) and Train (2008) show how to implement EM for a latent class logit model and for choice models with non-parametric mixing distributions. EM has shown to be more stable than DML; it also increases the log-likelihood function more than quadratic maximization routines when parame-

ters are far from optimal. However, in the neighborhood of a solution, EM converges slowly. (Bhat, 1997). Because of its good initial performance, Bhat (1997) adopts a hybrid algorithm that starts with EM but switches to DML after a few iterations. This hybrid procedure achieves faster, more stable convergence. Train (2008) implements EM estimation for discrete choice models with three types of non-parametric mixing distribution. EM is computationally attractive.

In practice, most software packages for LCCM estimation uses either DML (e.g. *PandasBiogeme*, R *gmnl* library) or EM (python *lccm* package, R *flexmix* library, R *poLCA* library). The most common choice of optimization algorithms for both methods are quasi-Newton algorithms BFGS or L-BFGS-B. For a detailed description of open-source packages available for mixture model estimation, readers can refer to Sarrias and Daziano (2017).

Although EM achieves more stable convergence, estimating more complex LCCMs with a large number of classes and parameters is still challenging in terms of estimation accuracy and computation time. The difficulty comes from at least two aspects. First, in a high-dimensional space, the objective function is very complex with many local optimal and saddle points. Estimation results are sensitive to initial parameter values and can be unstable. A review by Hess (2014) notes some common estimation problems with a large number of classes, such as very small class probabilities, parameters collapsing to the same values across classes, insignificant coefficients, and lack of interpretability. Experience indicates that with a large number of classes, coefficients often have abnormally large magnitudes and standard errors, especially for classes with small probabilities. When this problem occurs with real data, it is often uncertain whether it is a model identification issue or an estimation problem (e.g. bad initial parameters). While we can try different sets of initial parameters, each model run can take a significant amount of time and the results may still not be unreasonable.

Second, estimation with quasi-Newton routines is computationally heavy. Since quasi-Newton algorithms need to estimate and store second-order derivatives (Hessian), computation time and memory consumption grows quadratically ($O(K^2)$) as

the number of parameters K increases. Also, quasi-Newton methods have to use a large batch of data to estimate Hessian accurately. In practice, all training examples are processed at each iteration, which leads to poor scalability to large dataset. To the best of my knowledge, few studies compare alternative estimation approaches for LCCM. Improving LCCM estimation for complex structures is a key research avenue, as noted by Hess (2014).

2.2 Neural Network for Discrete Choice: A Data-Driven Approach

I provide an overview of the strengths and weaknesses of neural network (NN) compared to DCM in three aspects: model predictability, nonlinear utility specification, and interpretability. I show how NN and DCM are both irreplaceable and, in fact, well-complement each other. Recent studies start to benefit from both through a hybrid of neural network and conventional choice model. I provide a detailed look into this new avenue, the inspirations from the precedents, and the novelty of our approach.

2.2.1 Neural Network for Choice Prediction

Many empirical studies have shown better prediction performance of NNs than DCMs for various choice contexts, such as travel mode choice (Hensher and Ton, 2000; Cantarella and de Luca, 2005; Nam et al., 2017; Lee et al., 2018; Zhao et al., 2018), vehicle ownership choice (Mohammadian and Miller, 2002), and brand choice (Agrawal and Schorling, 1996; Bentz and Merunka, 2000; Hruschka et al., 2002, 2004).

Various DCM structures have been compared with NNs, including logit (Agrawal and Schorling, 1996; West et al., 1997; Omrani, 2015; Lee et al., 2018), nested logit (Hensher and Ton, 2000; Mohammadian and Miller, 2002; Cantarella and de Luca, 2005), cross-nested logit (Cantarella and de Luca, 2005), and mixed logit (Zhao et al., 2018).

Feedforward Neural Network (FFN), also known as Multilayer Perceptron (MLP), is the most popular architecture for model comparison. In the early days, a FFN with one or two hidden layers was often chosen, as more hidden layers suffer from an over-fitting problem. Nevertheless, a shallow FFN achieves better out-of-sample prediction performance than DCMs in nearly all cases.

Since the 2010s, many breakthroughs in deep learning (DL) have made deep neural network (DNN) the state-of-art for various applications in computer vision, speech recognition, and natural language. This triggers the interest in comparing DNNs with DCMs. A big part of DL success is attributed to new techniques for training deep architecture, categorized as regularization methods (e.g. parameter norm penalty, drop-out) and optimization algorithms (e.g. stochastic gradient descent, momentum, initialization strategies, learning rate). These methods help train deep architecture more effectively without over-fitting.

A couple of recent studies attempt DL techniques for choice modeling (Nam et al., 2017; Wang and Zhao, 2018) apply some DL techniques (drop-out, initialization, SGD) to train an MLP with 4 hidden layers. They call it "DNN" because it applies DL techniques. It is compared with conventional[1] MLPs with 4 and 1 hidden layer(s), nested logit and cross-nested logit for a mode choice application with Swissmetro data. Surprisingly, their DNN model gives almost the same performance as nested-logit and cross-nested logit with respect to the log-likelihood of hold-out data. The conventional MLPs seem to be the worst. But we suspect that the best hyper-parameter, such as hidden layer size, is not reached for either the MLPs or DNN, given no hyper-parameter search. It is inconclusive which type of model predicts better.

Another study (Wang and Zhao, 2019) compares a DNN with nested logit to combine revealed and stated preference data for mode choice. Despite an extensive hyper-parameter search, the final best DNN structure with weak regularization is inferior to a nested logit model. The authors highlight the importance of hyper-parameters for DNN to be as good as, if not worse than, DCMs.

To summarize, there is a clear win of shallow FFN vs DCM in terms of prediction

[1] no DL techniques

performance. Based on the limited evidence, DNNs have not worked effectively for discrete choice, perhaps due to small data, over-fitting, or the difficulty in finding hyper-parameters. It seems that simply increasing hidden layers cannot take us far. We need a better **regularization strategy** or **model architecture**. In this study, we find that by **incorporating expert knowledge to neural networks**, we can boost prediction performance by a large margin using one hidden layer.

2.2.2 Neural Network for Learning Nonlinear Utility Function

The outstanding predictability of neural network for discrete choice is often attributed to its ability to learn nonlinear utility functions, e.g. (Shmueli et al., 1996; West et al., 1997; Bentz and Merunka, 2000). Feed-forward neural networks are known as universal function approximators (Cybenko, 1989; Hornik, 1991): a feed-forward network with a single hidden layer and an activation function under mild assumptions can approximate any continuous functions.

While most studies have focused on comparing prediction performance with a brief explanation of why, a few dig into how and under what circumstances (West et al., 1997; De Carvalho et al., 1998; Bentz and Merunka, 2000). Perhaps more interestingly, these studies seek to understand from a behavioral perspective: whether a NN can discover the true behaviors, which can be different from or more complex than we assume; and if so, how to derive such knowledge from a NN.

A series of studies conduct Monte-Carlo experiments to show a NN can capture nonlinearity in utility functions (West et al., 1997; De Carvalho et al., 1998; Bentz and Merunka, 2000). Some nonlinear relationships between input and utility reflects the saturation effect or threshold effect of attributes on utility; some capture non-compensatory decision rules. West et al. (1997) simulate 3 choice scenarios: two non-compensatory decision rules with attribute thresholds (Satisfying Rule and Latitude of Acceptance Rule), and one compensatory decision rule (Weighted-Additive Rule). They find that NN models consistently outperform logit and discriminative analysis when predicting the outcome of non-compensatory choice rule for both training and test data. With synthetic data, De Carvalho et al. (1998) find that if a logit

assumption is broken, for example, with the time coefficient following a log-normal distribution, then NN predicts better than logit. These studies shed light on how NNs achieve better predictability. However they do not provide a way to extract behavior knowledge from NN models.

Bentz and Merunka (2000) show the analogy between NN and MNL, and NN with hidden layers as a more general version of MNL. With synthetic data and an empirical study, they show that NN can detect interaction and threshold effects in utility, and therefore can be used as a diagnostic tool to improve MNL utility specification. This sequential approach requires manual analysis of NN results to identify the nonlinear effect, and thus applies only to simple problems. Yet their idea of a hybrid approach inspired recent development of integrated models.

Hruschka et al. (2002) compare NN with MNL and a Latent Class Logit (LCL) model in an empirical study of brand choice. They find the NN model can identify interaction effects, threshold effects, saturation effects and other nonlinear forms (like inverse S-shape) of attributes on brand utility. Also, NN implies elasticities different from MNL or LCL. MNL sometimes gives wrong signs for elasticity due to its simplistic linear form. The NN predicts better on hold-out data than MNL or LCL. A follow-up study by Hruschka et al. (2004) compares NN with two other MNLs with flexible systematic utility, which draws similar conclusions.

To summarize, these studies explain why a NN can outperform a MNL, in particular when the nonlinear effects of attributes on utility are mistaken. However, these studies have not addressed misspecification of taste heterogeneity, nor compared NN with more advanced DCM structures, such as Mixed Logit. They consider NN as either an alternative to MNL, or a diagnostic tool to improve utility specification for MNL, which works only for simple cases.

2.2.3 Challenge: Lacking Interpretability

Being able to predict better and capture nonlinear utility is not enough for policy-scenario analysis, which is a great advantage of behavior models based on theory and prior knowledge. A major criticism of neural networks is the lack of *interpretability*.

As this popular term is not clearly defined in the literature, despite its wide usage, I summarize the popular understandings of interpretability into the following aspects.

The first is parameter-level interpretability. Clearly, individual weights from a neural network do not carry specific meanings (Agrawal and Schorling, 1996; Shmueli et al., 1996). In contrast, parameters of logit models can be directly interpreted as the marginal effect of an attribute (or "taste"). Another strict definition has to do with how a model is specified: based on external knowledge or learned from data. Statistical choice models clearly map the relationship between input and output with a theory behind it. In neural networks, the relationships are learned from data by arbitrary functions. Even if an NN mimics the true functions and achieves high prediction accuracy, this itself does not provide a theory of why inputs lead to choice outcomes. By either of the first two criteria, a NN model is not interpretable. However, these two definitions are not meaningful measures for model usability. Another view of interpretability by (Sifringer et al., 2018) is "the ability of the model to recover the true parameters' values of the variables that enter the interpretable part of the utility functions". This definition focuses on obtaining unbiased model estimates for the interpretable part. However, the unknown part of the utility modeled by a black-box can still give uninterpretable answers to "what-if" questions.

Perhaps the most popular view of interpretability is the model's ability to derive behavior indicators, such as elasticity, willingness-to-pay (WTP), marginal rate of substitution (MRS) and consumer surplus (CS), regardless of the blackbox nature. Studies that claim ML or NN model interpretability are mostly based on this criteria (Wang and Zhao, 2018; Sifringer et al., 2018; Zhao et al., 2018). Extracting behavior indicators from neural networks is simple. Bentz and Merunka (2000) show the similarity between MNL and a feed-forward neural network with no hidden layer and Softmax activation. Utilities in a NN model correspond to the output units before Softmax activation. Therefore, we can plot utility versus input and obtain marginal effect (Bentz and Merunka, 2000; Hruschka et al., 2004). Choice elasticities and other economic indicators can be computed analytically, since choice probabilities can be

obtained from the NN; and the derivative of a probability (or utility) regarding each input can be computed by applying the chain rule (Hruschka et al., 2002, 2004). These economic indicators can also obtained numerically through simulations (Wang and Zhao, 2018; Zhao et al., 2018) apply variable importance and partial dependence plots, and compute arc elasticities and marginal effects to compare NN with MNL and Mixed Logit.

We find this definition unsatisfying because a model that gives unreasonable behavioral indicators is not interpretable. A study by Wang and Zhao (2018) shows that individual NN estimation can generate unreasonable economic indicators. For example, a choice probability can be non-monotonically decreasing as cost increases and highly sensitive to a particular model run. The derivative of choice probabilities with respect to cost and time can be positive; and values of time can be negative, zero, arbitrarily large, or infinite. They conclude that neural based choice models generate reasonable economic information only at the aggregate level either through model ensemble or population average, due to the challenge of irregular probability fields and large estimation errors. This suggests that a particular NN model may not give unreasonable answers to what-if questions. If individual models can be far off, to what degree can we trust the ensembled results, or how many models do we need to obtain credible answers? Other studies (Hruschka et al., 2002, 2004; Zhao et al., 2018) find that NN gives different behavior interpretations compared to MNL or Mixed Logit.

We acknowledge the definition of interpretability is to some extent subjective and ultimately a philosophical question. We propose a definition close to the popular view but with extra conditions:

A model is interpretable if at a disaggregate level, it is able to give credible answers to "what will happen if" and "but for" questions.

Compared to the second definition, we emphasize the **credibility** of the economic indicators and interpretability at **disaggregated** (both model and choice-maker) level. By "credible", we mean the answer should conform with a set of prior knowledge. Basic priors should include universally acknowledged, such as non-positive

choice elasticity regarding cost and non-positive values of time. Priors reflect expert knowledge about a particular application and may change over time. More specific priors should be used with caution, since they may contradict the truth. **Disaggregated** level requires an interpretable model to learn meaningful relationship between input and output, not only fitting the data well.

A fundamental interpretability challenge for a neural network is that many networks may exist that fit the data equally well; but not all can draw reasonable behavior insights. We show our proposed model reduces estimation errors and obtains reasonable economic indicators at the disaggregated level. Also, we show predictability does not necessarily come at the cost of interpretability. We show that by integrating DCM with NN and imposing prior knowledge, a model can achieve similar or even better prediction performance while keeping interpretability.

2.3 Creating Synergy between Theory-based and Data-driven Models: A Hybrid Approach

Recent studies attempt to create a synergy between statistical DCM and NN through a hybrid structure. We call it the neural embeddded choice model. As far as we know, Learning-MNL (L-MNL) proposed by Sifringer et al. (2018) is the first of this kind.

The idea of a hybrid approach dates back to Bentz and Merunka (2000). They propose using NN as a diagnostic tool to detect nonlinear effects. They show that plotting utility against input variables can help identify nonlinear effect and add it to logit model utilities. The main drawbacks of this approach are the sequential nature and its ineffectiveness for large problems.

The L-MNL proposed by Sifringer et al. (2018) is the closest to our model. They divide the systematic utility into an "interpretable" part manually specified; and an "unknown" part, a nonlinear representation learned by a neural network. The NN is used to capture the effects of features not used in the interpretable part. This model structure is inspiring but with some limitations.

First, it is hard to justify keeping the set of variables in the interpretable part exclusive from the variable set in the representation part. The motive for this division is to make the interpretable utility have stable estimates, as NN can overpower the logit model and cause unstable estimates. The authors note that when the two sets of variables are correlated, estimated coefficients for the interpretable utility become unstable. Because the two parts of utility are *added* without overlapping variables, this model assumes that variables in the representation part have no interaction with those in the interpretable part. This can be unrealistic. Also the selection of variables to enter which part is arbitrary.

Compared to MNL, the gain of L-MNL comes from a flexible representation of the alternative specific constants (ASCs): L-MNL models the ASCs by a neural network as a flexible function of all the unused features. This assumption is too restrictive since the unused features can affect not only ASCs, but also other taste parameters in the interpretable utility (e.g. coefficient for time); also, features in the interpretable utility may have nonlinear effects that are not captured.

Inspired by their work, I propose a more general framework to model taste heterogeneity. The proposed TasteNet-MNL differs from L-MNL and traditional FFW in three aspects. First, we relax the restrictions in L-MNL with a more general framework. We allow all or a subset of taste parameters to be modeled by NN as a flexible function, not just the ASCs. This enhances the flexibility to model taste heterogeneity. Second, we impose constraints on taste parameters obtained by neural networks, as a strategy to regularize the network and obtain interpretable results. Third, we model taste parameters, instead of utility by a NN, different from previous approaches. By doing so, we give only the complicated job to a NN, and keep control over the part we have good knowledge about.

The nonlinear-LCCM is another hybrid of neural networks and discrete choice models. A neural network is used to model class membership assignment; while class-specific choice models are parametric and specified by modellers.

The TasteNet-MNL and nonlinear-LCCM address the challenge of uncertainty in model specification (section 2.1.5). They provide flexible tools to model heterogeneous

taste. A neural network based LCCM estimation tackles some of the difficulties in mixture model estimation (section 2.1.6).

Chapter 3

A Neural Embedded Choice Model: TasteNet-MNL

3.1 Introduction

In this Chapter I propose a neural network embedded choice model structure called *TasteNet-MNL*. The *TasteNet* module is a feed-forward neural network that learns taste parameters as flexible functions of individual characteristics. It replaces manual specification of systematic taste variations in a standard logit model. The choice model part is a multinomial logit model (MNL) with transparent utility specification. It takes taste parameters (output of *TasteNet*) as input, together with alternative attributes to compute utility and choice probability for each alternative. The two parts are integrated and estimated by back-propagation.

I create this hybrid structure to take advantage of both NN and DCM. First, by adding NN to DCM, we utilize NN's **flexibility** to capture systematic taste variation automatically. Second, by including an econometric DCM after the NN, I express and **incorporate expert knowledge** about the utility function (e.g. the trade-offs between time and cost) to guide the neural network so that it can learn behaviorally meaningful parameters. Third, I impose **parameter constraints** to ensure the realistic range of taste parameters.

With the **hybrid structure** and **parameter constraints**, TasteNet-MNL is

expected to give more interpretable results and with less variance in estimates than a black-box neural network does. We present the model structure in this chapter. Monte-Carlo results and a case study are shown in the following two chapters.

3.2 Model Structure

Problem Setup Suppose for a choice task, each of N individuals makes a choice from a choice set (C_n). Data $D = \{(x_n, z_n, y_n)\}_{n=1:N}$ contains N individuals. Observation for person n includes characteristics z_n, attributes of each alternative $x_{nj} \forall j \in C_n$, and the chosen alternative y_n. The goal is to model a person's choice probabilities conditional on characteristics and alternative attributes: $P(y_n = i|x_n, z_n)\ \forall i \in C_n$.

3.2.1 Homogeneous/Heterogeneous Taste in MNL

If tastes are homogeneous across individuals, the systematic utility of alternative i can be specified as a linear combination of attributes (Eqn. 3.1), where β_{ik} is the taste for attribute x_{ik} (alternative i's k-th attribute); and β_{i0} is the alternative-specific constant. βs do not vary across individuals.

$$V_i = \beta_{i0} + \sum_{k=1}^{K_i} \beta_{ik} x_{ik} \tag{3.1}$$

It is more realistic to assume that taste varies across individuals. Utility therefore includes interaction terms between attributes and characteristics. Eqn. 3.2 includes first-order interactions between alternative i's s-th attribute and characteristic z_q for all pairs of (i,s,q) in set I. Interaction effects are usually specified according to prior assumptions and verified by statistical tests. This theory-driven specification can be limited by imperfect knowledge. Also, it is often very difficult, if not impossible, to test all alternative specifications.

$$V_i = \beta_{i0} + \sum_{k=1}^{K} \beta_{ik} x_{ik} + \sum_{(i,s,q)\in I} \gamma_{isq} x_{is} z_q \tag{3.2}$$

3.2.2 TasteNet-MNL

We propose using a neural network to model heterogeneous taste. Instead of **estimating** taste coefficients, as the case in both DCMs and black-box NN, we **predict** coefficients with a neural network.

Let us first consider a simple linear utility V_i as a linear combination of attributes of alternative i. The coefficients are the taste parameters. If we know exactly how the attribute interacts with individual characteristics, we can add the interaction terms as in Eqn. 3.2. Homogeneous taste or known heterogeneous taste parameters are denoted as $\boldsymbol{\beta}^{MNL}$ in Eqn 3.3. For heterogeneous taste with potential nonlinearity that we are uncertain of, we use a neural network to model them, which correspond to $\boldsymbol{\beta}^{TN}$ in Eqn.3.3. The hybrid model therefore consists of two modules: a TasteNet and a MNL. TasteNet is a neural network that maps individual characteristics $\boldsymbol{z}$ to taste coefficients $\boldsymbol{\beta}^{TN}$ (Eqn. 3.4). MNL module uses $\boldsymbol{\beta}^{TN}$ and the corresponding attributes $\boldsymbol{x}^{TN}$, along with the inputs of manually specified part of the utility to compute the final utility and choice probability (Eqn. 3.3).

$$V_i = \boldsymbol{\beta}^{TN}(\boldsymbol{z};\boldsymbol{w})\boldsymbol{x}_{\boldsymbol{i}}^{TN} + \boldsymbol{\beta}^{MNL} f(\boldsymbol{x}_{\boldsymbol{i}}^{MNL}, \boldsymbol{z}) \tag{3.3}$$

$$\boldsymbol{\beta}^{TN} = TasteNet(\boldsymbol{z}, \boldsymbol{w}) \tag{3.4}$$

$$P(y = i|\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\beta}^{MNL}) = \frac{e^{\boldsymbol{\beta}^{TN}(\boldsymbol{z},\boldsymbol{w})\boldsymbol{x}_{\boldsymbol{i}}^{TN} + \boldsymbol{\beta}^{MNL} f(\boldsymbol{x}_{\boldsymbol{i}}^{MNL},\boldsymbol{z})}}{\sum_j e^{\boldsymbol{\beta}^{TN}(\boldsymbol{z},\boldsymbol{w})\boldsymbol{x}_{\boldsymbol{j}}^{TN} + \boldsymbol{\beta}^{MNL} f(\boldsymbol{x}_{\boldsymbol{j}}^{MNL},\boldsymbol{z})}} \tag{3.5}$$

This general formulation has special cases. For example, if we assume all taste parameters are heterogeneous, and heterogeneity is not known *a priori*, then every taste parameter is learned by TasteNet. This is the most general case with no prior assumptions. If we assume homogeneous taste or fully known heterogeneity, the model becomes an MNL. A taste parameter is either modeled by a neural network or manually specified, in other words, no intersection between $\boldsymbol{\beta}^{TN}$ and $\boldsymbol{\beta}^{MNL}$.

Figure 3-1 shows the diagram of an example of TasteNet-MNL. The structure

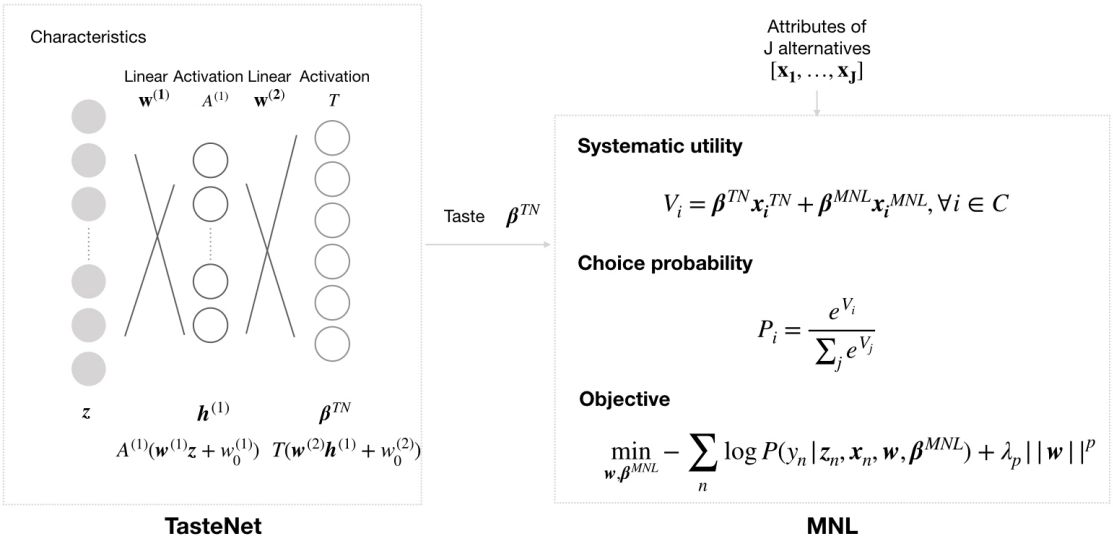


Figure 3-1: Diagram of a TasteNet-MNL (TasteNet as a MLP with 1 hidden layer)

of TasteNet includes a MLP with 1 hidden layer and H hidden units. We apply transformation function T on the output layer. The output of the network is $\boldsymbol{\beta}^{TN}$ further used in MNL utilities.

