THE UNIVERSITY OF CHICAGO

STATISTICAL MACHINE LEARNING METHODS FOR COMPLEX, HETEROGENEOUS DATA

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF STATISTICS

BY
MAHTIYAR BONAKDARPOUR

CHICAGO, ILLINOIS
JUNE 2019

To my loving parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This work would not have been possible without the support and guidance of my advisor, John Lafferty. John's eclectic research taste and expertise gave me exposure to a wide range of interesting topics, and his never-ending creativity planted the seed for each project found in the following chapters. I would also like to thank my committee members Rina Foygel Barber and Matthew Stephens. Rina was a pillar of support for me during my early years, and I am thankful to have been able to collaborate with her on the first chapter in this thesis. And Matthew's excellent course on Bayesian statistics continues to pay dividends in my research, shaping the way I think about probabilistic modeling and uncertainty.

I have also had the great fortune of meeting and interacting with an amazing number of graduate students at both the University of Chicago and Yale University. In the early days, I forged strong bonds with Chris McKennan, Jon Eskreis-Winkler, and Peter Hansen. As Jon once put it, "it was like getting married during war time." As the years went on, I enjoyed countless stimulating conversations with Chris McKennan, Vivak Patel, and Wooseok Ha. They all played a role in helping me shape my approach to research and I will forever be grateful for their friendship. At Yale, Natalie Voss and Dana Yang generously welcomed me into the department and played a crucial role in making it feel like home. And I would be remiss if I didn't mention my friends from Ecology & Evolution, Alex White and Joel Smith, who always reminded me to go outside and have some fun.

I also want to thank some wonderful collaborators including Min Xu, George Linderman, Sabyasachi Chatterjee, Qinqing Zheng, Annie Marsden, Panos Toulis, and members of The Hopper Project: Michi Yasunaga, Sam Helms, and Jay Dhanoa.

And last but certainly not least, I would like to thank my family for their never-ending support, and Molly for her encouragement, patience, and love.

ABSTRACT

This thesis develops statistical machine learning methodology for three distinct tasks. Each method blends classical statistical approaches with machine learning methods to provide principled solutions to problems with complex, heterogeneous datasets. The first framework proposes two methods for high-dimensional shape-constrained regression and classification. These methods reshape pre-trained prediction rules to satisfy shape constraints like monotonicity and convexity. The second method provides a nonparametric approach to the econometric analysis of discrete choice. This method provides a scalable algorithm for estimating utility functions with random forests, and combines this with random effects to properly model preference heterogeneity. The final method draws inspiration from early work in statistical machine translation to construct embeddings for variable-length objects like mathematical equations.

INTRODUCTION

Statistical machine learning blends statistical approaches with modern machine learning tools and algorithms to provide principled methods for data analysis. These tools provide scalable and accurate procedures rooted in statistical principles. As datasets become larger, higher dimensional, and more complex, new tools are required to make better predictions, satisfy qualitative constraints, and account for heterogeneity. This thesis introduces several statistical machine learning methods for complex, heterogeneous data.

Shape constraints arise naturally in many real-world situations. For example, holding all other variables constant, we might expect that housing prices are a decreasing function of neighborhood crime rate, and that utility is a concave, increasing function of income. Shape constrained estimators allow practitioners to impose qualitative assumptions on the regression function without appealing to parametric models.

Shape-constrained estimation has an extensive history of research focused on topics such as monotone, unimodal, and convex regression and log-concave density estimation. However, classical methods for shape-constrained regression suffer from the curse of dimensionality. On the other hand, machine learning approaches like random forests and gradient boosting have been shown to obtain outstanding empirical performance on high-dimensional prediction tasks.

Chapter 1 proposes two methods for shape-constrained prediction that blend the classical least-squares approach to shape-constraints with machine learning methods for high-dimensional regression and classification. The first method can be applied to any black box prediction rule, while the second is focused on random forests. Algorithms are developed for computing the estimators efficiently. This work was done jointly with Rina Foygel Barber, Sabyasachi Chatterjee, and John Lafferty, and was published in ICML 2018 Bonakdarpour et al. (2018).

Chapter 2 describes Random Choice Forests (RCFs), a method for estimating discrete choice models with random forests. Discrete choice models are appropriate for situations

in which agents are each making choices from a finite set of alternatives, and attributes are observed for both agents and items. Once a discrete choice model is estimated, it can be used to make predictions about future choices. They can also be used to answer counterfactual questions about how aggregate demand would change as a function of various attributes (e.g price). We discuss discrete choice models in more detail in Chapter 2. An efficient tree-growing algorithm is developed, along with a two-stage procedure for estimating random forests with random effects. This work was done jointly with Min Xu and John Lafferty.