This neural-embedded MNL achieves two goals. First, the utility function becomes more flexible to represent taste heterogeneity. Second, model interpretability is preserved, since taste parameters as output from the neural network carry explicit meanings (e.g. marginal effect of time), defined by the downstream logit model. These parameters are bounded by constraints according to expert knowledge.

TasteNet

MLP A common type of feed-forward neural network - Multi-Layer Perceptron (MLP) - can be used to model taste function. An MLP consists of an input layer, an output layer, and one or more hidden layers. MLP performs a series of functional transformations.

In our application, an MLP with 1 hidden layer of H hidden units is used. The k-th output of the network β_k^{TN} can be written as the function in Eqn. 3.6, where D is the input dimension, H is the number of hidden units, A is nonlinear activation on hidden units, and T is output transformation function. Non-linearity of this network

comes from the nonlinear activation $A^{(1)}$ and output transform T if T is nonlinear. Neural network parameters $w^{(1)}$ and $w^{(2)}$ correspond to the weights from input to hidden layer and hidden layer to output layer.

$$\beta_k^{TN}(\boldsymbol{z}, \boldsymbol{w}) = T(\sum_{h=1}^{H} w_{kh}^{(2)} A^{(1)}(\sum_{i=1}^{D} w_{hi}^{(1)} z_i + w_{h0}^{(1)}) + w_{k0}^{(2)}) \tag{3.6}$$

A general MLP with L hidden layers can denoted as $\text{MLP}(L, [H_1, .., H_L], [A^{(1)}, ..., A^{(L)}], T)$, where L is the number of hidden layers, H_l is the number of hidden units in the l-th hidden layer. $A^{(l)}$ is the activation function for hidden layer l.

Transformation We impose constraints on taste parameters. A typical constraint is on the sign of the parameters. For example, the coefficient for travel time or waiting time should be negative in most cases. The output activation function T plays the role of incorporating sign constraints on output units.

For βs with non-negative sign constraints, T can be $ReLU(\beta)$, softplus $ln(1 + e^x)$ or exponential function $\exp(\beta)$. For βs with non-positive signs, choices of T can be: rectified linear unit $-ReLU(-\beta)$, $-ln(1 + e^{-x})$ or $-\exp(-\beta)$. For βs without constraints, T is the identity function. Such transformations redistribute the data to the desirable range through continuous differentiable functions.

There are other methods to incorporate prior knowledge. For example, we can add a penalty for violating constraints to the model learning objective. This approach can impose the constraints on training data. Yet on test data, the constraints may not be enforced. An advantage of using transformations for imposing constraints is that the sign will always be kept.

Optimization objective and model training

The maximum likelihood principle is used for model estimation. The objective is to minimize a loss function, which is the sum of the negative log-likelihood and a regularization penalty (Eqn. 3.7). The probability of a chosen alternative is the outcome of MNL (Eqn. 3.5). To prevent the model from over-fitting, we add a regularization

penalty on the p-norm of the neural network weights. It is L2 penalty when p=2; and L1 penalty when p=1. L1 penalty tends to give more sparse parameters, since more weights are shrunk to near zero.

$$\min_{\boldsymbol{w},\boldsymbol{\beta}^{MNL}} -\sum_{n} \log P(y_n|\boldsymbol{z_n},\boldsymbol{x_n},\boldsymbol{w},\boldsymbol{\beta}^{MNL}) + \lambda_p||\boldsymbol{w}||^p \tag{3.7}$$

TasteNet-MNL is trained in an integrated fashion through back-propagation. Unknown parameters to estimate include TasteNet weights $\boldsymbol{w}$ and $\boldsymbol{\beta}^{MNL}$ in the logit model. Note that $\boldsymbol{\beta}^{TN}$ are not parameters to estimate; but the output of TasteNet.

Chapter 4

Monte-Carlo Experiments for TasteNet-MNL

4.1 Introduction

In this chapter, I examine whether TasteNet-MNL is able to recover the true taste function, improve prediction accuracy compared to misspecified MNLs/RCLs, and generate interpretable economic indicators. I compare TasteNet-MNL with benchmarking models (MNLs and continuous mixed logit models) on synthetic data, generated by a MNL model with nonlinear utility functions and known parameters. I first describe the synthetic data generation process (4.2) and compare the models (4.3). Then I present the results of model comparison with respect to: predictability, parameter estimates, and interpretability (4.4).

4.2 Synthetic Data Generation

I create a synthetic choice dataset with higher-order interactions between characteristics and attributes.

The true model The true model is a binary choice model. The systematic utility of each alternative j is a linear combination of attributes x_1 and x_2. The coefficient

(taste) for attribute x_2 is a function of characteristics $\boldsymbol{z}$ with both first and second order interactions among zs (Eqn. 4.1). We create a true model in this form to test if (1) TasteNet-MNL can recover the true model without expert knowledge; and (2) what the consequences would be if a domain expert misses out on some of the interaction terms in the utility function.

$$V_j = a_{0j}+a_1a_1+a_2x_2 = a_{0j}+a_1x_1+(b_0+b_1z_1+b_2z_2+b_3z_3+b_{12}z_1z_2+b_{13}z_1z_3+b_{23}z_2z_3)x_2 \tag{4.1}$$

To make this toy example intuitive, I assign real meanings to each variable. Characteristics include income (inc), full-time employment status ($full$) and flexible work schedule ($flex$). Attributes are travel cost ($cost$) and travel time ($time$) (See Table 4.1 for details). The value of time (VOT) can be directly read as the negative coefficient for variable $time$, since the coefficient for $cost$ is -1. VOTs are the same across alternatives.

Systematic utilities with true coefficients are in Eqn. 4.2 and 4.3. The true values of the coefficients are designed to reflect real behavior. We assume that income has positive effect on VOT, full-time workers have higher VOT and people with flexible schedule have lower VOT. Coefficients for second-order interactions among characteristics are also chosen according to intuition. Random errors in each utility follow Extreme Value distribution.

$$\begin{aligned} V_0 = -cost_0 + (&-0.1 - 0.5inc - 0.1full + 0.05flex \\ &- 0.2inc * full + 0.05inc * flex + 0.1full * flex) * time_0 \end{aligned} \tag{4.2}$$

$$\begin{aligned} V_1 = -0.1 - cost_1 + (&-0.1 - 0.5inc - 0.1full + 0.05flex \\ &- 0.2inc * full + 0.05inc * flex + 0.1full * flex) * time_1 \end{aligned} \tag{4.3}$$

Data generation procedure I first draw input characteristics $\boldsymbol{z}$ according to assumed input distribution in Table 4.1. Alternative attributes $cost$ and $time$ are drawn

Table 4.1: Description of Input Variables in Synthetic Data

		Variable	Description	Distribution
Characteristics	z_1	inc	Income ($ per minute)	LogNormal(log(0.5),0.25) for full-time; LogNormal(log(0.25),0.2) for not full-time
	z_2	full	Full-time worker (1=yes, 0=no)	Bern(0.5)
	z_3	flex	Flexible schedule (1=yes, 0=no)	Bern(0.5)
Attributes	x_1	cost	cost ($)	0.2 to 40$
	x_2	time	travel time (minutes)	1 to 90 minutes

from the ranges described in Table 4.1. With the true model, I compute choice probabilities for each individual. Finally, I draw a chosen alternative for each individual according to the predicted choice probabilities by the true model. I generate 6000, 2000 and 2000 examples for training, development and test data, respectively. Training data are used for model estimation. The development set is used for selecting hyper-parameters. The test set is not used in model training and selection. It reflects model generalization ability.

4.3 Benchmarking Models

I compare 3 models: MNL-I, MNL-II, and TasteNet-MNL against MNL-True (model with the true utility specification); and all models against the ground truth.

MNLs I specify three MNL models. **MNL-I**'s utility functions only include first-order interactions between characteristics and time (Eqn. 4.4). Compared to MNL-I, utilities of **MNL-II** have one additional interaction $inc{*}full{*}time$ (Eqn.4.5). **MNL-TRUE** is a logit model with the true utility specification. It is different from the ground truth since it is estimated on a sample. In all MNLs, alternative specific constants (ASCs) are fixed to 0 for alternative 0 ($ASC_0 = 0$).

Mixed Logit Two random coefficient logit (RCL) models are included to test whether modeling unobserved heterogeneity can compensate specification errors in the systematic utility. We assume time coefficient is randomly distributed, following

a Normal distribution with mean equal to a linear function of characteristics, and standard deviation σ. **RCL-I** and **RCL-II** represent two different specifications of mean function (Eqn. 4.6 and 4.7). RCL-II's mean function is closer to the true model.

$$V_i^{MNL-I} = ASC_i - cost_i + (b_0 + b_1 inc + b_2 full + b_3 flex) * time_i \tag{4.4}$$

$$V_i^{MNL-II} = ASC_i - cost_i + (b_0 + b_1 inc + b_2 full + b_3 flex + b_{12} inc * full) * time_i \tag{4.5}$$

$$U_{ni}^{RCL-I} = ASC_i - cost_{ni} + \beta_n * time_{ni},$$
$$\beta_n \sim N(b_0 + b_1 inc_n + b_2 full_n + b_3 flex_n, \sigma^2) \tag{4.6}$$

$$U_{ni}^{RCL-II} = ASC_i - cost_{ni} + \beta_n * time_{ni},$$
$$\beta_n \sim N(b_0 + b_1 inc_n + b_2 full_n + b_3 flex_n + b_{12} inc * full, \sigma^2) \tag{4.7}$$

TasteNet-MNL The structure of the TasteNet-MNL for this toy example is shown in Figure 4-1. Time coefficient (β_{vot}) is modeled by an MLP. Hyperparameters include the number of hidden layers (L), size of hidden layer(s) ($[H_1, ..., H_L]$), and type of regularizer (norm p), regularization strength λ_p, activation function for hidden units ($[A_1, ..., A_L]$), and output activation transform function (T).

For this synthetic example, we choose 1 hidden layer since the problem is fairly simple. We vary the number of hidden units from 5 to 30. We choose L2 penalty. For each hidden layer size, we apply different strengths of L2 penalty - λ_2 in [0, 0.0001, 0.001, 0.01]. ReLU and Tanh functions are tried as activation on hidden units. For output activation T, we experiment with functions $-ReLU(-\beta)$ and $-e^{-\beta}$ to impose $\beta_{vot} \leq 0$ constraint. We train TasteNet-MNL on training dataset with different combinations of hyper-parameters. For each scenario, we train the model 5 times with different random initialization.

The best model scenario is selected based on the lowest average negative log-likelihood (NLL) on development data. It has 1 hidden layer with 7 hidden units,

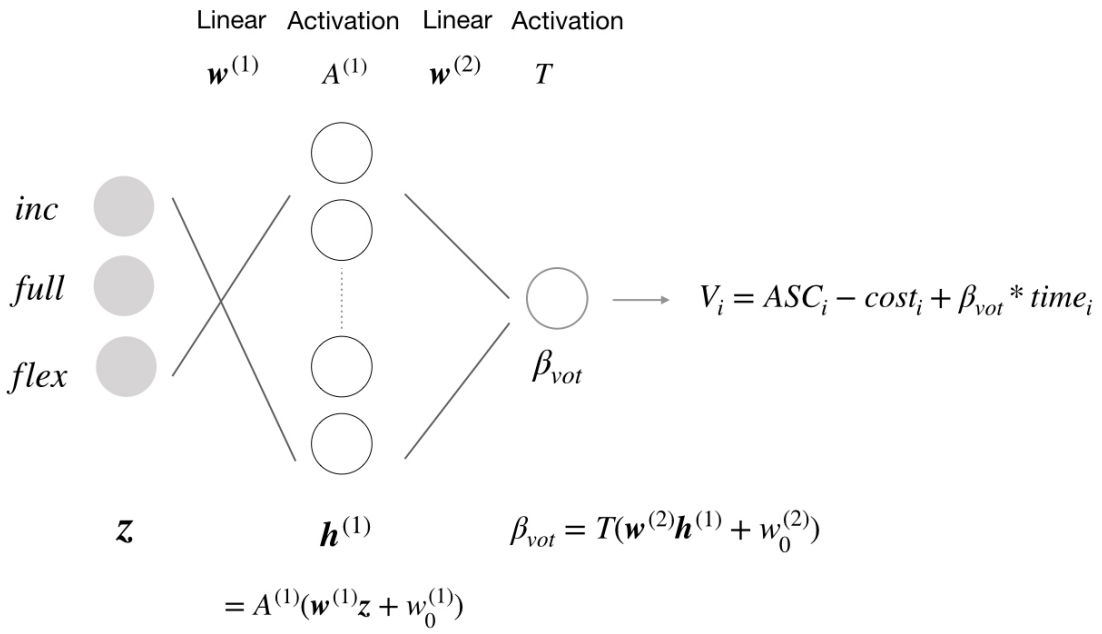


Figure 4-1: Diagram of the TasteNet-MNL for the Synthetic Data

$ReLU$ for hidden layer activation, $-ReLU(-\beta)$ for output transformation, and L2 penalty (λ_2) 0.001.

4.4 Results of Model Comparisons

4.4.1 Predictability: Log-likelihood and Accuracy

Model prediction performance is evaluated by average negative log-likelihood (NLL) and prediction accuracy (ACC) on training, development and test data. Prediction accuracy is measured as the percentage of correctly predicted choices. Table 4.2 summarizes the average NLL and prediction accuracy for synthetic data by different models.

First, MNL with the correct utility specification (MNL-TRUE) achieves the same NLL and ACC as the data generation model. Because of missing higher-order interaction terms, MNL-I and MNL-II result in a greater NLL loss (0.59 - 0.6) than MNL-TRUE (0.47); and lower prediction accuracy (70% - 72%) compared to MNL-

Table 4.2: Average Negative Log-likelihood (NLL) and Prediction Accuracy (ACC) for Synthetic Data

Model	NLL_train	NLL_dev	NLL_test	ACC_train	ACC_dev	ACC_test
MNL-I	0.54102	0.55699	0.54572	0.719	0.703	0.722
MNL-II	0.53755	0.55479	0.54695	0.717	0.706	0.724
RCL-I	0.52591	0.54594	0.52758	0.718	0.703	0.724
RCL-II	0.52298	0.54323	0.52808	0.719	0.701	0.723
TasteNet-MNL (H=7, λ_2=0.001)	0.45433	0.46803	0.46562	0.785	0.775	0.786
MNL-TRUE	0.45459	0.47268	0.45979	0.786	0.773	0.785
Data generation model	0.45502	0.47186	0.45877	0.786	0.772	0.787

TRUE(77% - 79%). Compared to MNL-I, MNL-II's utility includes one more interaction $inc * full * time$, which has the largest effect (-0.2) among the three missing terms in MNL-I. Surprisingly, model fit and prediction accuracy of MNL-II does not improve significantly. This indicates that prediction accuracy can be very sensitive to systematic utility specification. A small mistake can lead to significant prediction errors. Low prediction accuracy can be a sign for model misspecification.

Compared to MNL-I and MNL-II, RCL-I and RCL-II both achieve better fit: smaller NLL loss than misspecified MNLs. The better fit is because part of the missing terms is absorbed and modeled as random taste heterogeneity. Interestingly, however, prediction accuracy is not improved.

The best TasteNet-MNL obtains the same level of prediction accuracy (NLL and ACC). Moreover, this is done without defining every detail of the utility function. We give minimal instructions to the model: 1) utility is a function of time and cost; and 2) tastes for each attribute depend on characteristics. We do not need to specify in detail how characteristics interact with attributes. This is learned by the neural network. But you may still wonder: does TasteNet-MNL recover the true utility function?

4.4.2 Parameter Estimates

To answer this question, models are compared at the parameter level (Table 4.3). For MNLs, the coefficients can be directly retrieved. For TasteNet-MNL, I regress the predicted β_{VOT}s against the 3 input zs (inc, $full$, $flex$) and the interactions among

Table 4.3: Parameter Estimates by MNLs, RCLs and TasteNet-MNL

Coef	MNL-I	MNL-II	RCL-I	RCL-II	TasteNet-MNL[a]	MNL-TRUE	Truth
ASC_1	-0.1484	-0.1085	-0.141	-0.141	-0.1055	-0.1003	-0.1
time	-0.0998	-0.1233	-0.0914	-0.139	-0.1056	-0.0927	-0.1
inc*time	-0.5983	-0.5058	-0.636	-0.447	-0.4829	-0.5277	-0.5
full*time	-0.1154	-0.0651	-0.109	-0.0434	-0.1093	-0.1051	-0.1
flex*time	0.1113	0.1120	0.114	0.115	0.060	0.0458	0.05
inc*full*time		-0.1470		-0.223	-0.1904	-0.1741	-0.2
inc*flex*time					0.0182	0.0695	0.05
full*flex*time					0.1046	0.0932	0.1
σ(time)			0.0528	0.0504			
RMSE	0.093	0.051	0.098	0.058	0.014	0.016	
MAE	0.072	0.042	0.076	0.053	0.012	0.012	
MAPE	63%	52%	64%	61%	15%	11%	

[a] Estimates by regression
[b] RMSE: Root Mean Squared Error; MAE: Mean Absolute Error; MAPE: Mean Absolute Percentage Error

them to obtain the coefficients. Estimated ASC_1 is from the MNL module. I compare the estimated coefficients with the ground truth.

MNL-TRUE has the lowest parameter error. Its mean absolute percentage error (MAPE) is 11%. TasteNet-MNL is close to MNL-TRUE (15%). This means that TasteNet-MNL recovers the true form of the taste function via neural network. MNL-I, MNL-II, RCL-I and RCL-II have large biases in parameter estimates, with 52% to 64% error.

RCL-I and RCL-II both have statistically significant standard deviation for the random coefficient for time ($\sigma(time)$). This indicates that the missing higher-order interactions in the systematic utility are identified as unobserved heterogeneity. RCLs may not reduce the bias in parameter estimates, although they can fit the data better with higher log-likelihoods (see Table 4.2).

These results imply that if we do not have a flexible enough function to model the systematic taste variation, we might mistake systematic heterogeneity for random heterogeneity, and obtain biased estimates and low prediction accuracy. Neural networks can be utilized to exhaust the capacity of the systematic utility, and separate systematic from random effect.

4.4.3 Value of Time

I compute values of time estimated by different models, and compare them against the ground truth (Table 4.4).

MNL-I and MNL-II's error in predicted VOTs is 1.7 $ per hour on average, about 10% of the true values. TasteNet-MNL's average error is much lower: 0.05 $ per hour, 0.3% from the true VOTs. Taste-MNL's accuracy in VOT matches MNL-TRUE.

To test the model's ability to generalize to input from a different distribution, I generate a new dataset with 200 individuals with uniformly distributed characteristics. There are 50 individuals in each of the four groups defined by all combinations of full-time (yes/no) and flexible schedule(yes/no). Income of individuals from each group is evenly distributed in the range of 0 to 60$ per hour with interval size 1.2.

With this new input, MNL-I and MNL-II produce an errors of 2.3$ per hour (14%) and 1.6$ per hour (10%), respectively, compared to TasteNet-MNL's error of 0.3$ per hour (1.6%). TasetNet-MNL provides more accurate VOTs at the individual level.

I plot the predicted VOTs by different models and the ground truth (Figure 4-2). MNL-I cannot distinguish the difference in VOTs (given income is fixed) between the groups with and without full-time jobs and flexible schedules. Adding higher-order interaction in MNL-II helps, but large bias persists. TasteNet-MNL gives more accurate VOT estimates at the individual level. The root mean squared error (RMSE) of individual VOT estimates is 0.41, close to the true model, MNL-TRUE (0.35), and much lower than MNL-I (2.71) and MNL-II (1.71). The mean absolute percentage error (MAPE) by TasteNet-MNL is 1%, similar to MNL-TRUE and better than MNL-I (14%) and MNL-II (9%).

To summarize, TasteNet-MNL is able to predict disaggregate VOTs close to the ground truth, while misspecified MNLs suffer from biased estimates.

4.4.4 Choice Elasticity & Choice Probability

Elasticities are useful economic indicators derived from a choice model. They measure the effects of a change in one of the variables (e.g. income, cost) on the choice

Table 4.4: Errors in Estimated Values of Time (Unit: $ / Hour)

Input data	Error metric[a]	MNL-I	MNL-II	MNL-TRUE	TasteNet-MNL
Synthetic data	RMSE	1.805	1.730	0.098	0.111
	MAE	1.700	1.696	0.080	0.056
	MAPE	10.1%	10.1%	0.5%	0.3%
New input	RMSE	2.710	1.707	0.351	0.408
	MAE	2.274	1.573	0.244	0.280
	MAPE	13.9%	9.9%	1.5%	1.6%

[a] RMSE: Root Mean Squared Error; MAE: Mean Absolute Error; MAPE: Mean Absolute Percentage Error

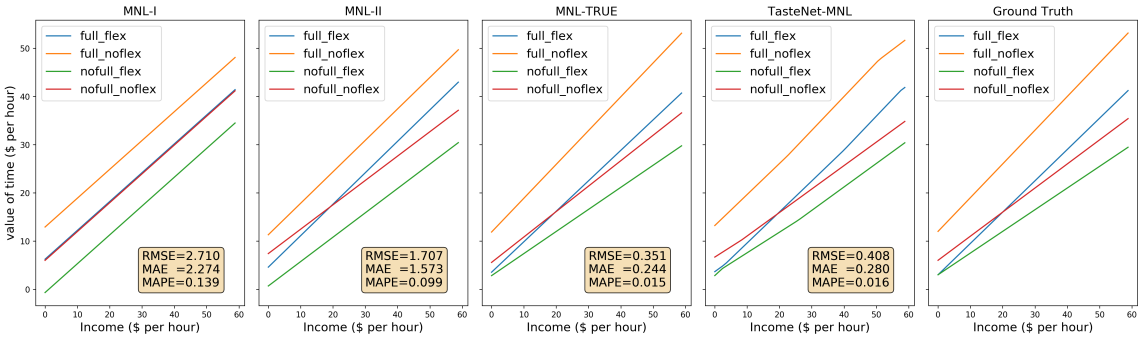


Figure 4-2: Estimated Values of Time and the Ground Truth for New Input

probability. I compare *disaggregated point* elasticities across models. "Disaggregate" refers to the responsiveness of an individual rather than the population. "Point" means the impact of an infinitesimal modification of the respective variable (see Ben-Akiva and Lerman (1985) for details).

The elasticity of demand with respect to alternative attribute x_{ink} is defined in Equation 4.8. $P_n(i)$ is the probability of choosing alternative i for person n. x_{ink} is the k-th attribute of alternative i for person n. Elasticities for a linear MNL and TasteNet-MNL are shown in Eqn. 4.9 and 4.10. Elasticity $E_{x_{ink}}^{P_n(i)}$ measures the percentage change in choice probability $P_n(i)$ with respect to one percentage change in attribute x_{ink}.

$$E_{x_{ink}}^{P_n(i)} = \frac{\partial P_n(i)}{\partial x_{ink}} \frac{x_{ink}}{P_n(i)} \tag{4.8}$$

$$E_{x_{ink}}^{P_n(i)} = (1 - P_n(i)) x_{ink} \beta_k \tag{4.9}$$

$$E_{x_{ink}}^{P_n(i)} = (1 - P_n(i)) x_{ink} \beta_k(z) \tag{4.10}$$

I compare choice elasticity and choice probability of alternative 1 with respect to time of alternative 1.

In the first analysis, I estimate elasticities for each sample in the synthetic data with different models (Eqn. 4.9 and 4.10). I measure the difference between the estimated elasticity and the true elasticity by RMSE, MAE and MAPE. Table 4.5 shows the results. It is clear that TasteNet-MNL achieves the same level of accuracy as the true model MNL-TRUE, while the mis-specified logit models result in 55% to 56% errors. Similar results hold for predicted choice probability.

The second analysis is performed on a selected individual with 60$ hourly wage, full-time job and flexible schedule. Mode 0's time and cost is fixed at 20 minutes and 2$. Cost of mode 1 is fixed to 8$. I vary $time_1$ from 0.2 to 20$, and compute choice elasticity and probability of this person for each value of $time_1$. Figure 4-3 shows the estimated elasticity v.s. $time_1$ and choice probability v.s. $time_1$ across models.

Table 4.5: Errors in Estimated Elasticities and Probabilities by Different Models

	Choice Elasticity				Choice Probability			
Error metric	MNL-I	MNL-II	MNL-TRUE	TasteNet-MNL	MNL-I	MNL-II	MNL-TRUE	TasteNet-MNL
RMSE	5.30	5.22	0.34	0.32	0.21	0.21	0.012	0.011
MAE	3.03	3.10	0.16	0.11	0.16	0.16	0.0079	0.0057
MAPE	55%	56%	3%	2%	61%	62%	3%	2%

RMSE: Root Mean Squared Error; MAE: Mean Absolute Error; MAPE: Mean Absolute Percentage Error

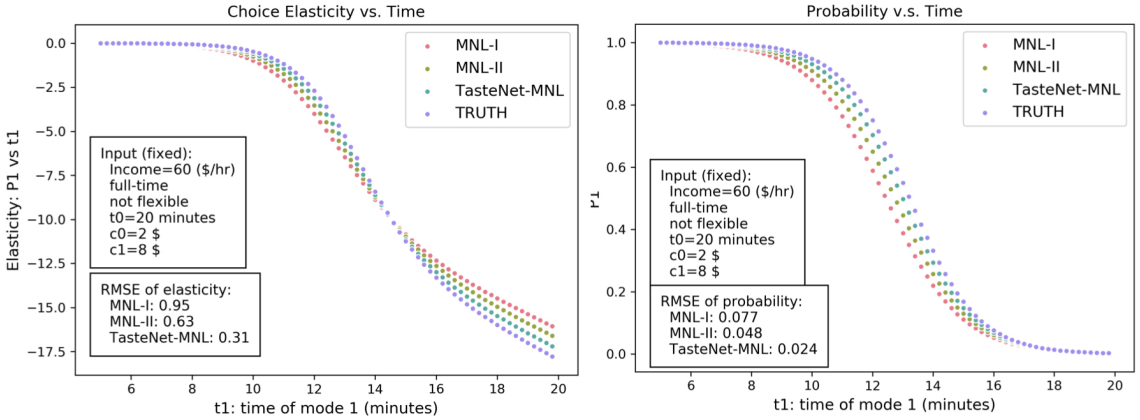


Figure 4-3: Elasticity and Choice Probability against $time_1$ for the Selected Person

Among the models, TasteNet-MNL gives the function closest to the ground-truth.

The third analysis compares predicted elasticities and probabilities by different models across 4 types of individuals. The four types of people are defined by the combinations of full-time (yes/no) and flexible schedule (yes/no) with income fixed at 30$ per hour. Time and cost of alternative 0 is given at 20 minutes and 2 $, and $cost_1$ is 8 $. I plot the elasticity and probability as a function of $time_1$ for each group predicted by different models (Figure 4-4 and Figure 4-5). MNL-I can barely distinguish the difference between the full-flex and nofull-noflex groups; while TasteNet-MNL can distinguish and give more accurate estimates than the misspecified MNLs.

4.5 Understanding the Neural Network

To understand how the neural network learns the effect of input variables and their interactions, we visualize the activation values of the hidden units with simulated

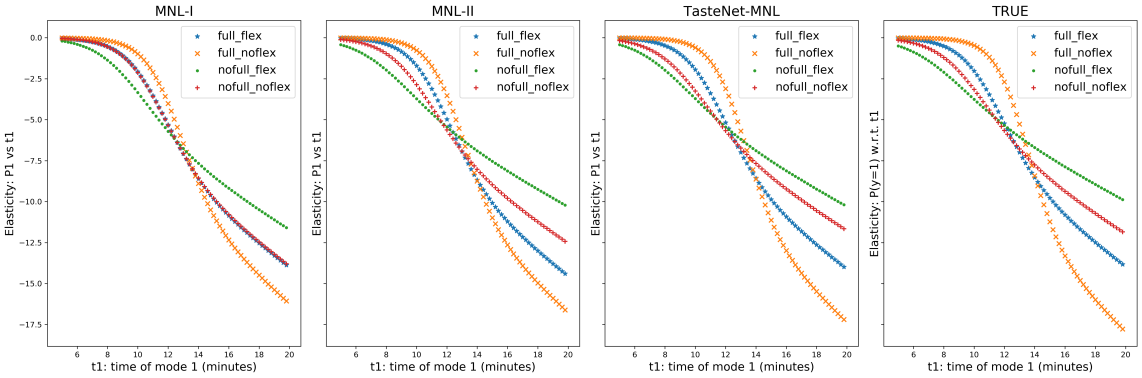


Figure 4-4: Elasticity v.s. $time_1$ for 4 Types of Individuals

(inc=30\$/hr, $time_0$=20 min, $cost_0$=2\$, $cost_1$=8\$)

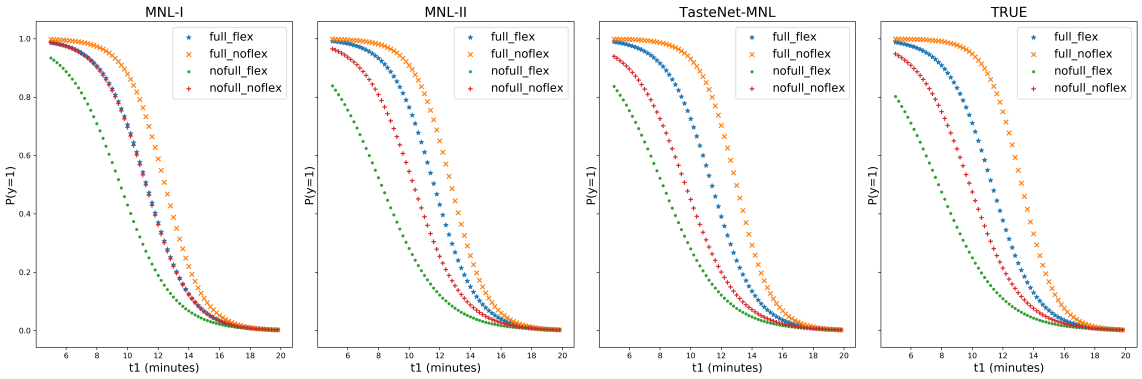


Figure 4-5: Choice Probability v.s. $time_1$ for 4 Types of Individuals

(inc=30\$/hr, $time_0$=20 min, $cost_0$=2\$, $cost_1$=8\$)

individual characteristics. We also show the estimated weights of TasteNet, including weights of the linear layer from input to hidden layer, and weights of the linear layer from hidden layer to output layer (Table 4.6). Interestingly, hidden unit 6 is not used since its associated weights are all zeros.

We generate four types of individuals with income varying from 0 to 60 $ per hour. We pass individual characteristics to the trained TasteNet and obtain activation values for each hidden unit. Figure 4-6 displays the activation values. Darker color indicates stronger activation. All activation values are non-negative since the activation function used is ReLU. By observing how a neuron gets activated as input varies, we can understand the "role" of each neuron in approximating the true taste function.

Hidden units 4 and 5 apparently capture income effect, since they become more activated as income increases in all 4 groups. Hidden units 4 and 5 also capture the non-flexible effect. Note that individuals with non-flexible schedules tend to have higher activation values for hidden units 4 and 5, all else equal (left vs right in Figure 4-6). Note that in this case, higher activation of units 4 and 5 leads to higher values of time (or more negative β_{VOT}). Their corresponding coefficients in the linear hidden-to-output layer are negative (-0.3595 and -0.1944, see Table 4.6). In other words, bigger activation leads to a more negative β_{VOT}. Hidden unit 3 captures the full-time effect. Full-time individuals tend to have a higher activation value for unit 3, which leads to a lower value of time since the hidden-to-output layer's coefficient for unit 3 is negative (-0.3641). Hidden units 1, 2 and 7 represent the three interaction effects: income * full-time, income * not flexible, and not full-time * not flexible, respectively. Again, we see hidden node 6 is never activated.

4.6 Summary

Through Monte-Carlo experiments, I demonstrate TasteNet-MNL's ability to capture nonlinear taste functions and uncover the true utility form. Misspecified systematic utility in MNLs or RCLs can lead to large bias in parameter estimates. TasteNet-MNL

Table 4.6: Estimated Weights of TasteNet-MNL

	Input-Hidden				Hidden-Output
Hidden units	z_1 income	z_2 fulltime	z_3 flexible	z_0 intercept	
1	-0.4257	0.3917	0.0479	-0.0315	0.462
2	0.3907	-0.0925	-0.114	-0.0543	-0.1536
3	-0.0001	0.4637	-0.0764	0.484	-0.3641
4	0.8034	-0.2261	-0.2055	0.4987	-0.3595
5	0.812	-0.317	-0.2252	0.5003	-0.1944
6	0	0	0	0	0
7	0.0396	-0.5551	-0.1817	0.5089	0.4905
intercept					0.0921

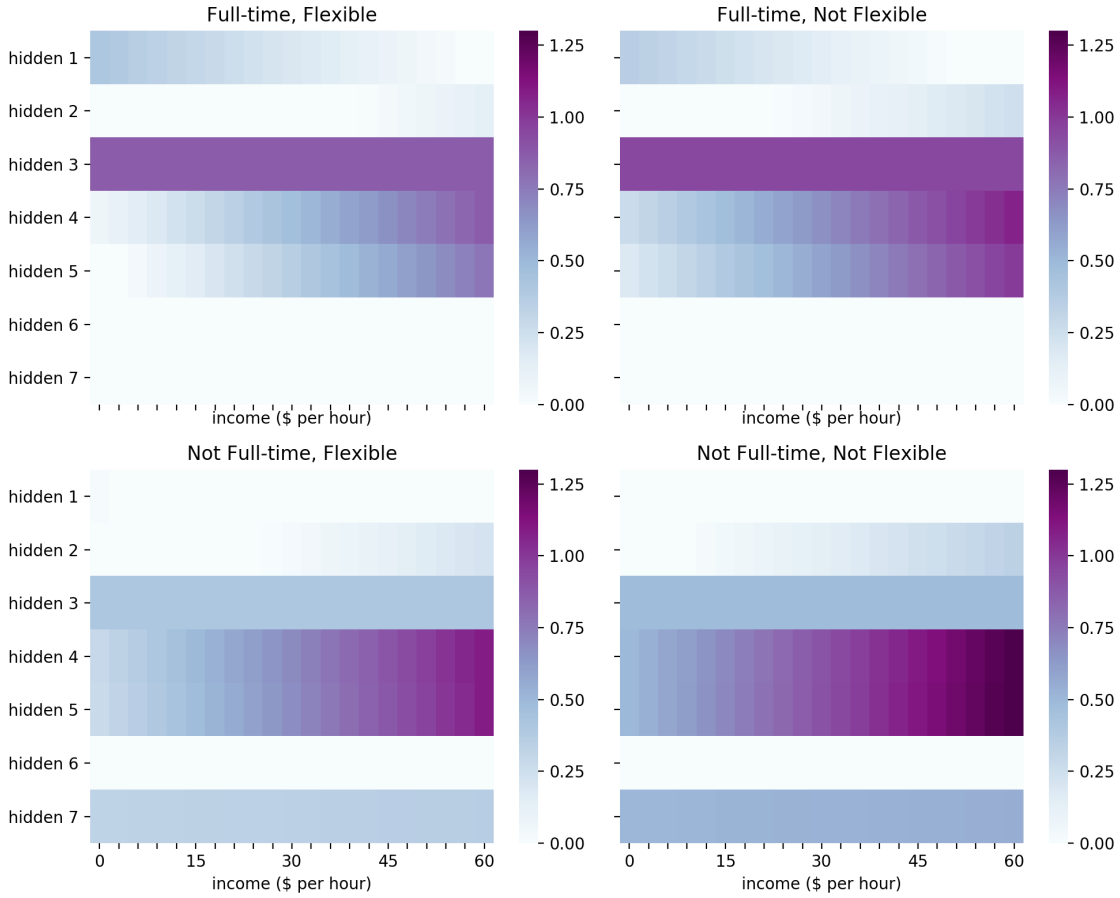


Figure 4-6: Activation of the Hidden Layer in TasteNet

can be used to identify specification errors in utility, and reduce potential biases.

TasteNet-MNL's prediction accuracy matches the true model (77% to 79%), higher than the misspecified MNLs and RCLs (70% to 72%). TasteNet-MNL also provides interpretable economic indicators, like value of time and demand elasticities, close to the ground truth; while MNLs and RCLs with misspecified utility can produce unreliable interpretations.

Chapter 5

TasteNet-MNL Application: Swissmetro Mode Choice

5.1 Introduction

In this chapter, I apply TasteNet-MNL to a publicly available dataset – *Swissmetro* to model mode choice for inter-city travel. The purpose of this application is to 1) examine whether TasteNet-MNL is able to predict more accurately compared to a manually specified, relatively sophisticated MNL; and 2) whether TasteNet-MNL can draw reasonable behavioral interpretations and, if so, how its interpretations differ from those of the MNLs?

To answer these questions, I set up three benchmarking MNL models with increasing complexity in the utility function. I compare TasteNet-MNL with these benchmarks in terms of model fit and prediction accuracy on hold-out datasets. I also examine individual tastes estimated by each model and the interpretations derived from TasteNet-MNL in comparison to the MNL benchmarks.

5.2 Data

The Swissmetro is a proposed revolutionary mag-lev underground system. To assess potential demand, the Swissmetro Stated Preference (SP) survey collected data from

Table 5.1: Description of Variables in the Swissmetro Dataset

Alternative	Alternative attributes	Availability
TRAIN	time, headway, cost (train_tt, train_hw, train_co)	train_av
SM (Swissmetro)	time, headway, seats[a], cost (sm_tt, sm_hw, sm_seats, sm_co)	sm_av
CAR	time, cost (car_tt, car_co)	car_av
Person/Trip variable	**Variable levels**	
AGE	0: age $\leq$ 24, 1: 24 $<$ age$\leq$30, 2: 39 $<$ age $\leq$ 54, 3: 54 $<$ age $\leq$ 65, 4: 65$<$age	
MALE	0: female, 1: male	
INCOME (thousand CHF per year)	0: under 50, 1: between 50 and 100, 2: over 100, 3: unknown	
FIRST (First class traveler)	0: no, 1: yes	
GA (Swiss annual season ticket)	0: no GA, 1: owns a GA	
PURPOSE	0: Commuter, 1: Shopping, 2: Business, 3: Leisure	
WHO (Who pays)	0: self, 1: employer, 2: half-half	
LUGGAGE	0: none, 1: one piece, 2: several pieces	

a. Seats configuration in Swissmetro: seats=1 if airline seats, 0 otherwise.

1,192 respondents (441 rail-based travellers and 751 car users), with 9 choices from each respondent. Each respondent is asked to choose one mode out of a set of alternatives for inter-city travel given the attributes of each mode (e.g. travel time, headway and cost). The universal choice set includes train (TRAIN), Swissmetro (SM), and car (CAR). For individuals without a car, the choice set includes only TRAIN and SM. Table 5.1 provides a description of the variables. For more information, readers can refer to Bierlaire (2018).

The original data has 10,728 observations, downloaded[1] in Jan 2019. After removing observations with unknown age, "other" trip purpose and unknown choice, we retain 10,692 observations. We randomly split the data into training ("train"), development("dev") and test("test") set with 7,484, 1,604 and 1,604 observations, respectively.

[1]Data link: https://biogeme.epfl.ch/data.html

Table 5.2: Estimated Coefficients of MNL-A

Variable Description	Train	Swissmetro	Car
Constant	0.1227	0.5726	
Travel time (minutes)	-1.3376	-1.4011	-1.0177
Headway (minutes)	-0.4509	-0.8171	
Seats (airline seating = 1)		0.1720	
Cost (CHF)	-1 (fixed)	-1 (fixed)	-1 (fixed)
GA (annual ticket = 1)	2.0656	0.5319	
Age			
1: $24 < \text{age} \leq 30$	-0.7548		
2: $39 < \text{age} \leq 54$	-0.9457		
3: $54 < \text{age} \leq 65$	-0.4859		
4: $65 \leq \text{age}$	0.6995		
Luggage			
1:one piece			-0.1538
2:several pieces			-0.9230

5.3 Benchmarks

The three benchmarks are logit models. **MNL-A** is similar to Bierlaire et al.(2001)'s MNL specification but with some enhancements: 1) the value of travel time and value of headway are made mode-specific; 2) all levels of age and luggage categories are included; and 3) cost coefficients are fixed to -1.0 for directly reading VOT from time coefficients (Table 5.2). In the benchmark **MNL-B**, I add the interaction terms: time*age, time*income and time*purpose (Table 5.3). The third benchmark **MNL-C** is a MNL with all pairs of first-order interactions between characteristics and attributes (Table 5.4). This model is equivalent to a TasteNet-MNL with all taste coefficients modeled by a neural network without hidden layers.

5.4 TasteNet-MNL Structure for Swissmetro

The TasteNet-MNL structure for Swissmetro data is shown in Figure 5-1. I specify the utility functions for each alternative in the MNL module. Coefficients for cost are fixed to -1 so that the coefficients for time is the negative value of time. There are 7 coefficients in the MNL utilities, including for the alternative specific constants

Table 5.3: Estimated Coefficients of MNL-B

Variable Description	Train	Swissmetro	Car
Constant	0.0056	0.4674	
Travel time (minutes)	-0.5006	-0.4010	-0.5600
Travel time * Age			
0: age $\leq$ 24			
1: 24 $<$ age $\leq$ 39	-0.6354	-0.3307	-0.5696
2: 39 $<$ age $\leq$ 54	-0.8475	-0.6101	-0.6105
3: 54 $<$ age $\leq$ 65	-0.1566	0.1419	-0.0915
4: 65 $<$ age	0.3265	-0.243	-0.0234
Travel time * Income			
0: under 50			
1: 50 to 100	-0.2688	0.1739	0.1623
2: over 100	-1.0181	-0.436	-0.4093
3: unknown	0.0852	0.2828	-0.0923
Travel time * Purpose			
0: Commute			
1: Shopping	-0.2081	-0.6192	-0.6062
2: Business	-0.1574	-0.8688	-0.1833
3: Leisure	-0.59	-0.9706	-0.0162
Headway (minutes)	-0.6158	-0.7011	
Seats (airline seating = 1)		0.189	
Cost (CHF)	-1 (fixed)	-1 (fixed)	-1 (fixed)
GA (annual ticket = 1)	1.6162	0.2988	
Luggage			
1:one piece			-0.1714
2:several pieces			-0.6718

Table 5.4: Estimated Coefficients of MNL-C

	Coefficients for alternative attributes							
z (characteristics)	TRAIN_TT	SM_TT	CAR_TT	TRAIN_HE	SM_HE	SM_SEATS	TRAIN_ASC	SM_ASC
Intercept	-0.0671	0.1455	0.0059	0.1713	0.0646	0.3064	0.2953	0.2067
Male	-0.1526	-0.0477	0.0742	-0.2384	0.0706	-0.1016	0.0671	0.149
Age								
1: (24,39]	-0.0965	-0.2422	-0.1093	0.0044	0.5682	0.0517	-0.1634	0.4285
2: (39,54]	-0.1467	-0.2022	-0.195	-0.2397	-0.0105	-0.2135	-0.2692	0.0959
3: (54,65]	0.0256	0.1201	0.0251	-0.2379	-0.0807	0.1619	-0.0861	-0.0344
4: (65,)	-0.1712	0.1435	0.1105	0.6032	-0.1488	-0.1529	0.618	-0.351
Income								
1: 50-100	0.0494	-0.039	0.0098	-0.1884	-0.2972	0.2349	-0.1776	0.1944
2: over 100	-0.2825	-0.1697	-0.2662	0.1393	0.0372	0.5288	-0.0406	-0.0789
3: unknown	0.0289	0.1467	-0.2037	0.1484	-0.0721	-0.4196	0.1621	-0.0459
First class	-0.1927	-0.0807	-0.3297	-0.4768	0.1183	0.1302	0.2228	-0.2085
Who pay								
1: employer	-0.2154	-0.1668	0.1231	0.028	-0.0045	0.0882	0.1191	0.3986
2: half-half	0.1537	0.4771	0.4391	-0.0311	0.3917	0.3114	-0.2414	-0.0332
Purpose								
1:Shopping	0.2339	-0.219	0.19	0.1509	0.0493	0.1994	0.4238	0.6996
2:Business	-0.0872	-0.3524	-0.181	-0.0544	-0.0195	-0.0647	0.0605	-0.2941
3:Leisure	-0.2678	-0.2778	-0.0043	0.3245	-0.4552	-0.0289	-0.302	-0.4739
Luggage								
1:one piece	-0.0375	0.0861	0.2525	0.58	-0.1993	0.0413	0.3364	0.3239
2:several pieces	0.022	-0.1785	-0.2731	-0.2946	0.0814	-0.1225	-0.0041	0.2158
Annual ticket	0.5912	-0.0075	-0.3181	0.2652	-0.2032	-0.5815	0.3576	0.1351

(Figure 5-1). I assume all MNL coefficients (taste parameters) are functions of individual characteristics, and model them as the output of the TasteNet. This is a special case of the general structure in Section 3.2.2: the set of $\boldsymbol{\beta}^{MNL}$ is empty and all taste parameters are modeled by TasteNet as $\boldsymbol{\beta}^{TN}$ (Eqn. 3.3 and 3.5).

The TasteNet module consists of a linear layer from input $\boldsymbol{z}$ to hidden layer $\boldsymbol{h}^{(1)}$, a nonlinear activation $A^{(1)}$ for the hidden layer, followed by a linear layer from hidden layer to output layer and an activation $A^{(2)}$ for the output. I choose only 1 hidden layer, since the predicted log-likelihoods on hold-out datasets do not improve with more hidden layers[2]. Input $\boldsymbol{z}$ includes all characteristics: age, gender, income, first class, who pays for travel cost, trip purpose and luggage. We experiment with various sets of hyper-parameters and activation functions (Table 5.5).