Finally, in Chapter 3, we propose a general framework for constructing embeddings for arbitrary objects. Word embeddings in particular have become a popular tool in natural language processing. A word embedding is a mapping from a discrete vocabulary of words to a vector space. Embeddings are constructed so that semantically similar words are mapped to nearby vectors. Approaches for constructing word embeddings can be viewed as feature engineering tools, and they have been shown to be useful for many downstream NLP tasks. Drawing inspiration from early work in statistical machine translation, we propose a method for constructing embeddings for variable-length objects such as words, sentences, and mathematical equations.

CHAPTER 1

PREDICTION RULE RESHAPING

1.1 Abstract

Two methods are proposed for high-dimensional shape-constrained regression and classification. These methods *reshape* pre-trained prediction rules to satisfy shape constraints like monotonicity and convexity. The first method can be applied to any pre-trained prediction rule, while the second method deals specifically with random forests. In both cases, efficient algorithms are developed for computing the estimators, and experiments are performed to demonstrate their performance on four datasets. We find that reshaping methods enforce shape constraints without compromising predictive accuracy.

1.2 Introduction

Shape constraints like monotonicity and convexity arise naturally in many real-world regression and classification tasks. For example, holding all other variables fixed, a practitioner might assume that the price of a house is a decreasing function of neighborhood crime rate, that an individual's utility function is concave in income level, or that phenotypes such as height or the likelihood of contracting a disease are monotonic in certain genetic effects.

Parametric models like linear regression implicity impose monotonicity constraints at the cost of strong assumptions on the true underlying function. On the other hand, nonparametric techniques like kernel regression impose weak assumptions, but do not guarantee monotonicity or convexity in their predictions. Shape-constrained nonparametric regression methods attempt to offer the best of both worlds, allowing practitioners to dispense with parametric assumptions while retaining many of their appealing properties.

However, classical approaches to nonparametric regression under shape constraints suffer from the curse of dimensionality (Han and Wellner, 2016; Han et al., 2017). Some methods

have been developed to mitigate this issue under assumptions like additivity, where the true function f is assumed to have the form $f(x) = \sum_j f_j(x_j) + c$, where a subset of the component f_j's are shape-constrained (Pya and Wood, 2015; Chen and Samworth, 2016; Xu et al., 2016). But in many real-world settings, the lack of interaction terms among the predictors can be too restrictive.

Approaches from the machine learning community like random forests, gradient boosted trees, and deep learning methods have been shown to exhibit outstanding empirical performance on high-dimensional tasks. But these methods do not guarantee monotonicity or convexity.

In this paper, we propose two methods for high-dimensional shape-constrained regression and classification. These methods blend the performance of machine learning methods with the classical least-squares approach to nonparametric shape-constrained regression.

In Section (1.3.1), we describe black box reshaping, which takes any pre-trained prediction rule and reshapes it on a set of test inputs to enforce shape constraints. In the case of monotonicity constraints, we develop an efficient algorithm to compute the estimator. Section (1.3.2) presents a second method designed specifically to reshape random forests (Breiman, 2001). This approach reshapes each individual decision tree based on its split rules and estimated leaf values. Again, in the case of monotonicity constraints, we present another efficient reshaping algorithm. We apply our methods to four datasets in Section (1.4) and show that they enforce the pre-specified shape constraints without sacrificing accuracy.

1.2.1 Related Work

In the context of monotonicity constraints, the black box reshaping method is related to the method of rearrangements (Chernozhukov et al., 2009, 2010). The rearrangement operation takes a pre-trained prediction rule and sorts its predictions to enforce monotonicity. In higher dimensions, the rearranged estimator is the average of one-dimensional rearrangements. In contrast, our methods focus on isotonization of prediction values, jointly reshaping multiple

dimensions in tandem. It would be interesting to explore adaptive procedures that average rearranged and isotonized predictions in future work.

Monotonic decision trees have previously been studied in the context of classification. Several methods require that the training data satisfy monotonicity constraints (Potharst and Feelders, 2002; Makino et al., 1996), a relatively strong assumption in the presence of noise. The methods we propose here do not place any restrictions on the training data.

Another class of methods augment the score function for each split to incorporate the degree of non-monotonicity introduced by that split (Ben-David, 1995; González et al., 2015). However, this approach does not guarantee monotonicity. Feelders and Pardoel (2003) apply pruning algorithms to non-monotonic trees as a post-processing step in order to enforce monotonicity. For a comprehensive survey of estimating monotonic functions, see Gupta et al. (2016) .

A line of recent work has led to a method for learning deep monotonic models by alternating different types of monotone layers (You et al., 2017). Amos et al. (2017) propose a method for fitting neural networks whose predictions are convex with respect to a subset of predictors.