[2]The results are not report here

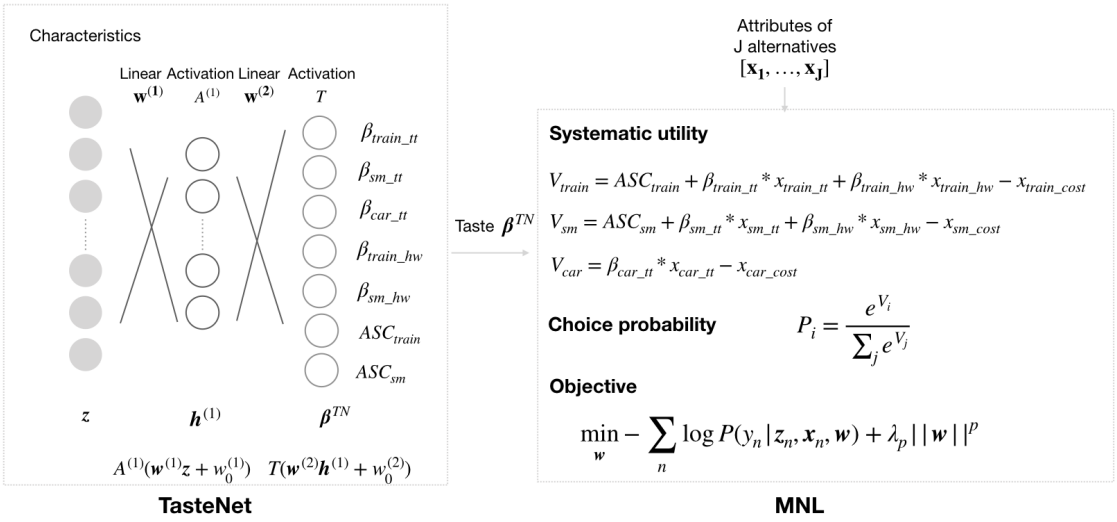


Figure 5-1: Diagram of TasteNet-MNL for Swissmetro Dataset

Table 5.5: Options of Hyper-parameters & Activation Functions

Options	Values
Hidden activation	$relu$, $tanh$
Output activation (for non-positive parameters)	$-relu(-\beta)$, $-exp(-\beta)$
Hidden layer size	[10, 20, ..., 100]
$l1$ or $l2$ regularization [a]	[0, 0.0001, 0.001, 0.01]

[a] Either $l1$ or $l2$ regularization is used.

5.5 Results

The estimated model coefficients for MNL-A, MNL-B and MNL-C are shown in Table 5.2, 5.3 and 5.4. Among all TasteNet-MNL scenarios, the one with 80 hidden units, *ReLU* for hidden layer activation, negative exponential for non-positive output activation, and no regularization achieves the best prediction performance on the development dataset.

5.5.1 Prediction Performance

TasteNet-MNL significantly out-performs MNL benchmarks in terms of prediction accuracy. I use average negative log-likelihood (NLL) and prediction accuracy (ACC) as metrics for model predictability (Table 5.6). From MNL-A to MNL-C, more interactions between attributes and individual characteristics are added. MNL-C has a full set of interactions between attributes and characteristics. Surprisingly, the predicted log-likelihood shows only marginal improvements: average NLL decreases from 0.728 (MNL-A) to 0.708 (MNL-B) and 0.691 (MNL-C). With TasteNet-MNL, we see a substantial improvement in prediction performance: NLL on development data drops from 0.691 to 0.646. This is attributed to the flexibility enabled by the hidden layer with nonlinear transformation in the TasteNet. The improved log-likelihood implies the existence of nonlinear effects in utility specification. The neural network is able to automatically learn taste as a nonlinear function of individual characteristics. Because it captures a more accurate relationship between characteristics and taste, it outperforms MNLs with linear utilities.

5.5.2 Individual Taste Estimates

We want to understand how estimated tastes, such as willingness-to-pay, differ across models. We apply each model to obtain taste parameters for each individual in the Swissmetro dataset. The averages tastes of the population are shown in Table 5.7.

We find that from MNL-A to MNL-C, the average value of time (VOT) increases as more interaction terms are added to the utility function. For example, train VOT

Table 5.6: Average Negative Log-likelihood (NLL), Prediction Accuracy (ACC) and F1 Score by Model

	NLL			ACC			F1		
Model	train	dev	test	train	dev	test	train	dev	test
MNL-A	0.762	0.728	0.755	0.662	0.691	0.66	0.535	0.557	0.534
MNL-B	0.73	0.708	0.72	0.678	0.69	0.678	0.578	0.585	0.573
MNL-C	0.704	0.691	0.698	0.685	0.706	0.678	0.588	0.611	0.574
TasteNet	0.607	0.646	0.645	0.737	0.718	0.703	0.668	0.634	0.620

increases from 1.34 to 1.85$ per minute. Swissmetro VOT increases from 1.4 to 1.51$ per minute, and Car VOT rises from 1 to 1.35$ per minute. Both MNL-B and MNL-C suggest that the VOT of train is higher than swissmetro and car. In terms of the value of headway time (VOHE), MNL-C also gives higher average VOHE than MNL-B or MNL-A.

TasteNet-MNL gives the largest mean of VOT and VOHE among all models (Table 5.7). Its average VOT estimates for train, swissmetro and car are 26%, 17% and 24% higher, respectively, than those predicted by MNL-C. Its average VOHE estimates for train and swissmetro are 25% and 67% higher than the MNL-C estimates.

To understand where the higher average VOTs come from, I plot population taste distributions (Figure 5-2). As interactions are added incrementally from MNL-A to MNL-C, the models capture a wider range of taste variations. Compared to the MNLs, however, TasteNet-MNL suggests more heterogeneous taste in the population, which is not captured by MNLs with linear utilities. In particular, the VOTs and VOHEs for all modes have longer tails on the high end of WTP. It seems that TasteNet-MNL's better prediction performance is a result of its more accurate estimates of individual tastes.

5.5.3 Taste Function

Each model provides a function that maps individual characteristics to a type of taste value (e.g. VOT, VOHE). We compare taste functions provided by different models. Since function input $\boldsymbol{z}$ is multi-dimensional, we cannot directly visualize the functions.

Table 5.7: Population Mean of Taste Parameters Estimated by Models

Mode	Taste	MNL-A	MNL-B	MNL-C	TasetNet-MNL	(vs MNL-C)
TRAIN	TT	-1.338	-1.710	-1.846	-2.327	26%
	HE	-0.451	-0.616	-0.880	-1.102	25%
	ASC	-0.198	0.234	0.368	0.801	117%
SM	TT	-1.401	-1.514	-1.505	-1.764	17%
	HE	-0.817	-0.701	-1.039	-1.733	67%
	SEATS	0.172	0.189	0.420	0.266	-37%
	ASC	0.648	0.510	0.512	0.669	31%
CAR	TT	-1.018	-1.251	-1.354	-1.685	24%

Instead, we pick an individual with characteristics $\boldsymbol{z}$. We vary one dimension of $\boldsymbol{z}$: z_i, while keeping other dimensions fixed $z_{j\neq i}$. We ask different models for the value of a particular taste parameter $\beta = f_{model}(z_i; z_{j\neq i})$. We plot the estimated taste versus input z_i for each model.

For example, we pick a person with characteristics shown in Table 5.8. We vary this person's income and ask each model a question, for example: what are the VOTs for such a person as his income varies? We compare the answers given by different models. Figure 5-3 shows the VOTs and VOHEs estimated by different models versus income.

Compared with the benchmark MNLs, the values of VOT and VOHE estimates by TasteNet-MNL all fall within credible ranges. TasteNet-MNL gives more or less different estimates. Swissmetro VOT estimates are not very different between TasteNet-MNL and MNL-C. Regarding VOT for train, TasteNet-MNL gives smaller estimates for all three income groups than MNL-C. Car VOTs estimated by TasteNet-MNL are larger for higher income groups and lower for the lowest income group. With respect to VOHEs, TasteNet-MNL gives higher estimates for train and lower estimates for swissmetro for all income levels. MNL-C shows a monotonic relationship between VOT and income only for train VOT, while TasteNet-MNL identifies the monotonicity for swissemtro VOT and car VOT.

Table 5.8: Example Person Selected for Comparing Taste Function by Models

	Characteristics	Value
$\boldsymbol{z}_{fixed}$	MALE	Male
	AGE	(39,54]
	PURPOSE	Commute
	WHO	Self
	LUGGAGE	One piece
	GA	Yes
	FIRST	No
$\boldsymbol{z}_{vary}$	INCOME	0: under 50, 1: 50 to 100, 2: over 100

5.5.4 Elasticity & Choice Probability

First we apply the models to calculate dissaggregate point elasticities for each observation in Swissmetro dataset (Eqn.4.9 and 4.10). We measure the differences between the models by mean absolute difference.

With individual elasticities, we compute aggregate elasticity, which measures a group of decision-makers' response to an incremental change in a variable. This is defined in Eqn 5.1 as the percentage change in the expected share of the group choosing alternative i (W_i) with respect to one percentage change in variable x_{ik}. It is equivalent to a weighted average of the individual elasticities using the choice probabilities as weights. TasteNet-MNL's individual elasticities estimates differ from MNL-C by 0.2287 on average.

The aggregate elasticities of Swissmetro mode share with respect to Swissmetro travel time are similar: -0.43, -0.45, and -0.41 for MNLs and -0.437 for TasteNet-MNL. We also compare aggregate elasticity by group, such as income (Table 5.9). TasteNet-MNL predicts higher elasticities for low income and high income groups than MNL-C. But overall, TasteNet-MNL gives choice elasticities close to MNLs and within reasonable range.

$$E_{x_{ik}}^{W(i)} = \frac{\partial W(i)}{\partial x_{ik}} \frac{x_{ik}}{W(i)} = \frac{\sum_n P_n(i) E_{x_{ink}}^{P_n(i)}}{\sum_n P_n(i)} \tag{5.1}$$

Table 5.9: Aggregate Choice Elasticity of Swissmetro w.r.t Time by Income

	INCOME0	INCOME1	INCOME2
MNL-A	-0.3765	-0.4297	-0.4706
MNL-B	-0.3975	-0.3923	-0.5329
MNL-C	-0.3759	-0.3706	-0.4653
TasteNet-MNL	-0.4200	-0.3982	-0.4810

5.6 Summary

On the Swissmetro dataset, TasteNet-MNL discovers a wider spectrum of taste variations in the population than the benchmarking MNLs. TasteNet-MNL also predicts more accurately on hold-out datasets (dev and test). Its superior predictability is a result of its flexibility in capturing nonlinear taste functions. Values of time and elasticities derived from TasteNet-MNL are reasonable compared to the results from the MNLs. However, the average VOTs estimated by TasteNet-MNL are higher than the MNLs, due to the longer tails on the high end of willingness-to-pay. Through this exercise, I show that TasteNet-MNL can not only predict more accurately, but also provide interpretable indicators for policy analysis.

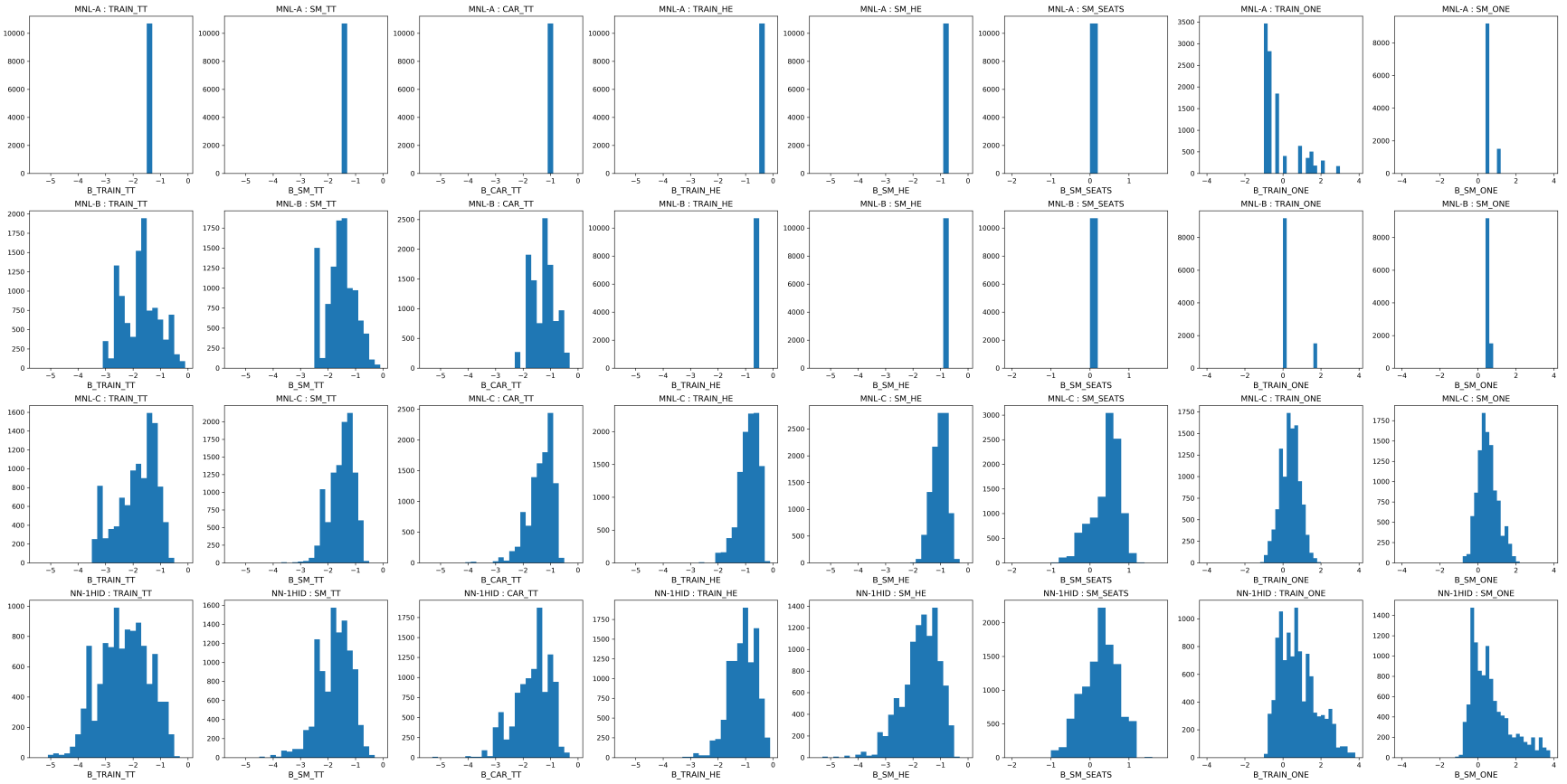


Figure 5-2: Population Taste Distributions by Models (Swissmetro Dataset

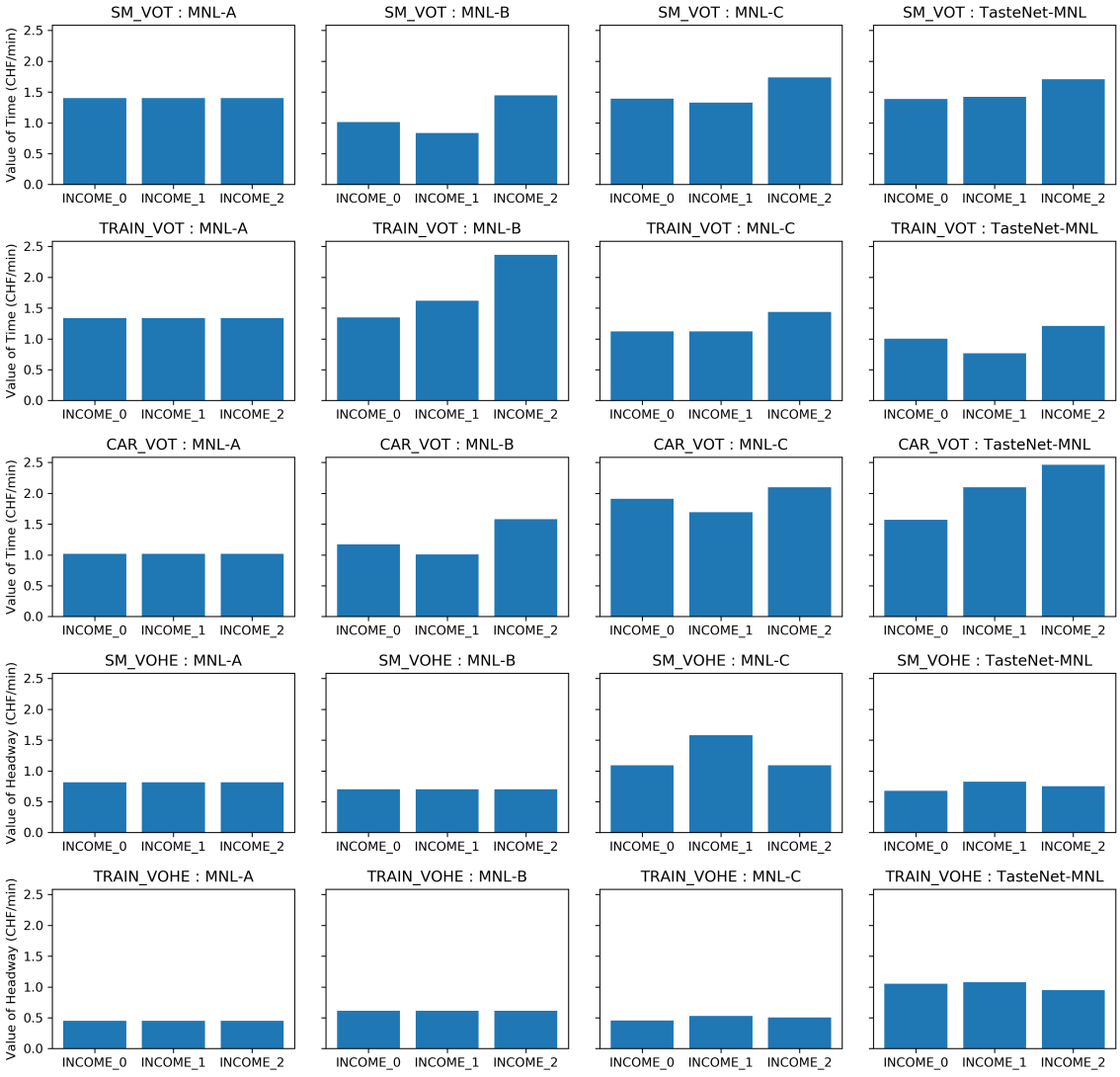


Figure 5-3: Tastes as Functions of Income for a Selected Person by Different Models

Chapter 6

A Neural Network Representation of the Latent Class Choice Model

6.1 Introduction

As discussed, relatively recent studies have started to recognize the connections between neural networks (NNs) and DCMs, and attempt to take advantage of the strengths of both (Wong et al., 2018; Sifringer et al., 2018; van Cranenburgh and Alwosheel, 2019). Along these lines, we propose a NN approach to estimate a Latent Class Choice Model (LCCM). We also enrich the class membership model by including nonlinearity in the corresponding part of the neural network.

This approach addresses two common challenges in LCCM. First is the computational challenges with complex models and/or large-scale datasets. Current LCCM estimation methods - direct maximum likelihood with quasi-Newton routine or Expectation-Maximization algorithm - are computationally expensive with a large number of classes, parameters, and/or observations. In particular, estimating a model with a large number of classes is still challenging. For example, some classes obtain very small probabilities; and parameters tend to collapse to the same values across classes (Hess, 2014).

DNNs are well known for allowing a large number of parameters (even millions) and having a deep structure with many layers of nonlinear transformation and compli-

cated objective functions with a proliferation of local optimal, saddle points and flat regions (Goodfellow et al., 2016). DNNs are almost exclusively trained with stochastic gradient descent (SGD) algorithms, because of their computational efficiency and scalability for large datasets and high-dimensional optimization. We formulate LCCM as a neural network to examine whether we can improve LCCM estimation speed and accuracy by an SGD algorithm developed for DNN training.

The second challenge relates to the uncertainty in class membership model specification. LCCM represents random taste heterogeneity through a discrete mixture of choice models, representing different preference groups characterized by different model structures, utility forms and/or taste parameters. In contrast to choice models, where we more clearly know about the trade-offs between attributes, the class membership model specification is less clear to define. In practice, class definition is carried out by trial-and-error, with the best model selected according to fitted log-likelihood and interpretability. The relationship between class membership and individual characteristics can be nonlinear. It is almost impossible to test all nonlinear scenarios. A misspecified class membership model can lead to biased parameter estimates and mixing distribution; and it also results in poor predictability.

Given limited prior knowledge about class membership, a data-driven approach can be beneficial to learn the mixing distribution. We propose using a neural network, which is a universal function approximator (Cybenko, 1989; Hornik, 1991), to learn the class membership model. By adding a hidden layer with nonlinear transformation to the class membership network, we can learn more flexible mixing distribution. We call this extended model nonlinear-LCCM. With an application to Swissmetro mode choice, we show that the nonlinear-LCCM outperforms LCCM in log-likelihood and prediction accuracy on a hold out dataset. We confirm that the prediction gain comes from more a accurate latent class assignment.

6.2 Advantages of Stochastic Gradient Descent

Optimization algorithms that use the entire training data are called batch gradient methods (BGD), as they use all training examples as a single batch to estimate gradient and update parameters. Conventional quasi-Newton algorithms are batch gradient methods. Stochastic gradient methods are widely used in machine learning, especially in deep learning. They are a class of optimization algorithms that process one example or a small set of examples (mini-batch) in each iteration. They obtain an unbiased gradient estimate by computing the average gradient on a random sample from the data generation distribution (Goodfellow et al., 2016).

The most popular class of stochastic optimization algorithms is stochastic gradient descent (SGD). Training a deep neural network almost exclusively uses SGD due to its superior performance for complex optimization and its scalability to large datasets. Compared to Newton or quasi-Newton algorithms, it saves computation cost significantly for two reasons. First, it does not need to compute the Hessian. The computation cost is $O(K)$ instead of $O(K^2)$. Second, it uses mini-batch instead of the entire data to estimate the gradient. Therefore its computation time per update does not grow with the number of training examples. Most algorithms converge much faster in terms of total computation if they rapidly approximate gradients rather than slowly computing the gradient exactly (Goodfellow et al., 2016). This is because the returns of using more examples to estimate the gradient are less than linear and a large training dataset can be redundant. (Goodfellow et al., 2016).

Given these advantages of SGD, we want to examine whether it can help improve LCCM estimation, especially under challenging scenarios. We formulate LCCM as a neural network and estimate it with SGD. We compare SGD with traditional batch methods, DML and EM. We expect that SGD estimates faster, and scales better to large datasets and the many-class problem.

It should be noted that a neural network is not a necessary condition to apply SGD. I choose a neural network representation for several reasons. First, it is convenient to access many advanced SGD algorithms developed for DNNs. Training this neural

network is similar to performing maximum likelihood estimation. Second, a neural network provides a generic platform for model development, extension and integration, thanks to the auto-differentiation (AD) functionality. AD is a technique that numerically evaluates the derivative of a function specified by a computer program by applying chain rule to a sequence of arithmetic operations. It is more accurate and scales better than numerical gradient approximation (Baydin et al., 2018). Since it is automatic and generic, we do not need to derive a gradient for a new model structure, which can be time-consuming and prone to error. We can extend LCCM in various ways with only small changes in the objective function and network structure, such as adding nonlinearity to the class membership model, and extending LCCM for multi-dimensional choices, or multi-output with mixed types (continuous and discrete). Thirdly, with a neural network, we can explore flexible model specifications, such as a nonlinear class membership model.

6.3 Neural Network Structures in Parallel to Discrete Mixture Models

In neural network literature, Mixture-of-Experts (MoEs) networks (Jacobs et al., 1991; Jordan and Jacobs, 1994) and Mixture Density Networks (MDNs) (Bishop, 1994) have very similar structure as discrete mixture models except that they are non-parametric. Jacobs et al. (1991) develop a neural network composed of several "expert" networks and a "gating" network that decides which of the experts should be assigned for each training case. The gating network is a feed-forward neural network, where the information flows only in the forward direction. The outputs of the gating network are normalized to logit probabilities. The gating network serves as a stochastic switch with probabilities of selecting the expert networks. Expert networks specialize in different sub-tasks and have different weights. The total likelihood function is a sum of expert selection probabilities times expert likelihood over all experts. The gating network corresponds to the class membership model in LCCM. The

expert networks resemble class-specific choice models. In parallel to MoE assigning tasks to different experts given their features, LCCM assigns each choice-maker to latent preference groups according to their characteristics.

Jordan and Jacobs (1994) extends MoE to Hierarchical Mixtures of Experts(HME). Instead of one gating network, HME is a tree structure with gating networks sitting at the non-terminals of the tree, and expert networks sitting at the leaves of the tree. Gating networks, each as a multinomial logit, produce partition probabilities at different levels of the tree given input x. The expert networks at the bottom are also logit models produce output vector y or probability of y for each input x. These output vectors travel up the tree blended by the gating network probabilities.

Bishop (1994) proposes a general framework called Mixture Density Network (MDN) to model an arbitrary conditional density function $p(y|x)$. MDN is similar to MoE: it combines a feed-forward neural network and a mixture model. The neural network models the mixing coefficients (prior probabilities) as a general function of input x. Each component in the mixture models has a kernel function to compute probability density. By choosing a mixture model with a sufficient number of kernels and a neural network with a sufficient number of hidden units, MDN can approximate as closely as desired any conditional density function $p(y|x)$.

We represent LCCM by a neural network in a similar fashion as the MoE or MDN. Instead of entirely using black-box neural networks, we keep the choice models parametric. Class membership models can be confirmatory or exploratory. Since the relationship between individual characteristics and latent class can be nonlinear, without good prior knowledge it would be hard to specify it correctly. A black-box neural network can learn these relationships from data. So we extend the class membership model by adding a hidden layer with nonlinear transformation to the corresponding sub-network. Our hypothesis is that if a nonlinear effect does exist, the class membership network can learn a better discrete mixing distribution for modeling taste heterogeneity, and will predict more accurately than a logit class membership model with linear utility functions can.

6.4 A Neural Network Representation of LCCM

A latent class choice model consists of a class membership model and class-specific choice models. The class membership model probabilistically assigns a decision-maker to latent classes. It outputs the probability of individual n belonging to class s (Q_{ns}) based on individual characteristics z_n. A logit form of class membership probability is commonly used (Eqn.6.1).

Each latent class is associated with a class-specific choice model. The choice model of a latent class predicts the conditional probability of a person choosing an alternative given that the person is a member of the latent class. The conditional choice probability (P_{nis}) and conditional choice likelihood (P_{ns}) is shown in Eqn. 6.2 and Eqn. 6.3, respectively. The objective of model estimation is to maximize the total log-likelihood function (Eqn. 6.4) by solving class membership parameters γ and choice model parameters ($\beta_s \forall s$).

$$Q_{ns} = Q(s_n = s|\boldsymbol{z_n}; \boldsymbol{\gamma}) = \frac{e^{f(\boldsymbol{z_n};\boldsymbol{\gamma_s})}}{\sum_l e^{f(\boldsymbol{z_n};\boldsymbol{\gamma_l})}} \tag{6.1}$$

$$P_{nis} = P(y_n = i|\boldsymbol{x_n}, \boldsymbol{z_n}; \boldsymbol{\beta_s}) = \frac{e^{V_{nis}}}{\sum_j e^{V_{njs}}} = \frac{e^{V(\boldsymbol{x_{ni}},\boldsymbol{z_n};\boldsymbol{\beta_s})}}{\sum_j e^{V(\boldsymbol{x_{nj}},\boldsymbol{z_n};\boldsymbol{\beta_s})}} \tag{6.2}$$

$$P_{ns} = \prod_i P_{nis}^{[y_n=i]} \tag{6.3}$$

$$L(\gamma, \beta) = \sum_n \log \sum_s Q_{ns} P_{ns} \tag{6.4}$$

A neural network represents the same structure by a set of feed-forward networks (choice models) with a gating network (class membership model) on top (Figure 6-1). The *class membership model* is represented as a feed-forward neural network with one linear layer and softmax activation on the output (class-net). The linear layer takes explanatory variables $\boldsymbol{z}$ as inputs, and outputs a linear combination of the inputs for each output unit, which corresponds to the utility of each latent class (Eqn. 6.5). The

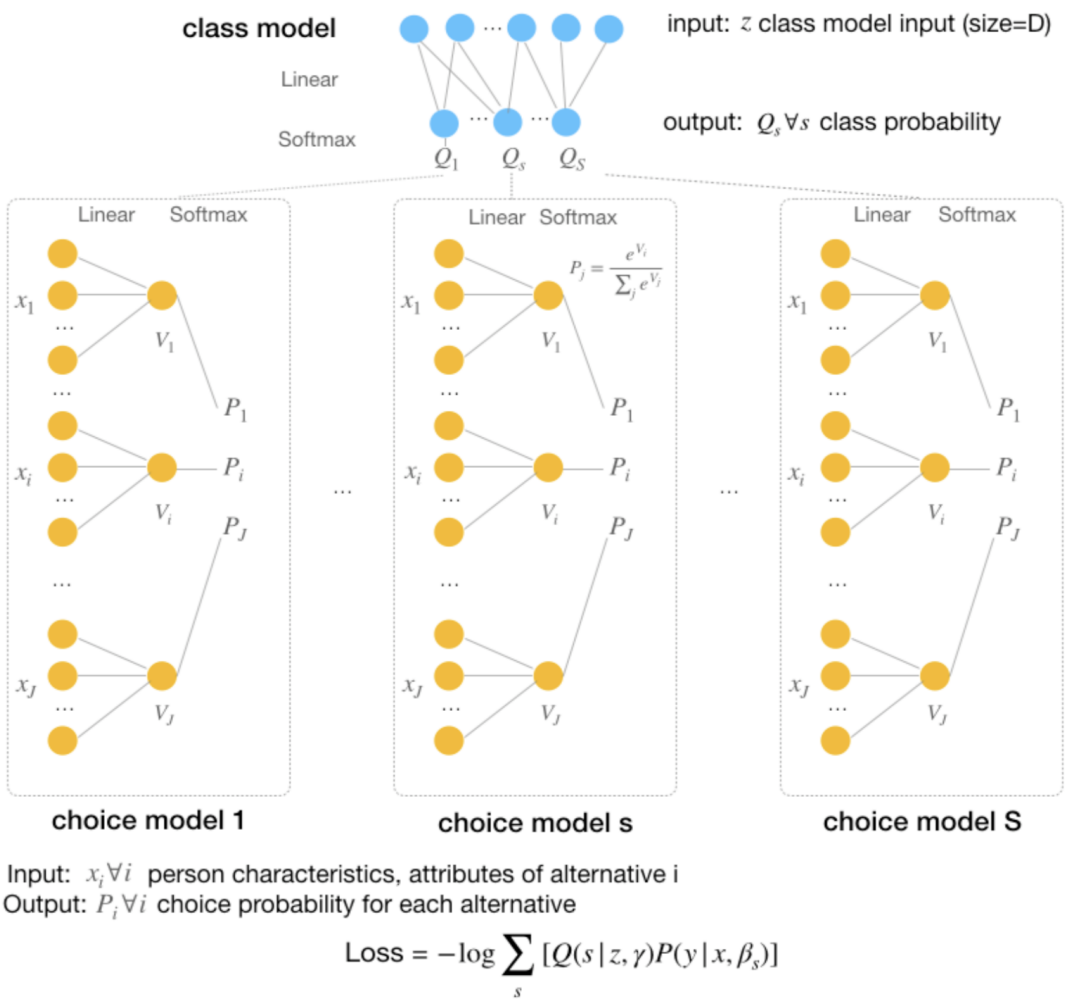


Figure 6-1: A Neural Network Representation of the Latent Class Logit Model

softmax function then transforms utilities into probability space (Eqn. 6.6) with the same logit probability formula. Parameters of the class membership model $\boldsymbol{\gamma}$ are the coefficients in the class utilities, which correspond to the weights of the linear layer.

$$V_s = \boldsymbol{\gamma}_s' \boldsymbol{z} + \gamma_{s0}, \forall s = 1, .., S \tag{6.5}$$

$$Q_s = \frac{e^{V_s}}{\sum_q e^{V_q}}, \forall s = 1, ..., S \tag{6.6}$$

Class-specific *choice models* are represented by S separate neural networks (choice-nets). A choice-net consists of a number of linear layers in parallel. Each linear layer

outputs the utility of alternative i as a function of the input variables $\boldsymbol{x_i}$ (Eqn. 6.7). Softmax activation transforms utilities to choice probabilities (Eqn. 6.8)

$$V_{is} = \boldsymbol{\beta'_{is}} \boldsymbol{x_i} + \beta_{0s}, \forall i = 1, ..., J \tag{6.7}$$

$$P_{is} = \frac{e^{V_{is}}}{\sum_j e^{V_{js}}}, \forall i = 1, ..., J \tag{6.8}$$

The goal of training of a neural network is to find the optimal parameters that minimize a loss function. We choose negative log-likelihood (NLL) defined in Eqn. 6.4 as the loss function. Minimizing NLL loss is similar to maximizing log-likelihood.

6.5 Network Training

Neural networks are trained through back-propagation (Rumelhart et al., 1988). During one training epoch (a full pass through the training data), training data are randomly split into mini-batches. At each iteration, one mini-batch is passed through the network to compute loss (forward-pass). Then gradients of loss with respect to network weights are estimated using the mini-batch of examples by auto-differentiation. Weights are updated based on the gradient to reduce loss by SGD (back-propagation). This process of forward-pass and back-propagation is repeated on mini-batches, and run through the entire data for multiple times (epochs) until a stopping criteria is met. We usually stop when the change of loss function on training data is small, or change of loss on development data is small (to prevent over-fitting).

A disadvantage of SGD is that batch size and learning rate need to be decided. Goodfellow et al. (2016) provides general guidance to such decisions. Larger batch sizes can give more accurate gradient estimates, but the return is less than linear. Also, memory requirement scales with batch size, since all examples in the batch are to be processed in parallel. Small batches can have a regularization effect due to the noise they add to the learning process (Wilson and Martinez, 2003). Overall the algorithm is faster since it requires fewer examples to estimate the gradient. In

practice, we can try several batch sizes (e.g. 64, 128, 256) and decide based on computation time and loss value.

Another critical decision relates to the learning rate (or step size). The basic SGD algorithm updates the weights by taking a small step along the negative gradient direction. If the learning rate is too big, loss function will oscillate a lot and even increase. If the learning rate is too low, loss function decreases very slowly and may get stuck in a flat region with high values. The learning rate is decided by observing the loss function in the first few iterations (for more details, see chapter 8 in Goodfellow et al. (2016)). The learning rate should also be decreased over time as it gets closer to a local optimal because the gradient noise from the mini-batch does not vanish even when we arrive at a minimum.

More advanced SGD algorithms incorporate adaptive learning rate and momentum. Momentum (Polyak, 1964) is developed to accelerate learning, especially to solve the problem of poor conditioning of the Hessian matrix and variance in the stochastic gradient. It accumulates a moving average of past gradients with exponential decay to continue to move in their direction. Readers can find details about a variety of SGD algorithms in Goodfellow et al. (2016). For our application, I choose Adam (Kingma and Ba, 2014), an algorithm with adaptive learning rates and momentum. Adam is robust to the choice of hyper-parameters, and well-suited for problems that are large in terms of data and/or parameters. It is one of the most popular algorithms for training deep neural networks.

6.6 Summary

In this chapter, I have discussed why it is beneficial to formulate and estimate an LCCM as a neural network. The main reasons are to use SGD algorithms for scalability and the auto-differentiation ability to ease model extension/integration. The resemblance between discrete mixture models and MDN/MoE neural networks are drawn. I describe how to estimate an LCCM by training its corresponding network.

Chapter 7

Experiments for Comparing LCCM Estimation Methods

7.1 Synthetic Data Generation

To compare the performance of LCCM estimation methods, we design three groups of experiments. Table 7.1 provides the parameter notations and dimensions used in the data generation models (DGM). All the DGMs have a logit structure for the class membership model and class-specific choice models. The total number of latent classes is S. The utility functions of latent classes share the same set of explanatory variables of size D. Class membership model coefficients are class-specific with one latent class coefficient fixed to zero as the reference. The dimension of class membership model parameters ($\boldsymbol{\gamma}$) is (D+1)(S-1).

Each class-specific choice model has J alternatives. The utility of each alternative is a linear function of K input variables plus an intercept. Choice model coefficients are alternative-specific. They are also class-specific. Thus we have a total of S*(KJ+J-1) parameters ($\boldsymbol{\beta}$) in choice models. Group I tests the effects of sample size on different estimation approaches. The DGM of group I has 3 latent classes, 7 individual characteristics, 10 attributes and 3 alternatives. We use the same DGM to generate three datasets of different sizes: 10k, 50k and 100k.

Group II is created to compare different estimation approaches under the scenario

Table 7.1: Summary of the Synthetic Data for LCCM Estimation

Size	Description	Group I	Group II	Group III
N	Number of persons	[10k, 50k, 100k]	[50k, 100k]	[10k, 50k]
D	Size of input (z) in class membership model	7	20	7
K	Size of input (x) in choice model	10	10	10
S	Number of latent classes	3	10	3
J	Number of alternatives	3	3	3
$\lvert\boldsymbol{\gamma}\rvert$	Parameters in class membership model: $(D+1)\times(S-1)$	16	189	16
$\lvert\boldsymbol{\beta}\rvert$	Parameters in choice models: $(K\times J+J-1)\times S$	96	320	96
	Total parameters	112	509	112
	Class shares	balanced	balanced	unbalanced

when the number of latent classes is fairly large. We increase the DGM's number of latent classes from 3 to 10 and input dimension from 7 to 20. With the same DGM, we generate two datasets with 50,000 and 100,000 samples.

Both Group I and II have relatively balanced class membership shares, and the smallest class has a fairly large amount of data points.

Group III is designed to test a more extreme scenario: the class membership distribution is highly unbalanced. The underlying DGM has 3 classes and the same input dimensions as Group I. Shares of the 3 latent classes are 89.3%, 3.7% and 7.0%. We generate two datasets with 10k and 50k examples. The smallest class in the 10k dataset has 370 observations, while the 50k dataset has 740 observations.

Model parameters of underlying DGMs are drawn from Normal distribution with zero mean and standard deviation 2. Model inputs are randomly generated from standard Normal distribution.

7.2 Estimation Methods

We compare three alternative estimation procedures: direct maximum likelihood with quasi-Newton routine (BFGS), EM and Adam. We use *PandasBiogeme* to realize BFGS estimation. For EM, we choose python *lccm* package[1] developed by El Zarwi (2017). Both packages utilize scipy.optimize.minimize for maximizing log-likelihood. We program the neural network version of LCCM in PyTorch[2], an open-source deep

[1]https://github.com/ferasz/LCCM
[2]https://pytorch.org/

learning platform. We compare model estimation time and parameter estimation accuracy among the three methods. Parameter accuracy is measured by Mean Absolute Error (MAE).

Stopping criteria is critical to obtain accurate estimates. There are 3 commonly used criteria to decide whether estimation has converged: gradient norm, function value change, and parameter change. The criteria adopted by *PandasBiogeme* are gradient norm smaller than 1e-7 and function change smaller than the smallest float representable by machine[3]. The *lccm* python package uses a log-likelihood value change smaller than 1e-4 to stop EM iterations. Based on gradient norm or function value change, parameters can still move around when the algorithm stops if the model is not identifiable. Parameter change is a stricter convergence criteria. In Monte-Carlo experiments, we choose average absolute percentage change in parameters being smaller than a threshold (0.1%) as the stopping criteria. However, this threshold may be varied depending on real data application.

Because of the noise from mini-batch, Adam's parameter and objective value can still oscillate near convergence. To be comparable with batch methods, we enlarge the batch size towards the end to half the size of the data, to steadily reach the convergence. We increase batch size when parameter percentage change between two epochs is below 5%.

7.3 Results

Table 7.2 summarizes the estimation time and parameter estimation accuracy by BFGS, EM and Adam.

Group I

With 3 latent classes and 10k data (S3N10k), three methods take about the same amount of time (0.3 minutes) to converge. Adam's γ estimate is more accurate than BFGS and EM. But the advantage is small (MAE = 0.06 vs. 0.09). Adam's and

[3]opts = {'gtol' : 1e-7,'ftol' : np.finfo(float).eps}

Table 7.2: Convergence Time[a] and Mean Absolute Error (MAE) of Parameter Estimates[b] (Balanced Classes)

Group	Data		Adam	BFGS	EM
3 classes, balanced	S3N10k	Time	0.32	0.35	0.35
			(0.025)	(0.077)	(0.035)
		MAE($\boldsymbol{\gamma}$)	0.0682	0.0941	0.0932
			(3e-3)	(1e-3)	(4e-5)
		MAE($\boldsymbol{\beta}$)	0.1062	0.1308	0.1095
			(7e-3)	(1e-2)	(7e-6)
	S3N50k	Time	0.9	1.5	3.7
			(0.045)	(0.17)	(0.033)
		MAE($\boldsymbol{\gamma}$)	0.052	0.03	0.03
			(2e-3)	(2e-3)	(2e-5)
		MAE($\boldsymbol{\beta}$)	0.0688	0.0957	0.0441
			(2e-3)	(1e-2)	(6e-6)
	S3N100k	Time	1.6	3.3	10.3
			(0.30)	(0.35)	(0.41)
		MAE($\boldsymbol{\gamma}$)	0.0236	0.0221	0.0224
			(2e-3)	(8e-4)	(1e-5)
		MAE($\boldsymbol{\beta}$)	0.0367	0.0873	0.0364
			(2e-3)	(0.018)	(2e-6)
10 classes, balanced	S10N50k	Time	1.6	NA[d]	69
			(0.22)	NA	(28.5)
		MAE($\boldsymbol{\gamma}$)	0.0905	NA	0.063[c]
			(0.0103)	NA	(9e-6)
		MAE($\boldsymbol{\beta}$)	0.1192	NA	0.08[c]
			(7e-3)	NA	(1e-6)
		Success rate	5/5	0/5	3/5
	S10N100k	Time	2.5	NA	140
			(0.18)	NA	(74.5)
		MAE($\boldsymbol{\gamma}$)	0.0485	NA	0.0526
			(7e-3)	NA	(9e-6)
		MAE($\boldsymbol{\beta}$)	0.0912	NA	0.061
			(6e-3)	NA	(8e-7)
		Success rate	5/5	0/5	5/5

[a] Time in minutes

[b] Values are means and standard deviations (in parenthesis) of all 5 model runs without a special note.

[c] Values are only for the 3 successful runs (global optimal found).

[d] NA: Not available.

Table 7.3: Convergence Time[a] and Mean Absolute Error (MAE) of Parameter Estimates[b] (Unbalanced Classes)

Group	Data		Adam	EM
3 classes, unbalanced	S3N10k_ub	Time	1.37	8.62
			(0.12)	(0.92)
		MAE($\boldsymbol{\gamma}$)	0.1778	1.2
			(8e-3)	(8e-5)
		MAE($\boldsymbol{\beta}$)	0.2095	10.1
			(0.011)	(1e-3)
		Success rate	5/5	0/5
	S3N50k_ub	Time	1.19	27.6
			(0.20)	(5)
		MAE($\boldsymbol{\gamma}$)	0.0894	0.0537
			(3e-3)	(2e-6)
		MAE($\boldsymbol{\beta}$)	0.1308	0.0817
			(3e-2)	(2e-5)
		Success rate	5/5	5/5

[a] Time in minutes
[b] Values are means and standard deviations (in parenthesis) of all 5 model runs.

EM's β accuracy are similar and slightly better than BFGS.

As sample size becomes 5 times as large (S3N50k), the estimation time of BFGS is multiplied by 5, with EM by 10 and Adam by 3. EM takes almost 4 minutes to converge while Adam needs less than 1 minute. Accuracy-wise, EM obtains the most accurate estimates for γ (MAE=0.03) and β (MAE=0.044). Adam is slightly less accurate (MAE(γ)=0.052, MAE(β)=0.069). Again, BFGS gives the highest β error among the three (MAE=0.096), but its γ error is similar to EM. Note that the error differences among the three methods are not significant, about 2 to 4% of the average magnitude of the true γ and β in the DGM.

When sample size increases to 100k (S3N100k), the gap of computation time between the methods widens. BFGS takes 10 times as long as it does on 10k data. EM takes 30 times as long. The estimation time of Adam is only multiplied by 5. As a result, EM takes 10.3 minutes to converge, while BFGS and Adam need 3.3 and 1.6 minutes, respectively. Adam and EM reach the same accuracy for γ and β. BFGS, however, persistently has the biggest error in β among the three.

In summary, the results of group I shows that SGD is more time-efficient on a large dataset. The time for Adam increases at a much slower pace than sample size, while the time needed by BFGS grows in proportion to sample size. EM's estimation time increases more than proportional to sample size, due to the posterior computation in the E-step, and its slow convergence rate. Regarding accuracy, differences among them are not substantial. Adam gives slightly more accurate estimates than BFGS or EM when sample size is relatively small (10k). As sample size increases, EM is more accurate than Adam and BFGS. When sample size is very large (100k), parameter errors by EM and Adam match. BFGS constantly obtains higher β error than EM and Adam under the different sample size scenarios.

Group II

The DGM in group II is more complex with 10 latent classes and a total of 509 parameters. On both 50k and 100k data, BFGS suffers severely. None of the 5 runs converge after two days. We are not able to obtain any results as PandasBiogeme does not terminate. Various reasons may explain this result. First, the second-order gradient method is computationally expensive with cost $O(K^2)$ quadratic to the number of parameters K. Also, as discussed in Train (2008), with an increasing number of parameters, standard maximum likelihood becomes numerically difficult because the inversion of the Hessian becomes difficult, with the possibility of singularity at some iteration. The optimization algorithm can get stuck in areas where the objective function are not well approximated by quadratic form.

EM is more stable compared to BFGS. On the 50k data, all 5 runs converge, but 2 of them end up in a local optimum with much lower log-likelihoods. EM convergence time varies from 35 to 120 minutes, with an average of 69 minutes. Adam performs more consistently than EM. All 5 runs converge successfully and find solutions close to the truth. Adam takes an average of 1.6 minutes with a standard error of 0.2 minutes, only 1/70th the amount of time for EM.

If we only compare the 3 successful EM runs with Adam, EM outperforms Adam in parameter accuracy by a small margin: MAE of γ by EM is 0.06 compared to 0.09

by Adam; and MAE of β is 0.08 compared to 0.12 by Adam. Regarding the mean absolute values of γ and β (2.2 and 1.6, respectively, EM is more precise than Adam by 2 to 3 percent of the true parameter values.

When the sample size is enlarged from 50k to 100k, EM takes 140 minutes on average to converge, with large standard deviation, while Adam needs 2.5 minutes. On average, EM takes 56 times as much time as Adam. Accuracy-wise, the two methods are close.

Comparing group II with group I, EM and Adam perform differently as model complexity increases. Let us fix sample size to 50k and compare the 3-class model with 112 parameters in group I (S3N50k) and the 10-class model with 509 parameters in group II (S10N50k). Adam's time increases from 0.9 to 1.6 minutes, less than double; while EM's time increases from 3.7 minutes to 69 minutes (multiplied by 19 times).

In summary, this group of experiments shows that BFGS and EM require significantly longer time than Adam on problems with a large number of classes and a big set of parameters. BFGS fails to converge for a 10-class estimation after a long period of time. EM takes around 60 to 70 times as long as Adam on 10-class synthetic datasets, with large fluctuations across runs. When sample size is not large enough (e.g. 50k), EM is more prone to local optimal solutions (2 out of 5) than Adam. When EM does converge to global optimum, it obtains similar or slightly more precise parameter estimates than Adam. Adam scales better in terms of estimation time and stability than EM or BFGS for estimating more complex LCCMs, while maintaining a similar level of parameter accuracy.

Group III

The two datasets in group III (S3N10k_ub, S3N50k_ub) share the same DGM that produces highly unbalanced class shares. However, the total sample size difference (10k vs 50k) results in a different absolute number of observations in the small classes. Here we only compare EM with Adam, since BFGS is less stable than EM.

a) S3N10k_ub

Table 7.4: Class Membership Model Coefficients

(Data: S3N10k_ub)

Ground Truth		Adam		EM	
Class 2	Class 3	Class 2	Class 3	Class 2	Class 3
9.000	0.500	8.625	-0.389	5.757	4.216
-0.398	-2.671	-0.302	-2.693	1.144	1.212
3.239	1.383	3.116	1.121	1.878	2.017
1.999	-0.243	1.901	-0.51	1.805	1.65
0.128	-2.361	0.296	-2.353	1.501	1.312
-1.861	-0.632	-1.845	-0.434	-1.287	-1.494
-2.249	1.103	-2.244	1.282	-2.475	-2.478
2.223	1.620	2.17	1.578	0.946	0.837

We first examine the unbalanced class scenario with 10k data. Compared to the more balanced class scenario in group I, both EM and Adam take a longer time to converge (S3N10k_ub vs S3N10k). But EM takes 5 times longer than Adam (8.6 vs 1.4 minutes), while their estimation is similar in the balanced class scenario.

A more severe problem with EM is that it fails to recover the true parameters. All EM runs get stuck in local maxima with large parameter errors. Adam is more robust and obtains more accurate parameter estimates. EM's final log-likelihood is -4059 compared to Adam's -3704 and DGM's -3753. EM's γ error is 1.2 and β error is 10.1, compared to Adam's 0.18 and 0.21. Where do EM's large parameter error come from? Table 7.4 and Table 7.5 show the coefficients estimated by Adam and EM compared to the ground-truth. Because the class ID can be arbitrarily swapped, for all model runs, we reorder the class to match the original class ID in the ground-truth model. The matching is primarily based on class membership shares. If two class shares are close, we match the classes by matching class-specific choice model parameters.

According to the ground truth, class 2 is the largest class with a share of 89%. EM does relatively well on this large class' choice model estimates. But for the two small classes 1 and 3, EM's estimates of choice model coefficients are far off for the majority of the parameters. In particular, it produces very large coefficients for the

Table 7.5: Class-specific Choice Model Coefficients: EM, Adam and the Ground Truth

	Ground Truth			Adam			EM		
	j=1	j=2	j=3	j=1	j=2	j=3	j=1	j=2	j=3
Class 1	0.0	0.861	3.187	0.0	0.406	2.539	0.0	0.032	0.513
	-1.672	-2.622	1.374	-1.2	-2.19	0.893	-0.118	0.402	-0.403
	-1.729	-0.384	-0.264	-1.861	-0.631	-0.092	-0.511	-0.494	-0.425
	-4.023	0.122	-0.128	-3.13	0.502	0.25	-0.487	0.143	-0.404
	-0.609	2.186	5.399	-0.152	1.773	4.08	0.433	-0.583	0.498
	-2.051	-2.102	1.127	-1.285	-1.705	0.662	-0.764	0.143	0.34
	1.255	2.751	3.102	1.106	1.891	2.669	-0.171	-0.362	0.908
	-0.695	-0.556	1.246	-0.363	-0.375	0.867	-1.402	0.079	0.827
	-0.624	-2.232	5.328	-0.588	-1.376	4.431	-0.25	-0.335	0.424
	4.074	-1.191	0.923	3.563	-1.196	0.375	0.553	-0.099	-0.621
	-1.55	-2.947	0.42	-1.713	-2.407	-0.176	0.467	-0.977	-0.322
Class 2	0.0	0.877	0.146	0.0	0.874	0.097	0.0	1.034	0.106
	2.984	-0.195	-1.512	3.001	-0.178	-1.456	3.054	-0.099	-1.653
	2.539	0.387	-0.171	2.462	0.512	-0.14	2.47	0.543	-0.148
	1.652	-0.337	-0.17	1.699	-0.342	-0.147	1.673	-0.326	-0.147
	1.536	0.287	-1.763	1.528	0.386	-1.783	1.477	0.509	-1.791
	0.851	0.193	-0.649	0.753	0.181	-0.666	0.741	0.291	-0.7
	3.462	-1.095	-0.316	3.494	-1.017	-0.399	3.463	-1.065	-0.324
	2.735	2.228	-0.52	2.733	2.287	-0.454	2.653	2.224	-0.411
	-0.674	-0.671	0.486	-0.738	-0.675	0.504	-0.678	-0.675	0.659
	2.713	-0.529	1.202	2.701	-0.583	1.233	2.797	-0.475	1.545
	-1.931	2.258	0.95	-1.894	2.266	0.984	-1.865	2.411	0.992
Class 3	0.0	0.379	1.053	0.0	0.243	1.283	0.0	5.064	-4.527
	0.367	1.681	-1.983	0.504	1.97	-2.188	68.245	-14.707	-17.442
	-1.032	-1.824	-1.498	-0.968	-1.852	-1.69	59.954	11.443	-4.429
	-0.613	-0.336	-1.791	-0.233	-0.083	-1.985	43.608	-10.707	-4.159
	1.007	-2.771	-0.408	1.045	-2.497	-0.517	41.927	-3.701	-45.819
	-2.006	1.41	0.866	-1.931	1.675	0.918	20.88	-10.241	-17.207
	-0.917	-2.064	2.213	-0.661	-2.105	2.461	95.964	-20.253	-15.941
	-4.406	0.707	2.929	-4.796	0.514	2.848	79.442	63.673	-14.433
	-0.449	-0.592	-0.107	-0.541	-0.641	-0.258	-24.985	-18.592	-0.971
	0.462	0.094	-1.855	0.57	-0.104	-1.794	61.721	-29.41	0.61
	2.577	-2.452	-1.406	2.242	-2.555	-1.662	-52.852	43.812	24.048

Table 7.6: Class Membership and Choice Prediction Accuracy (Adam v.s. EM)

		Train data			Hold-out data		
		Adam	EM	Truth	Adam	EM	Truth
Class prediction							
	Accuracy	0.945	0.892	0.945	0.958	0.901	0.96
	Precision	0.821	0.395	0.821	0.876	0.411	0.873
	Recall	0.748	0.521	0.746	0.788	0.53	0.784
	F1 sore	0.781	0.431	0.78	0.819	0.449	0.817
Choice prediction							
	Accuracy	0.842	0.83	0.84	0.837	0.815	0.838
	Precision	0.84	0.827	0.838	0.835	0.813	0.837
	Recall	0.839	0.826	0.837	0.835	0.812	0.837
	F1 sore	0.839	0.827	0.837	0.835	0.812	0.837

Table 7.7: Confusion Matrix for Class Membership Prediction (Adam v.s. EM)

		Adam (Predicted)				EM (Predicted)		
		class 1	class 2	class 3		class 1	class 2	class 3
	class 1	210	120	30	class 1	211	149	0
True	class 2	50	8756	121	class 2	214	8712	1
	class 3	25	203	485	class 3	513	200	0

choice model of class 3. Compared to EM, Adam's choice model coefficients are more accurate for all three classes. Regarding the class membership model coefficients, EM gives large errors for the majority of the parameters, while Adam's estimates are well aligned with the true parameters.

The prediction performance for Adam is startlingly better than EM, especially in class membership assignment. Table 5 shows accuracy metrics for latent class prediction and choice prediction for both training data and hold-out data. Prediction is based on maximum predicted probability. Adam is able to estimate the small classes accurately. All accuracy measures for both class membership and choice prediction are similar to the true model. EM's prediction is less accurate. For class membership prediction, EM is only 89% accurate compared to Adam (95%). EM's precision, recall and F1 score is much lower than Adam. This is because EM misclassifies the examples that are most likely in class 3 to class 1 and 2. From the confusion matrix in Table 7.7, it is clear that EM predicts almost no membership in class 3. Basically class 3 has almost zero membership. This explains why the choice model of class 3 has abnormally large parameter estimates. Interestingly, EM is also less accurate in choice prediction than Adam, but by a far lower magnitude than its mistake on class membership prediction.

b) S3N50k_ub

To test whether the unbalanced shares of latent classes cause the difficulty for EM or the sample size of the smaller classes, we increase the total sample size to 50k, with 1911, 44648, 3441 observations for each latent class, compared to 382, 8929, 688 observations in the 10k data. The underlying model is the same. We find with increased sample size, EM is able to identify the three classes and obtain more accurate parameter estimates than Adam. Yet EM takes 28 minutes compared to Adam's 1.2 minutes. When sample size increases from 10k to 50k, Adam's time does not change, while EM's time triples from 9 minutes to 28 minutes. This again shows that Adam estimation time does not scale with sample size as much as EM. It is majorly affected by the complexity of the model.

The real difficulty for EM to estimate LCCM with unbalanced classes is in not

having a large enough number of observations in the small classes. This situation can be common in practice for two reasons. First, heterogeneous preference may not be evenly distributed in the population. There can be a majority preference group along with small classes with uncommon or extreme preferences. Secondly, with an increasing number of classes (here even with only 3), it is more likely to obtain unbalanced class shares. When total sample size is not sufficiently big, the smallest class would have a relatively small number of observations.

Based on the synthetic data experiments, I find EM is more prone to the small class vanishing problem. In the 10k dataset case, each choice model has 32 parameters to estimate, and the smallest class has 382 examples. The smallest class' sample–to–parameter ratio is about 11. Small class 1 and 3 collapse in EM estimation results.

Why does SGD estimate small class more accurately than EM?

The main reason, we suspect, for SGD's performance advantage lies in the difference between mini-batch and batch training. EM uses the entire dataset. With unbalanced classes and random initialization, most observations will have large prior probability in the big class after a few iterations. For small class examples to have large posterior probability in their own class, they either have to have a large prior probability in the small class or large choice likelihood conditional on their true class in the beginning. Without good initial parameters – either good class membership model parameters that assign small class examples correctly or good class-specific choice model parameters that yield distinctive conditional likelihood for the small class examples – it will be difficult to generate correct posterior probability for small-class examples. In the M-step, since posterior probabilities are used to weight log-likelihood conditioning on different classes, small classes may not have enough posterior membership and not be counted as a significant mode in the total log-likelihood function. The less accurate parameter updates in the M-step for small class is further fed back to the E-step, and make it even more difficult for small class examples to have the correct posterior in the next iteration. Therefore, small classes usually degenerate in EM without good initialization.

The SGD algorithm handles this problem better, perhaps because it uses small

random samples (e.g., 128 data points) to estimate the gradient and update parameters. The noise from small samples allows for more exploration, and prevents the majority class from quickly dominating. SGD is known to be less prone to over-fitting. Each mini-batch to the algorithm is like new data; the algorithm learns gradually and tries to incorporate new information. This may explain why SGD is able to estimate small classes well.

7.4 Summary

I find the stochastic gradient descent algorithm Adam has significant advantages for LCCM estimation. First, Adam scales to *large data* better than BFGS or EM. On synthetic datasets, estimation time by BFGS grows in proportion to sample size; EM's estimation time increases more than proportionally to sample size; while Adam's estimation time scales less than proportionally. This results in large differences in estimation time on large datasets. For example, EM takes 70 times as long as Adam (69 vs 2 minutes) on a 50k data with 10 classes; and 56 times as long as Adam on a 100k data with 10 classes (140 vs 2.5 minutes).

Second, Adam is more time-efficient and stable under *complex model* scenarios, such as with a large number of classes and parameters and unbalanced class membership. When the number of latent classes increases from 3 to 10, with the total number of parameters increasing from 112 to 509, BFGS fails to converge after two days. EM takes about 60 to 70 times as long as Adam, with large variations across runs. Adam's estimation time is shorter with low variability.

Thirdly, Adam obtains more accurate parameter estimates under certain difficult scenarios. When the model is complex (e.g.10-class) and the sample size is not big enough (e.g. 50k), Adam is less prone to local optima than EM. However, if EM converges to global optimum, it obtains similar or more precise parameter estimates than Adam. When class membership is highly unbalanced, and the smaller classes do not have sufficient observations, EM frequently fails to identify small classes, generates large parameter bias, and predicts class membership poorly. Adam is more robust.

It is able to estimate small classes and predict class membership more accurately.

The superior performance of the SGD algorithm is attributed to its avoidance of second-order derivatives (better scalability to high-dimensional problems) and mini-batch training (scalability to large data and less prone to over-fitting).

Chapter 8

LCCM with Flexible Class Membership Model: A Case Study of Swissmetro Mode Choice

I apply LCCM-net to model mode choice using Swissmetro dataset. I first estimate a basic LCCM with a logit class membership model and linear class utilities. Then I extend LCCM by adding nonlinearity to the class membership model. This nonlinear-LCCM outperforms the basic LCCM in out-of-sample prediction accuracy. It also provides interpretable results for class-specific choice models. According to its better predictability and interpretable results, the nonlinear-LCCM learns a more flexible discrete mixing distribution than the basic LCCM with linear class membership utility functions.

8.1 Data

The data used come from the same Swissmetro dataset, processed in the same way as described in section 5.2.

Table 8.1: Description of Variables in the Swissmetro Dataset

Alternative	Alternative attributes	Availability
TRAIN	time, headway, cost (train_tt, train_hw, train_co)	train_av
SM (Swissmetro)	time, headway, seats[a], cost (sm_tt, sm_hw, sm_seats, sm_co)	sm_av
CAR	time, cost (car_tt, car_co)	car_av
Person/Trip variable	**Variable levels**	
AGE	0: age $\leq$ 24, 1: 24 $<$ age$\leq$30, 2: 39 $<$ age $\leq$ 54, 3: 54 $<$ age $\leq$ 65, 4: 65$<$age	
MALE	0: female, 1: male	
INCOME (thousand CHF per year)	0: under 50, 1: between 50 and 100, 2: over 100, 3: unknown	
FIRST (First class traveler)	0: no, 1: yes	
GA (Swiss annual season ticket)	0: no GA, 1: owns a GA	
PURPOSE	0: Commuter, 1: Shopping, 2: Business, 3: Leisure	
WHO (Who pays)	0: self, 1: employer, 2: half-half	
LUGGAGE	0: none, 1: one piece, 2: several pieces	

a. Seats configuration in Swissmetro: seats=1 if airline seats, 0 otherwise.

8.2 Baseline LCCM

In the baseline LCCM, choice models are class-specific. Choice model inputs include all the attributes of each alternative (see Table 8.1). The choice model is specified as a logit model with linear utility functions (Eqn. 8.1). The coefficients of attributes are alternative-specific. Cost coefficients are fixed to -1, so that all coefficients in the utility functions are interpreted as willingness-to-pay measured by monetary unit (CHF: Swiss Franc). Alternative specific constants (ASCs) are included for TRAIN and SM.

The class membership model is a logit model with linear utilities. Explanatory variables include all available characteristics (AGE, MALE, INCOME, FIRST, WHO, LUGGAGE, PURPOSE and GA). These variables are categorical. We use 0-1 encoding and choose level 0 as the reference level. This leads to 17 dummy variables as inputs to the class membership model. All classes share the same set of inputs. One class is the reference, with coefficients fixed to 0. Intercepts are included for all classes except for the reference class.

$$V^s_{TRAIN} = ASC^s_{TRAIN} + \beta^s_{TRAIN\_TT} TRAIN\_TT + \beta^s_{TRAIN\_HE} TRAIN\_HE - TRAIN\_CO$$

$$V^s_{SM} = ASC^s_{SM} + \beta^s_{SM\_TT} SM\_TT + \beta^s_{SM\_HE} SM\_HE + \beta^s_{SM\_SEATS} SM\_SEATS - SM\_CO$$

$$V^s_{CAR} = \beta^s_{CAR\_TT} CAR\_TT - CAR\_CO \tag{8.1}$$

Parameter constraints

We find that adding parameter constraints is necessary to obtain interpretable model results in the Swissmetro case. Coefficients for time and headway should be non-positive in common situations. Without parameter constraints, we obtain coefficients with counter-intuitive signs (e.g. positive values of time) with both traditional estimation and neural network estimation. Train (2008) encounters the same issue in an empirical study and suggests constraining parameters in future studies.

To include parameter constraints to LCCM-net, we add a penalty to the average negative log-likelihood loss, which incurs a positive cost when constraints are violated (Eqn.8.3). We choose a linear penalty with a Rectified Linear Unit (ReLU) (Eqn.8.2), a nonlinear transformation function.

Suppose parameter b has the constraint: $b_{min} \leq b \leq b_{max}$. If b is less than b_{min}, the penalty will $b_{min} - b$ according to the value of ReLU function; otherwise, the penalty is zero. Similarly, the penalty for exceeding the maximum value is linear with respect to the size of the violation.

The constraint violation penalty λ is a tune-able hyper-parameter that controls the trade-off between the NLL loss and constraint violation loss. If it is too small, the constraint may not be strictly enforced. We impose non-positive constraints on time and headway coefficients. In our case, a penalty of 0.1 is enough to keep the

constraints satisfied.

Estimation

LCCM-net is implemented in PyTorch. We train it in two steps. Table 8.2 shows a list of hyper-parameters related to optimization. In the first SGD step, we use Adam algorithm to get close to convergence. We set the batch size to 128 and the learning rate at 0.01. The algorithm stops until the change in NLL loss on development data between two consecutive epochs is smaller than 0.001 (nll_tol_near), or a maximum 500 number of epochs is reached. We choose development NLL change instead of training NLL to prevent over-fitting on training data.

In the batch gradient descent (BGD) step, we increase batch size to the entire training dataset size. Training stops if the NLL change is less than 1e-5, or a maximum number of 500 epochs is reached. Each model scenario is estimated 5 times with a different random initialization. Out of the 5 runs, we choose the one with the lowest NLL on development data for analysis.

$$ReLU(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{8.2}$$

$$Loss = NLL + \lambda[\sum_{b \in R} ReLU(b_{min} - b) + \sum_{b \in R} ReLU(b - b_{max})] \tag{8.3}$$

We also estimate the baseline LCCM using *PandasBiogeme* and the Python *lccm* package. We include parameter sign constraints in both methods. *PandasBiogeme* uses L-BFGS-B to allows for parameter bounds. For the Python *lccm* package, the optimizer in M-step by default is BFGS. We switch it to L-BFGS-B to allow for parameter sign constraints. The L-BFGS-B algorithms employed in both packages are from the Scipy library scipy.optimize.minimize.

Table 8.2: Hyper-parameters for LCCM-net Optimization

Method	Hyper-parameter	Description	Value
Adam (PyTorch)			
SGD step	batch_size	size of each mini-batch	128
	lr	learning rate of SGD	0.01
	ftol_near	stop if loss function change below this threshold	0.001
	max_epochs	maximum SGD epochs	500
BGD step	batch_size_BGD	size of training data	7484
	lr_BGD	learning rate of BGD	0.01
	ftol	stop if loss function change below this threshold	1e-5
	max_epochs_BGD	maximum BGD epochs	500

8.3 Baseline LCCM Estimation Results

We vary the number of latent classes from 1 to 10, and estimate the LCCM 5 times with different random initialization. Both EM and direct maximum likelihood give some abnormally large parameter estimates when the number of latent classes is more than 3. Although we tried many different specifications and added more constraints, this problem persists as long as the number of classes gets large. Based on our synthetic data experience, this is likely due to a relatively small number of observations in some of the classes. With Adam, we are able to estimate up to 10 classes, with coefficient estimates within reasonable range and no small class vanishing. Both DML and EM take longer time than Adam to converge. For example, to estimate an 8-class model, PandasBiogeme needs 3 to 4 hours. EM takes about 40 to 60 minutes, while Adam takes less than 1 minute. The following results are based on the neural network estimation of LCCM.

Table 8.3 shows AIC, BIC and log-likelihood on development data against the number of latent classes. We find AIC is lowest with 8 latent classes; and BIC is lowest with 3 classes. BIC is known to have a stronger penalty on model complexity. We further examine log-likelihood on development data, which evaluates the modelâĂŹs generalization performance. It is also the lowest with 8 latent classes. So we choose 8 as the optimal number of classes. As expected, the predicted class membership shares

Table 8.3: AIC, BIC and Log-likelihood (LL) of LCCM

Number of Classes	AIC	BIC	Training LL	Development LL
1	12072	12127	-6028	-1244
2	10978	11206	-5456	-1140
3	10606	11007	-5245	-1111
4	10406	10981	-5120	-1098
5	10317	11065	-5051	-1094
6	10279	11199	-5026	-1085
7	10321	11415	-5004	-1085
8	10246	11512	-4940	-1068
9	10339	11779	-4962	-1072
10	10319	11931	-4926	-1076

Table 8.4: Predicted Class Membership (LCCM with 8 latent classes)

Number of Classes	Counts	Share
1	4973	46.5%
2	254	2.4%
3	920	8.6%
4	1910	17.9%
5	998	9.3%
6	382	3.6%
7	1039	9.7%
8	217	2.0%

are unevenly distributed (Table 8.4). The largest class (class 1) and second largest class (class 4) have 47% and 18% of the population, respectively. The rest of latent classes have small shares ranging from 2% to 10%.

Table 8.5 and Table 8.6 show the estimated coefficients and standard errors for the choice models and the class membership model. Standard errors are computed using asymptotic formula (square root of the diagonals of inverse Hessian). Figure 8-2 visualizes the estimated value of time, value of headway, and other choice model coefficients for each of the 8 classes.

Class 1 is the largest class with a share of 47%. For this class, the marginal disutility of SM and CAR travel time is similar, about -2 CHF/min. The marginal

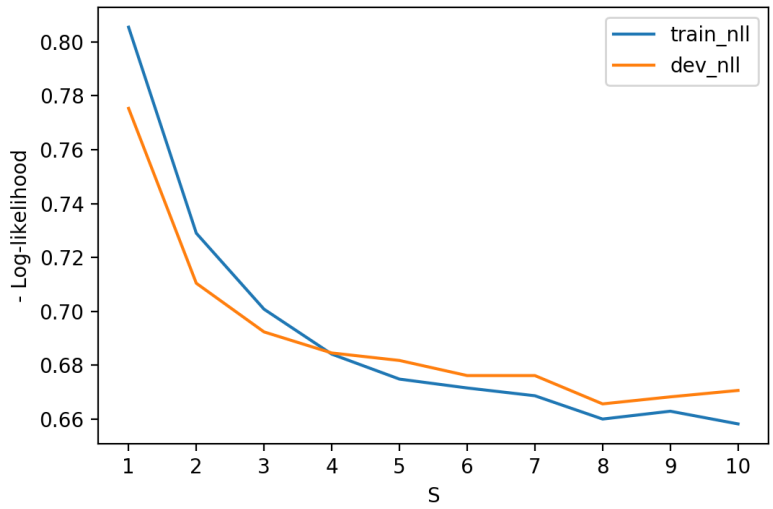


Figure 8-1: Average Negative Log-likelihood v.s. Number of Latent Classes (Baseline LCCM)

disutility of train travel time is higher (-2.76 CHF/min). Class 1 is more sensitive to SM headway time than TRAIN headway time. Class 5's travel time has similar marginal effect compared to class 1. Different from class 1, class 5 is more sensitive to TRAIN headway than SM headway.

Classes 3 and 4 stand out as the groups that strongly dislike car travel time, and are insensitive to SM time. The difference between the two is that class 4 favors SM travel time more than TRAIN; while class 3 is indifferent between SM and TRAIN travel time.

Class 3 and 4 also have opposite preferences towards headway time. Although class 3 is indifferent to in-vehicle time by TRAIN or SM, it is the most sensitive class to SM headway time. Class 4 is more sensitive to both travel time and headway for TRAIN than those for SM. The ASCs also show that class 4 prefers SM to TRAIN, while class 3 is the opposite. Besides class 3 and 4, class 8 is another class insensitive to SM travel time.

Classes 2 and 6 are the most indifferent to CAR travel time and most sensitive to SM travel time. Class 6 is more sensitive to TRAIN travel time than class 2. Class 7 generally has smaller VOT. People in this class dislike TRAIN travel time more than

SM and CAR. Class 7 is the most supportive for airline seating in SM.

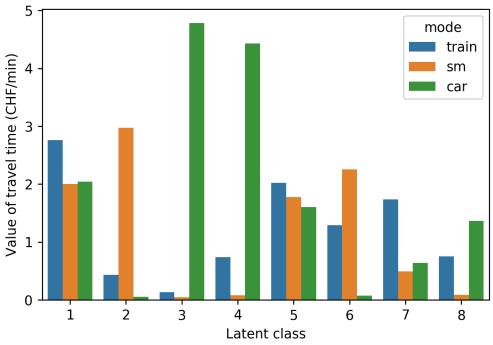
To understand what type of individuals are more likely to belong to which class, we plot the predicted class membership shares conditioning on person characteristics in Figure 8-3 (a). Each row represents the predicted shares of latent classes given one population segment in the Swissmetro dataset. The darker the color, the higher the class share. According to the predicted class membership shares conditioning on population segment, people with an annual seasonal ticket (GA=1) are more likely to be in class 3 and 4 than those without. They are not sensitive to TRAIN and SM travel time, and strongly disfavor car travel time. Young people (AGE_0: age$\leq$ 24) are more likely to be in class 3, insensitive to SM or TRAIN travel time, in contrast with other age groups. The oldest group has a more diverse class distribution. The majority of Business trips (PURPOSE_2) concentrate in the largest class 1, but other trip purposes have class shares distributed in other classes.

8.4 LCCM Extension: Flexible Class Membership Model

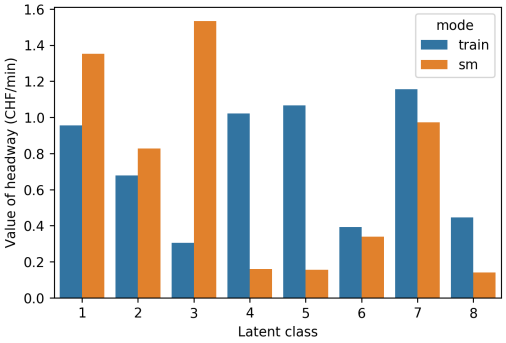
The extended LCCM has one hidden layer in the class membership network and nonlinear transformation on the hidden units. We call this model nonlinear-LCCM. We want to test whether the class membership part of the neural network can learn a better mixing distribution and predict choice more accurately.

A summary of the hyper-parameters tried is listed in Table 8.7. In addition to the number of latent classes, choices must be made about the number of hidden layers, the hidden layer size, the activation function, and the $l2$ regularization penalty (on class membership model parameters).

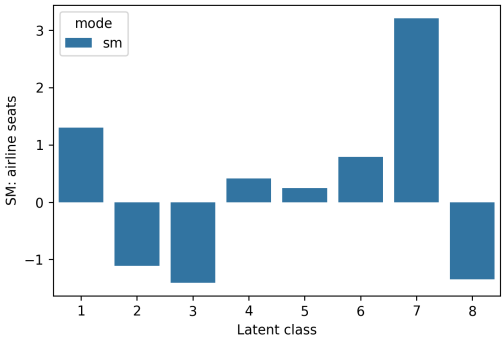
We try 1 and 2 hidden layers for the class membership model. Since a model with 2 hidden layers does not show improvement in prediction performance, we focus on model scenarios with only 1 hidden layer. We experiment with both the *relu* and *tanh* activation functions. The prediction performance is about the same. However,



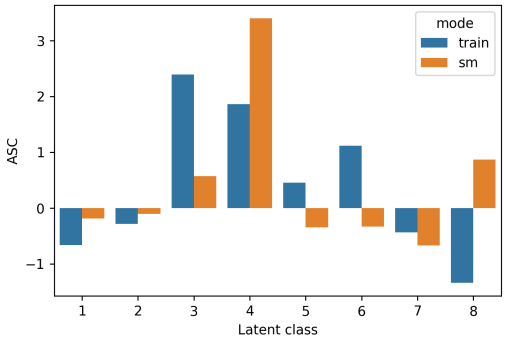
(a) Values of Travel Time



(b) Values of Headway

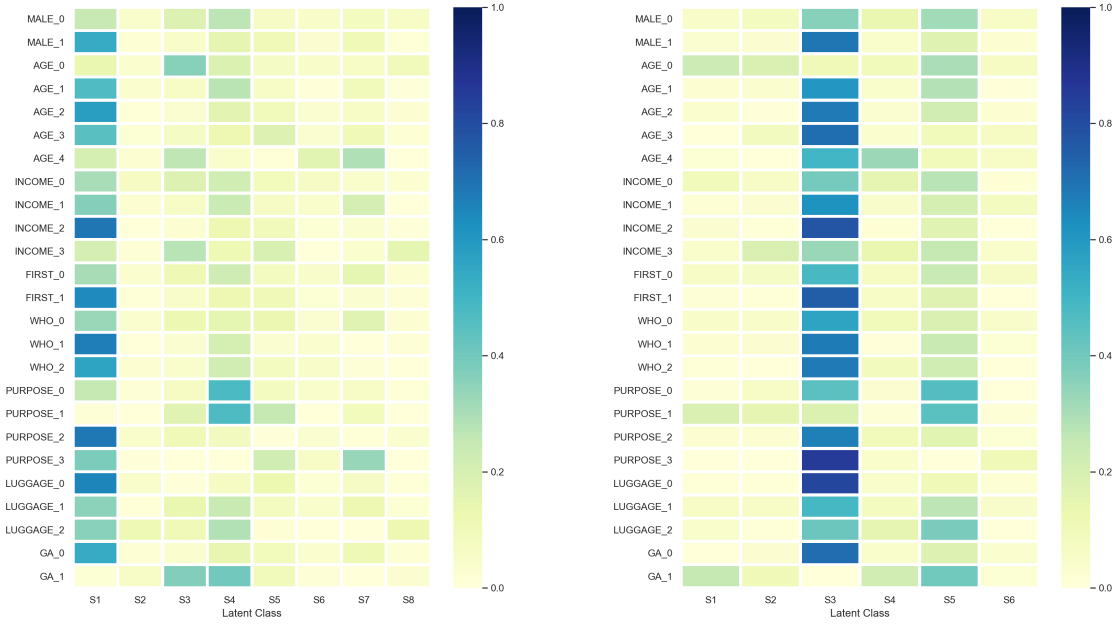


(c) SM Airline Seating Coefficients



(d) Alternative Specific Constants

Figure 8-2: Values of Travel Time, Values of Headway and Other Choice Model Coefficients by Latent Class (LCCM with 8 classes)



(a) LCCM (b) nonlinear-LCCM

Figure 8-3: Predicted Class Shares by Individual Characteristics

Table 8.5: Class-Specific Choice Model Coefficients (LCCM with 8 Latent Classes)

	class1	class2	class3	class4	class5	class6	class7	class8
TRAIN_TT	-2.761	-0.431	-0.134	-0.741	-2.022	-1.287	-1.737	-0.749
	(0.147)	(0.043)	(0.043)	(0.091)	(0.064)	(0.048)	(0.119)	(0.101)
TRAIN_HE	-0.956	-0.679	-0.305	-1.021	-1.066	-0.392	-1.155	-0.446
	(0.209)	(0.081)	(0.094)	(0.171)	(0.105)	(0.077)	(0.182)	(0.172)
TRAIN_ASC	-0.663	-0.285	2.394	1.859	0.453	1.117	-0.432	-1.336
	(0.225)	(0.092)	(0.106)	(0.173)	(0.115)	(0.093)	(0.194)	(0.186)
SM_TT	-2.002	-2.973	-0.047	-0.081	-1.776	-2.258	-0.494	-0.086
	(0.065)	(0.169)	(0.061)	(0.148)	(0.061)	(0.127)	(0.059)	(0.079)
SM_HE	-1.352	-0.828	-1.534	-0.16	-0.156	-0.339	-0.972	-0.142
	(0.35)	(0.537)	(0.44)	(0.686)	(0.316)	(0.468)	(0.333)	(0.45)
SM_SEATS	1.301	-1.114	-1.405	0.412	0.248	0.792	3.207	-1.345
	(0.118)	(0.143)	(0.186)	(0.181)	(0.085)	(0.091)	(0.157)	(0.095)
SM_ASC	-0.189	-0.103	0.57	3.397	-0.344	-0.332	-0.672	0.868
	(0.095)	(0.152)	(0.107)	(0.187)	(0.086)	(0.134)	(0.089)	(0.118)
CAR_TT	-2.039	-0.053	-4.783	-4.43	-1.602	-0.076	-0.636	-1.363
	(0.021)	(0.018)	(0.383)	(0.518)	(0.019)	(0.019)	(0.017)	(0.035)

Table 8.6: Class Membership Model Coefficients (LCCM with 8 Latent Classes)

	Class2	Class3	Class4	Class5	Class6	Class7	Class8
ASC	0.131	-0.601	1.368	-0.589	-0.316	-0.7	-0.59
	(0.517)	(0.381)	(0.275)	(0.364)	(0.402)	(0.385)	(0.522)
MALE							
1: male	-1.215	-1.213	-0.723	0.844	-1.106	0.717	-2.418
	(0.21)	(0.142)	(0.121)	(0.164)	(0.177)	(0.159)	(0.256)
AGE							
1: 24 $<$ age$\leq$30	0.09	-1.883	-0.234	-1.119	-3.574	-2.018	-3.51
	(0.382)	(0.25)	(0.244)	(0.317)	(0.446)	(0.357)	(0.474)
2: 39 $<$ age $\leq$ 54	-0.625	-1.371	-0.317	0.065	-1.11	-1.886	0.438
	(0.427)	(0.27)	(0.253)	(0.325)	(0.367)	(0.368)	(0.346)
3: 54 $<$ age $\leq$ 65	0.458	0.034	0.407	0.896	-0.696	-1.376	-0.475
	(0.441)	(0.276)	(0.268)	(0.332)	(0.38)	(0.378)	(0.404)
4: 65 $<$ age	0.734	1.533	0.274	-2.838	2.171	1.417	-5.213
	(0.518)	(0.319)	(0.351)	(0.54)	(0.376)	(0.39)	(2.768)
INCOME							
1: between 50 and 100	-0.318	0.344	0.587	-0.48	0.454	2.919	-2.202
	(0.237)	(0.187)	(0.156)	(0.193)	(0.234)	(0.22)	(0.512)
2: over 100	-3.198	-1.4	-0.949	-1.098	-0.961	0.232	-2.047
	(0.563)	(0.231)	(0.179)	(0.204)	(0.259)	(0.244)	(0.407)
3: unknown	-1.758	1.62	-0.059	1.858	-3.863	0.261	2.847
	(0.492)	(0.216)	(0.226)	(0.246)	(1.049)	(0.357)	(0.296)
FIRST							
1: first-class	-1.158	-0.949	-0.987	0.405	-1.749	-2.02	0.127
	(0.253)	(0.159)	(0.118)	(0.136)	(0.184)	(0.168)	(0.264)
WHO PAY							
1: employer	-4.141	-1.352	-0.645	-1.682	1.089	-4.506	-0.707
	(0.626)	(0.176)	(0.132)	(0.19)	(0.221)	(0.534)	(0.275)
2: half-half	0.116	-1.551	-0.896	-0.978	0.854	-2.69	-3.659
	(0.331)	(0.253)	(0.197)	(0.231)	(0.267)	(0.356)	(1.813)
PURPOSE							
1: shopping	-0.667	2.692	2.299	4.263	-1.986	2.393	-0.639
	(1.566)	(0.345)	(0.321)	(0.346)	(1.186)	(0.369)	(1.325)
2: business	0.001	-1.986	-3.113	-4.354	-1.527	-4.186	-0.528
	(0.328)	(0.19)	(0.143)	(0.307)	(0.226)	(0.269)	(0.377)
3: leisure	-2.434	-5.667	-6.365	0.169	-0.297	-0.72	-4.134
	(0.489)	(0.624)	(0.508)	(0.195)	(0.265)	(0.206)	(0.807)
LUGGAGE							
1: one piece	-2.388	2.907	1.421	-0.734	1.437	1.834	0.505
	(0.295)	(0.267)	(0.127)	(0.139)	(0.186)	(0.145)	(0.28)
2: several pieces	-0.402	1.501	1.08	-2.17	-1.917	-1.038	0.84
	(0.393)	(0.479)	(0.329)	(0.863)	(2.654)	(2.615)	(0.45)
GA							
1: owns a GA	5.099	4.575	3.384	3.491	1.356	-2.02	3.388
	(0.371)	(0.259)	(0.252)	(0.288)	(0.39)	(1.09)	(0.349)

Table 8.7: Model Structure Related Hyper-parameters

Structure-related	Hyper-parameter	Description	Range
LCCM	S	number of latent classes	[1,2,..,10]
	λ	constraint violation penalty	0, 0.1
nonlinear-LCCM	S	number of latent classes	[5,6..,10]
	H	hidden layer size	[10,20,...,100]
	A	nonlinear activation function	*tanh*, *relu*
	$l2$	$l2$ regularization penalty (class membership model parameters only)	0, 1e-5, 1e-4
	λ	constraint violation penalty	0, 0.1

relu tends to have collapsed classes (some classes are empty), perhaps due to the sparse activation of the *relu* function, while *tanh* activation gives more stable and regular class membership probability distributions. So we choose *tanh* for the rest of the analysis.

For nonlinear-LCCM, we vary the number of latent classes from 5 to 10; and for each number of latent classes, we vary the hidden layer size from 10 to 100 hidden units with a step size of 10. To prevent potential over-fitting, we also try $l2$ regularization on class membership model parameters with strength 0.0001 and 0.00001.

Among all the hyper-parameter scenarios in Table 8.7, the best model, based on the lowest NLL loss on the development dataset, has 6 classes, one hidden layer of 50 hidden units, and no regularization. This nonlinear-LCCM achieves better prediction performance on hold-out datasets (Table 8.8). Compared to the basic LCCM, the average NLL for the development data decreases from 0.666 to 0.638; and the average NLL on the test data decreases from 0.663 to 0.628. The F1 score improves from 0.60∼0.61 to 0.66∼0.67 on hold-out data. This result shows that a more flexible class membership estimated via neural network significantly improves choice prediction accuracy.

Table 8.9 shows the predicted shares of the latent classes. Class 3 and 5 are the largest classes. Shares of other classes range from 3.5% to 6.5%. Table 8.10 and Figure 8-4 show the value of time, value of headway and other choice model coefficients for the 6 latent classes.

Compared to the 8-class LCCM, the majority class 3 of the nonlinear-LCCM

Table 8.8: Prediction Performance of the Best LCCMs

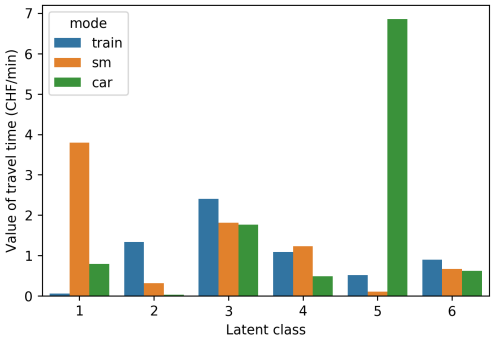
		NLL			ACC			F1		
S	Hidden size	train	dev	test	train	dev	test	train	dev	test
8		0.66004	0.66567	0.66328	0.703	0.708	0.694	0.620	0.613	0.602
6	50	0.59149	0.63842	0.62804	0.741	0.732	0.724	0.690	0.671	0.664

Table 8.9: Predicted Class Membership Shares

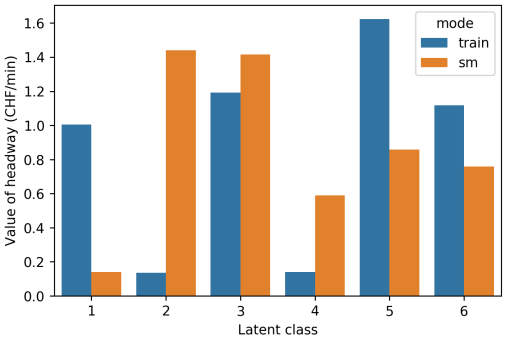
Number of Classes	Counts	Share
1	413	3.9%
2	437	4.1%
3	6535	61.1%
4	690	6.5%
5	2245	21.0%
6	373	3.5%

Table 8.10: Class-specific Choice Model Coefficients (nonlinear-LCCM)

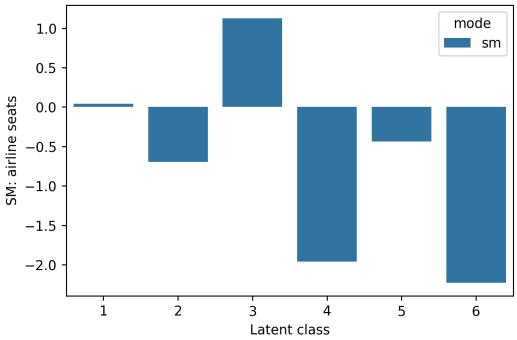
	class1	class2	class3	class4	class5	class6
TRAIN_TT	-0.06	-1.34	-2.41	-1.09	-0.52	-0.89
TRAIN_HW	-1.01	-0.14	-1.19	-0.14	-1.62	-1.12
TRAIN_ASC	1.14	3.08	-0.65	2.26	1.48	-2.32
SM_TT	-3.80	-0.32	-1.81	-1.23	-0.10	-0.67
SM_HW	-0.14	-1.44	-1.42	-0.59	-0.86	-0.76
SM_SEATS	0.04	-0.70	1.12	-1.96	-0.44	-2.23
SM_ASC	2.33	1.13	-0.18	-1.65	2.94	-1.51
CAR_TT	-0.79	-0.03	-1.76	-0.48	-6.86	-0.62



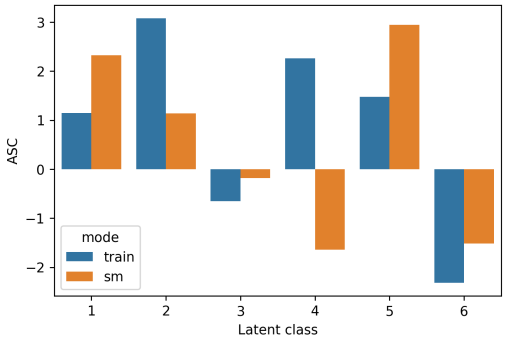
(a) Values of Travel Time



(b) Values of Headway



(c) SM Airline Seating Coefficients



(d) Alternative Specific Constants

Figure 8-4: Values of Travel Time, Values of Headway and Other Choice Model Coefficients by Latent Class (nonlinear-LCCM with 6 classes)

corresponds to the majority class 1 in the LCCM. The marginal disutility of travel time is similar between SM and CAR, and lower than TRAIN.

Class 5 has the largest disutility of CAR travel time, which corresponds to a combination of class 3 and 4 in the 8-class model. Individuals in class 5 are insensitive to SM time. They prefer travel time and headway by SM more than by TRAIN.

Class 1 is insensitive to TRAIN travel time and most sensitive to SM travel time among all classes. However, the marginal disutility of TRAIN headway is much higher than SM.

Class 2 is insensitive to car travel time, and more sensitive to TRAIN time and SM time. Classes 4 and 6 have relatively small values of time for all modes. They differ in their values of headway and alternative specific constants.

Figure 8-3 (b) shows the predicted class shares for each population segment. Most segments have the highest share in class 3, the major class except for a few groups. For example, females have large shares in both class 3 and class 5, a class that strongly dislikes car travel time and is least sensitive to SM time. The youngest group (AGE_0) is more evenly distributed among the 6 classes, with the largest share in class 5. Commute trips and shopping trips also tend to appear in class 5. People with annual seasonal ticket also have the highest share in class 5.

How does nonlinear-LCCM predict better?

The confusion matrix in Table 8.12 displays where the nonlinear-LCCM's prediction gain comes from. The 8-class LCCM mispredicts many choices of TRAIN as SM or CAR. Nonlinear-LCCM reduces such mistakes and improves the recall for TRAIN. Nonlinear-LCCM also improves the recall for CAR, as LCCM tends to mispredict CAR as SM.

We further examine two individual cases: Persons A and B, with characteristics listed in Table 8.11.

Person A's true choice is TRAIN. LCCM mispredicts A's choice as SM while nonlinear-LCCM gives the correct prediction. We check person A's class membership prediction by the two models. It turns out that LCCM assigns person A to class 8 (Figure 8-2), a class insensitive to SM time. Nonlinear-LCCM assigns A to class

Table 8.11: Characteristics of Two Example Persons

	Person A	Person B
MALE	No	Yes
AGE	24 to 30	54 to 65
FIRST	No	No
GA	Yes	No
INCOME	below 50	over 100
LUGGAGE	1 piece	1 piece
PURPOSE	commute	commute
WHO	self	self

Table 8.12: Confusion Matrix of Choice Prediction

		LCCM (Predicted)				nonlinear-LCCM (Predicted)		
		TRAIN	SM	CAR		TRAIN	SM	CAR
	TRAIN	470	850	93	TRAIN	723	618	72
True	SM	230	5219	750	SM	246	5204	749
	CAR	13	1248	1819	CAR	88	1038	1954

1 (Figure 8-4), which is very sensitive to SM travel time and insensitive to TRAIN time. Because nonlinear-LCCM makes a better class assignment, it leads to the correct choice prediction (TRAIN).

Person B's true choice is CAR. LCCM predicts SM while nonlinear-LCCM makes the correct prediction. LCCM assigns person B to class 8 (Figure 8-2), which has a very low marginal disutility of SM travel time. This leads to a higher chance of predicting SM. Nonlinear-LCCM predicts person B in class 3, which has a similar marginal disutility of travel time for SM and CAR. Due to the inaccurate class assignment, LCCM makes the wrong choice prediction.

8.5 Summary

I propose using a neural network to learn a flexible class membership model. In the Swissmetro case study, the neural network learns a better mixing distribution than the manually specified class membership model with linear utilities, and improves choice

prediction accuracy. I show that a hybrid neural network and discrete choice model is promising, since it takes advantage of neural network's function approximation ability, while keeping the economic theory and interpretability of the discrete choice framework.

Compared to traditional estimation methods, SGD algorithms require important decisions on optimization-related hyper-parameters, such as learning rate, batch size etc. To use nonlinear-LCCM, there are more model structure-related hyper-parameters to select, such as the number of hidden layers, hidden layer size, activation function, and regularization penalty. Modellers need to conduct many experiments and search for the best set of hyper-parameters. This requires basic knowledge of neural networks and familiarity with coding. Also, the model becomes less transparent. Nonlinear-LCCM partly loses interpretability due to the black-box class membership model.

Chapter 9

Conclusions & Future Work

9.1 Conclusions

This dissertation takes an initial step towards integrating theory-based and data-driven approaches for discrete choice problems. Based on the strengths and weaknesses of DCMs and neural networks, I propose two hybrid models that can benefit from neural networks' flexibility and training procedures, and DCMs' behavioral theory and interpretability. Through Monte-Carlo experiments and applications to real data, I show that these neural-embedded discrete choice models (NEDCMs) can recover nonlinear utility forms, particularly complex taste variations, and achieve better predictability than misspecified models. Also, by designing special hybrid structures and imposing parameter constraints, I demonstrate the general concept of incorporating expert knowledge into neural networks, to make them more interpretable and, potentially, useful for transportation policy analysis.

9.2 Practical Implications

Through Monte-Carlo experiments and real data applications, I demonstrate that a misspecified DCM can cause large biases in estimated parameters and economic indicators, and low prediction accuracy.

Using a synthetic data (see section 4.4), I show that even a small amount of mis-

specification in MNLs (missing a couple of second-order interactions between socio-economic characteristics), result in prediction accuracy 7 percent points lower than the true model, and 50% to 60% errors in estimated parameters. Values of time estimates can be mistaken by 10%. Demand elasticities and choice probabilities can be off by 55% and 62%, respectively. However, a TasteNet-MNL without *a priori* knowledge can recover the true utility function, achieve similar prediction accuracy as the true model, and generate values of time and demand elasticities close to the truth. TasteNet-MNL also discovers a wider spectrum of taste variations among respondents in the Swissmetro dataset, and suggests on average, higher values of time than MNLs with linear utilities. Nonlinear-LCCM outperforms a typical LCCM in choice predication accuracy using the Swissmetro dataset.

These findings suggest that neural-embedded discrete choice models (NEDCMs) can be beneficial to transport modeling and planning, by reducing potential biases and improving model forecast accuracy.

At the model development stage, NEDCMs can *complement* manually specified DCMs to detect misspecification. For example, modellers can compare a NEDCM and a manually specified DCM's log-likelihood and prediction accuracy on hold-out datasets. If an NEDCM predicts better, the DCM specification can be potentially improved. In practice, I suggest trying both approaches and comparing the economic indicators derived from each, to see where disagreements occur, as a clue to detect data outliers or discover systematic patterns or signals that are ignored.

For model estimation, SGD algorithms can be implemented in software packages to speed up LCCM estimation; and improve estimation accuracy and stability for many-class problems. The improved speed can allow modelers to try many alternative specifications quickly.

For model application, e.g., aggregate demand forecast and policy scenario analysis, NEDCMs can be integrated into trip-based (e.g. a four-step model) or activity-based model systems. For example, certain choice modules, such as the vehicle ownership model or the mode choice model, can have a NEDCM as a second option. The NEDCMs in this dissertation are written in Python using the deep learning platform

PyTorch. To integrate them into a four-step model or a micro-simulation model, we need to build an interface between the two.

9.3 Limitations & Future Work

This research has several limitations that require future work.

First, the current TasteNet-MNL model only accommodates systematic taste variations. Random taste heterogeneity as an important source of heterogeneity can be incorporated in future research. This leads to the intriguing question of how to model distributions of taste parameters with neural networks.

Second, the neural embedded choice models proposed in this dissertation focus on modeling *taste* heterogeneity. Non-linearity in attributes is another important aspect in systematic utility specification. Studies have shown that utility can be a nonlinear function of attributes, such as price. Nonlinear effects, such as the saturation effect and threshold effect have been observed empirically and explained by prospect theory and assimilation-contrast theory. Future work may extend TasteNet-MNL to model nonlinearity in attributes.

Third, the scope of model comparisons in this dissertation is limited. For TasteNet-MNL, other forms of nonlinearity can be examined to see if TasteNet-MNL can capture them. Also, the types of benchmarks for model comparison are limited. Mixed logit structures, such as LCCM, random coefficient logit with other distributional assumptions and/or systematic utilities, and models with non-parametric mixing distributions have not been thoroughly examined. It is inconclusive whether DCMs that incorporate random heterogeneity can predict better or worse than TasteNet-MNL, which only models systematic taste variation. Most likely, it varies across cases, depending on the magnitude of systematic vs random taste variation in the datasets. Future work can conduct a more systematic comparison of TasteNet-MNL and DCMs.

For nonlinear-LCCM, the comparison is limited to an LCCM with a logit class membership model with linear utilities. Although we find this discrete mixing dis-

tribution learned by a neural network improves prediction performance, it is unclear whether this gain comes from a systematic or random effect. More comparisons can be conducted with a variety of mixture models, such as continuous mixed logit, mixed-mixed models, and models with non-parametric mixing distributions (e.g. fixed mass points or endogenous support points), in terms of explanatory power, out-of-sample predictability, and behavioral interpretation.

Lastly, I recommend integrating neural-embedded choice models into transportation model systems. We can compare the more data-driven NEDCMs with manually specified DCMs under different empirical settings, and evaluate the uncertainty in aggregate forecasts (e.g. mode shares) to better inform transportation planning and policy decisions.

Bibliography

M. Abe. Measuring consumer, nonlinear brand choice response to price. *Journal of Retailing*, 74(4):541 – 568, 1998. ISSN 0022-4359. doi: https://doi.org/10.1016/S0022-4359(99)80107-9. URL http://www.sciencedirect.com/science/article/pii/S0022435999801079.

M. Abe. A generalized additive model for discrete-choice data. *Journal of Business & Economic Statistics*, 17(3):271–284, 1999.

D. Agrawal and C. Schorling. Market share forecasting: An empirical comparison of artificial neural networks and multinomial logit model. *Journal of Retailing*, 72(4): 383–407, 1996.

G. M. Allenby and P. E. Rossi. Marketing models of consumer heterogeneity. *Journal of econometrics*, 89(1-2):57–78, 1998.

R. L. Andrews, A. Ainslie, and I. S. Currim. An empirical comparison of logit choice models with discrete versus continuous representations of heterogeneity. *Journal of Marketing Research*, 39(4):479–487, 2002. doi: 10.1509/jmkr.39.4.479.19124. URL https://doi.org/10.1509/jmkr.39.4.479.19124.

P. Bajari, J. T. Fox, and S. P. Ryan. Linear regression estimation of discrete choice models with nonparametric distributions of random coefficients. *American Economic Review*, 97(2):459–463, 2007.

A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18 (153), 2018.

F. Becker, M. Danaf, X. Song, B. Atasoy, and M. Ben-Akiva. Bayesian estimator for logit mixtures with inter- and intra-consumer heterogeneity. *Transportation Research Part B: Methodological*, 117:1 – 17, 2018. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2018.06.007. URL http://www.sciencedirect.com/science/article/pii/S0191261517304824.

M. Ben-Akiva and M. Bierlaire. *Discrete Choice Models with Applications to Departure Time and Route Choice*, pages 7–37. Springer US, Boston, MA, 2003. ISBN 978-0-306-48058-4. doi: 10.1007/0-306-48058-1_2. URL https://doi.org/10.1007/0-306-48058-1_2.

M. Ben-Akiva, D. Bolduc, and M. Bradley. Estimation of travel choice models with randomly distributed values of time. *Transportation Research Record*, 1413:88–97, 1993.

M. Ben-Akiva, D. McFadden, M. Abe, U. Böckenholt, D. Bolduc, D. Gopinath, T. Morikawa, V. Ramaswamy, V. Rao, D. Revelt, and D. Steinberg. Modeling methods for discrete choice analysis. *Marketing Letters*, 8(3):273–286, 1997.

M. Ben-Akiva, D. Mcfadden, K. E. Train, J. Walker, C. Bhat, M. Bierlaire, D. Bolduc, A. Boersch-Supan, D. Brownstone, D. S. Bunch, A. Daly, A. De Palma, D. Gopinath, A. Karlstrom, and M. A. Munizaga. Hybrid choice models: Progress and challenges. *Marketing Letters*, 13(3):163–175, Aug 2002. ISSN 1573-059X. doi: 10.1023/A:1020254301302. URL `https://doi.org/10.1023/A:1020254301302`.

M. Ben-Akiva, D. McFadden, and K. Train. Foundations of stated preference elicitation: Consumer behavior and choice-based conjoint analysis. *Foundations and Trends® in Econometrics*, 10(1-2):1–144, 2019. ISSN 1551-3076. doi: 10.1561/0800000036. URL `http://dx.doi.org/10.1561/0800000036`.

M. E. Ben-Akiva. *Structure of passenger travel demand models.* PhD thesis, Massachusetts Institute of Technology, 1973.

M. E. Ben-Akiva and S. R. Lerman. *Discrete choice analysis: theory and application to travel demand*, volume 9. MIT press, 1985.

Y. Bentz and D. Merunka. Neural networks and the multinomial logit for brand choice modelling: a hybrid approach. *Journal of Forecasting*, 19(3):177–200, 2000.

C. Bhat. Accommodating variations in responsiveness to level-of-service measures in travel mode choice modeling. *Transportation Research, Part A: Policy and Practice*, 32A(7):495–507, 1 1998. ISSN 0965-8564. doi: 10.1016/S0965-8564(98)00011-1.

C. R. Bhat. An endogenous segmentation mode choice model with an application to intercity travel. *Transportation science*, 31(1):34–48, 1997.

C. R. Bhat. Incorporating observed and unobserved heterogeneity in urban work travel mode choice modeling. *Transportation science*, 34(2):228–238, 2000.

M. Bierlaire. Swissmetro, 2018. URL `http://transp-or.epfl.ch/documents/technicalReports/CS_SwissmetroDescription.pdf`.

C. M. Bishop. Mixture density networks. 1994.

J. H. Boyd and R. E. Mellman. The effect of fuel economy standards on the us automotive market: an hedonic demand analysis. *Transportation Research Part A: General*, 14(5-6):367–378, 1980.

A. Bujosa, A. Riera, and R. L. Hicks. Combining discrete and continuous representations of preference heterogeneity: a latent class approach. *Environmental and Resource Economics*, 47(4):477–493, 2010.

G. E. Cantarella and S. de Luca. Multilayer feedforward networks for transportation mode choice analysis: An analysis and a comparison with random utility models. *Transportation Research Part C: Emerging Technologies*, 13(2):121–155, 2005.

N. S. Cardell and F. C. Dunbar. Measuring the societal impacts of automobile downsizing. *Transportation Research Part A: General*, 14(5-6):423–434, 1980.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

M. De Carvalho, M. Dougherty, A. Fowkes, and M. Wardman. Forecasting travel demand: a comparison of logit and artificial neural network methods. *Journal of the Operational Research Society*, 49(7):717–722, 1998.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

X. Dong and F. S. Koppelman. Comparison of continuous and discrete representations of unobserved heterogeneity in logit models. *Journal of Marketing Analytics*, 2(1): 43–58, 2014.

F. El Zarwi. *Modeling and Forecasting the Impact of Major Technological and Infrastructural Changes on Travel Demand*. PhD thesis, UC Berkeley, 2017.

M. Fosgerau and S. Hess. Competing methods for representing random taste heterogeneity in discrete choice models. 2007.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

D. A. Gopinath. *Modeling heterogeneity in discrete choice processes: Application to travel demand*. PhD thesis, Massachusetts Institute of Technology, 1995.

W. H. Greene and D. A. Hensher. A latent class model for discrete choice analysis: contrasts with mixed logit. *Transportation Research Part B: Methodological*, 37(8): 681–698, 2003. ISSN 0191-2615. doi: 10.1016/S0191-2615(02)00046-2.

W. H. Greene and D. A. Hensher. Revealing additional dimensions of preference heterogeneity in a latent class mixed multinomial logit model. *Applied Economics*, 45(14):1897–1902, 2013. doi: 10.1080/00036846.2011.650325. URL `https://doi.org/10.1080/00036846.2011.650325`.

W. H. Greene, D. A. Hensher, and J. Rose. Accounting for heterogeneity in the variance of unobserved effects in mixed logit models. *Transportation Research Part B: Methodological*, 40(1):75–92, 2006.

S. Gupta and P. K. Chintagunta. On using demographic variables to determine segment membership in logit mixture models. *Journal of Marketing Research*, 31(1): 128–136, 1994. ISSN 00222437. URL `http://www.jstor.org/stable/3151952`.

S. Gupta and L. G. Cooper. The discounting of discounts and promotion thresholds. *Journal of consumer research*, 19(3):401–411, 1992.

T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, pages 297–310, 1986.

T. Hastie and R. Tibshirani. Generalized additive models: some applications. *Journal of the American Statistical Association*, 82(398):371–386, 1987.

D. A. Hensher and W. H. Greene. The mixed logit model: The state of practice. *Transportation*, 30(2):133–176, May 2003. ISSN 1572-9435. doi: 10.1023/A:1022558715350. URL https://doi.org/10.1023/A:1022558715350.

D. A. Hensher and T. T. Ton. A comparison of the predictive potential of artificial neural networks and nested logit models for commuter mode choice. *Transportation Research Part E: Logistics and Transportation Review*, 36(3):155–172, 2000.

S. Hess. Latent class structures: taste heterogeneity and beyond. In *Handbook of choice modelling*, pages 311–329. Edward Elgar Publishing Cheltenham, 2014.

S. Hess and J. M. Rose. Allowing for intra-respondent variations in coefficients estimated on repeated choice data. *Transportation Research Part B: Methodological*, 43(6):708 – 719, 2009. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2009.01.007. URL http://www.sciencedirect.com/science/article/pii/S0191261509000137.

S. Hess and K. E. Train. Recovery of inter- and intra-personal heterogeneity using mixed logit models. *Transportation Research Part B: Methodological*, 45(7):973 – 990, 2011. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2011.05.002. URL http://www.sciencedirect.com/science/article/pii/S019126151100052X.

S. Hess, M. Bierlaire, and J. W. Polak. Estimation of value of travel-time savings using mixed logit models. *Transportation Research Part A: Policy and Practice*, 39(2):221 – 236, 2005. ISSN 0965-8564. doi: https://doi.org/10.1016/j.tra.2004.09.007. URL http://www.sciencedirect.com/science/article/pii/S0965856404001028. Positive Utility of Travel.

S. Hess, M. Ben-Akiva, D. Gopinath, and J. Walker. Taste heterogeneity, correlation and elasticities in latent class choice models. In *Transportation Research Board 88th Annual Meeting*, pages 09–2428. Citeseer, 2009.

G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

H. Hruschka, W. Fettes, M. Probst, and C. Mies. A flexible brand choice model based on neural net methodology a comparison to the linear utility multinomial logit model and its latent class extension. *OR spectrum*, 24(2):127–143, 2002.

H. Hruschka, W. Fettes, and M. Probst. An empirical comparison of the validity of a neural net based multinomial logit choice model to alternative model specifications. *European Journal of Operational Research*, 159(1):166–180, 2004.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

D. Kahnemann and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):363–391, 1979.

G. Kalyanaram and J. D. C. Little. An empirical analysis of latitude of price acceptance in consumer package goods. *Journal of Consumer Research*, 21(3):408–418, 1994. ISSN 00935301, 15375277. URL `http://www.jstor.org/stable/2489682`.

W. A. Kamakura and G. J. Russell. A probabilistic choice model for market segmentation and elasticity structure. *Journal of Marketing Research*, 26(4):379–390, 1989. ISSN 00222437. URL `http://www.jstor.org/stable/3172759`.

M. Keane and N. Wasi. Comparing alternative models of heterogeneity in consumer choice behavior. *Journal of Applied Econometrics*, 28(6):1018–1045, 2013.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

T. Kneib, B. Baumgartner, and W. J. Steiner. Semiparametric multinomial logit models for analysing consumer choice behaviour. *AStA Advances in Statistical Analysis*, 91(3):225–244, Oct 2007. ISSN 1863-818X. doi: 10.1007/s10182-007-0033-2. URL `https://doi.org/10.1007/s10182-007-0033-2`.

L. Krishnamurthi and S. P. Raj. A model of brand choice and purchase quantity price sensitivities. *Marketing Science*, 7(1):1–20, 1988. ISSN 07322399, 1526548X. URL `http://www.jstor.org/stable/183911`.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

D. Lee, S. Derrible, and F. C. Pereira. Comparison of four types of artificial neural network and a multinomial logit model for travel mode choice modeling. *Transportation Research Record*, 2672(49):101–112, 2018. doi: 10.1177/0361198118796971. URL `https://doi.org/10.1177/0361198118796971`.

P. J. Lenk and W. S. DeSarbo. Bayesian inference for finite mixtures of generalized linear models with random effects. *Psychometrika*, 65(1):93–119, 2000.

C. F. Manski. The structure of random utility models. *Theory and Decision*, 8 (3):229–254, Jul 1977. ISSN 1573-7187. doi: 10.1007/BF00133443. URL https://doi.org/10.1007/BF00133443.

D. McFadden. Conditional logit analysis of qualitative choice behavior. 1973.

D. McFadden. Modeling the choice of residential location. *Transportation Research Record*, (673), 1978.

D. McFadden and K. Train. Mixed mnl models for discrete response. *Journal of applied Econometrics*, 15(5):447–470, 2000.

A. Mohammadian and E. J. Miller. Nested logit models and artificial neural networks for predicting household automobile choices: Comparison of performance. *Transportation Research Record*, 1807(1):92–100, 2002. doi: 10.3141/1807-12. URL https://doi.org/10.3141/1807-12.

K. B. Monroe. Buyers' subjective perceptions of price. *Journal of Marketing Research*, 10(1):70–80, 1973. doi: 10.1177/002224377301000110. URL https://doi.org/10.1177/002224377301000110.

D. Nam, H. Kim, J. Cho, and R. Jayakrishnan. A model based on deep learning for predicting travel mode choice. In *Proceedings of the Transportation Research Board 96th Annual Meeting Transportation Research Board, Washington, DC, USA*, pages 8–12, 2017.

H. Omrani. Predicting travel mode of individuals by machine learning. *Transportation Research Procedia*, 10:840–849, 2015.

J. H. Pedrick and F. S. Zufryden. Evaluating the impact of advertising media plans: A model of consumer purchase dynamics using single-source data. *Marketing Science*, 10(2):111–130, 1991.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

B. Provencher and R. C. Bishop. Does accounting for preference heterogeneity improve the forecasting of a random utility model? a case study. *Journal of Environmental Economics and Management*, 48(1):793 – 810, 2004. ISSN 0095-0696. doi: https://doi.org/10.1016/j.jeem.2003.11.001. URL http://www.sciencedirect.com/science/article/pii/S0095069603001359.

B. Provencher, K. A. Baerenklau, and R. C. Bishop. A finite mixture logit model of recreational angling with serially correlated random utility. *American Journal of Agricultural Economics*, 84(4):1066–1075, 2002. ISSN 00029092, 14678276. URL http://www.jstor.org/stable/1244746.

D. Revelt. *Three Discrete Choice Random Coefficients Papers and One Police-crime Study*. University of California, Berkeley, 1999. URL https://books.google.com/books?id=kU3EHgAACAAJ.

D. Revelt and K. E. Train. Mixed logit with repeated choices: households' choices of appliance efficiency level. *Review of economics and statistics*, 80(4):647–657, 1998.

D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

M. Sarrias and R. Daziano. Multinomial logit models with continuous and discrete individual heterogeneity in r: The gmnl package. *Journal of Statistical Software, Articles*, 79(2):1–46, 2017. ISSN 1548-7660. doi: 10.18637/jss.v079.i02. URL https://www.jstatsoft.org/v079/i02.

M. Schindler, B. Baumgartner, and H. Hruschka. Nonlinear effects in brand choice models: comparing heterogeneous latent class to homogeneous nonlinear models. *Schmalenbach Business Review*, 59(2):118–137, 2007.

J. Shen. Latent class model or mixed logit model? a comparison by transport mode choice data. *Applied Economics*, 41(22):2915–2924, 2009. doi: 10.1080/00036840801964633. URL https://doi.org/10.1080/00036840801964633.

D. Shmueli, I. Salomon, and D. Shefer. Neural network analysis of travel behavior: evaluating tools for prediction. *Transportation Research Part C: Emerging Technologies*, 4(3):151–166, 1996.

B. Sifringer, V. Lurkin, and A. Alahi. Let me not lie: Learning multinomial logit. *arXiv preprint arXiv:1812.09747*, 2018.

J. Siikamaki. *Discrete choice experiments for valuing biodiversity conservation in Finland*. PhD thesis, University of California, Davis, 2001.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

J. Swait. A structural equation model of latent segmentation and product choice for cross-sectional revealed preference choice data. *Journal of Retailing and Consumer Services*, 1(2):77 – 89, 1994. ISSN 0969-6989. doi: https://doi.org/10.1016/0969-6989(94)90002-7. URL http://www.sciencedirect.com/science/article/pii/0969698994900027.

C. Torres, N. Hanley, and A. Riera. How wrong can you be? implications of incorrect utility function specification for welfare measurement in choice experiments. *Journal of Environmental Economics and Management*, 62(1):111–121, 2011.

K. E. Train. A comparison of hierarchical bayes and maximum simulated likelihood for mixed logit. *University of California, Berkeley*, pages 1–13, 2001.

K. E. Train. Em algorithms for nonparametric estimation of mixing distributions. *Journal of Choice Modelling*, 1(1):40–69, 2008.

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

K. E. Train. Mixed logit with a flexible mixing distribution. *Journal of choice modelling*, 19:40–53, 2016.

K. E. Train, D. L. McFadden, and M. Ben-Akiva. The demand for local telephone service: A fully discrete model of residential calling patterns and service choices. *The RAND Journal of Economics*, pages 109–123, 1987.

S. van Cranenburgh and A. Alwosheel. An artificial neural network based approach to investigate travellersâĂŹ decision rules. *Transportation Research Part C: Emerging Technologies*, 98:152–166, 2019.

M. van der Pol, G. Currie, S. Kromm, and M. Ryan. Specification of the utility function in discrete choice experiments. *Value in Health*, 17(2):297 – 301, 2014. ISSN 1098-3015. doi: https://doi.org/10.1016/j.jval.2013.11.009. URL `http://www.sciencedirect.com/science/article/pii/S109830151304391X`.

A. Vij and R. Krueger. Random taste heterogeneity in discrete choice models: Flexible nonparametric finite mixture distributions. *Transportation Research Part B: Methodological*, 106:76–101, 2017.

J. L. Walker. *Extended discrete choice models: integrated framework, flexible error structures, and latent variables*. PhD thesis, Massachusetts Institute of Technology, 2001.

J. L. Walker and J. Li. Latent lifestyle preferences and household location decisions. *Journal of Geographical Systems*, 9(1):77–101, 2007.

J. L. Walker, M. Ben-Akiva, and D. Bolduc. Identification of parameters in normal error component logit-mixture (neclm) models. *Journal of Applied Econometrics*, 22(6):1095–1125, 2007.

S. Wang and J. Zhao. Using deep neural network to analyze travel mode choice with interpretable economic information: An empirical example. *arXiv preprint arXiv:1812.04528*, 2018.

S. Wang and J. Zhao. Multitask learning deep neural network to combine revealed and stated preference data. *arXiv preprint arXiv:1901.00227*, 2019.

M. Wedel and P. S. Leeflang. A model for the effects of psychological pricing in gabor-granger price studies. *Journal of Economic Psychology*, 19(2):237 – 260, 1998. ISSN 0167-4870. doi: https://doi.org/10.1016/S0167-4870(98)00006-3. URL `http://www.sciencedirect.com/science/article/pii/S0167487098000063`.

M. Wedel, W. Kamakura, N. Arora, A. Bemmaor, J. Chiang, T. Elrod, R. Johnson, P. Lenk, S. Neslin, and C. S. Poulsen. Discrete and continuous representations of unobserved heterogeneity in choice modeling. *Marketing Letters*, 10(3):219–232, 1999.

P. M. West, P. L. Brockett, and L. L. Golden. A comparative analysis of neural networks and statistical methods for predicting consumer choice. *Marketing Science*, 16(4):370–391, 1997.

H. C. W. L. Williams. On the formation of travel demand models and economic evaluation measures of user benefit. *Environment and Planning A: Economy and Space*, 9(3):285–344, 1977. doi: 10.1068/a090285. URL `https://doi.org/10.1068/a090285`.

R. S. Winer. A reference price model of brand choice for frequently purchased products. *Journal of consumer research*, 13(2):250–256, 1986.

R. S. Winer. Behavioral perspective on pricing. In *Issues in Pricing*, pages 35–57. Lexington Books, 1988.

M. Wong, B. Farooq, and G.-A. Bilodeau. Discriminative conditional restricted boltzmann machine for discrete choice and latent variable modelling. *Journal of choice modelling*, 29:152–168, 2018.

X. Zhao, X. Yan, A. Yu, and P. Van Hentenryck. Modeling stated preference for mobility-on-demand transit: A comparison of machine learning and logit models. *arXiv preprint arXiv:1811.01315*, 2018.