Our methods differ from this work in several ways. First, our techniques can be used to enforce both monotonic and convex/concave relationships. Unlike pruning methods, neither approach presented here changes the structure of the original tree. Black box reshaping, described in Section (1.3.1), can be applied to any pre-trained prediction rule, giving practitioners the flexibility of picking the method of their choice. And both methods guarantee that the intended shape constraints are satisfied on test data.

1.3 Prediction Rule Reshaping

In what follows, we say that a function $f : \mathbb{R}^d \to \mathbb{R}$ is monotone with respect to variables $\mathcal{R} \subseteq [d] = \{1, \ldots, d\}$ if $f(x) \leq f(y)$ when $x_i \leq y_i$ for $i \in \mathcal{R}$, and $x_i = y_i$ otherwise.

Similarly, a function f is convex in $\mathcal{R}$ if for all $x, y \in \mathbb{R}^d$ and $\alpha \in [0, 1]$, $f(\alpha x + (1-\alpha)y) \leq$

$\alpha f(x) + (1-\alpha)f(y)$ when $x_i = y_i \; \forall i \notin \mathcal{R}$.

1.3.1 Black Box Reshaping

Let $\widehat{f} \; : \; \mathbb{R}^d \to \mathbb{R}$ denote an arbitrary prediction rule fit on a training set and assume we have a candidate set of shape constraints with respect to variables $\mathcal{R} \subseteq [d]$. For example, we might require that the function be monotone increasing in each variable $v \in \mathcal{R}$.

Let $\mathcal{F}$ denote the class of functions that satisfy the desired shape constraints on each predictor variable $v \in \mathcal{R}$. We aim to find a function $f^* \in \mathcal{F}$ that is close to $\widehat{f}$ in the L_2 norm:

$$f^* = \underset{f \in \mathcal{F}}{\arg\min} \|f - \widehat{f}\|_2 \tag{1.3.1}$$

where the L_2 norm is with respect to the uniform measure on a compact set containing the data. We simplify this infinite-dimensional problem by only considering values of $\widehat{f}$ on certain fixed test points.

Suppose we take a sequence $t^1, t^2, \ldots, t^n$ of test points, each in $\mathbb{R}^d$, that differ only in their v-th coordinate so that $t^i_k = t^{i'}_k$ for all $k \neq v$. These points can be ordered by their v-th coordinate, allowing us to consider shape constraints on the vector $(f(t^1), f(t^2), ..., f(t^n)) \in \mathbb{R}^n$. For instance, under a monotone-increasing constraint with respect to v, if $t^1_v \leq t^2_v \leq \cdots \leq t^n_v$, then we consider functions f such that $(f(t^1), f(t^2), ..., f(t^n))$ is a monotone sequence.

There is now the question of choosing a test point t as well as a sequence of values $t^1_v, ..., t^n_v$ to plug into its v-th coordinate. A natural choice is to use the observed data values as both test points and coordinate values.

Denote $\mathcal{D}_n = \{(x^1, y^1), \ldots, (x^n, y^n)\}$ as a set of observed values where y^i is the response and $x^i \in \mathbb{R}^d$ are the predictors. From each x^i, we construct a sequence of test points that can be ordered according to their v-th coordinate in the following way. Let $x^{i,k,v}$ denote the observed vector x^i with its v-th coordinate replaced by the v-th coordinate of x^k, so that

$$x^{i,k,v} = (x^i_1, x^i_2, \ldots, x^i_{v-1}, x^k_v, x^i_{v+1}, \ldots, x^i_d). \tag{1.3.2}$$

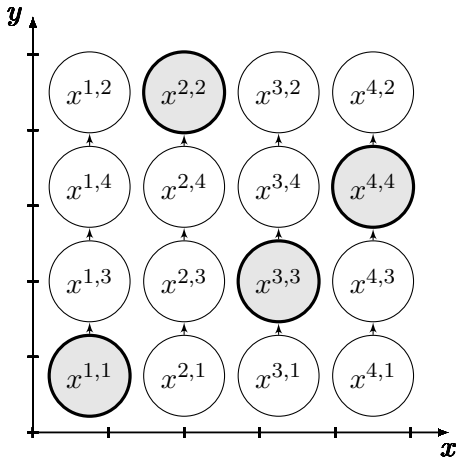


Figure 1.1: Two-dimensional illustration of the black box setup when reshaping the y dimension. $x^{i,k}$ denotes the original test point x^i with its y-coordinate replaced by the y-coordinate of x^k. Dark gray nodes represent the original observed points. For monotonicity constraints, a directed edge from node x to node y represents the constraint $f(x) \leq f(y)$ on the shape-constrained function f.

This process yields n points from x^i that can be ordered by their v-th coordinate, $x^{i,1,v}, x^{i,2,v}, \ldots, x^{i,n,v}$. We then require $(f(x^{i,1,v}), f(x^{i,2,v}), \ldots, f(x^{i,n,v})) \in S_v$ where $S_v \subset \mathbb{R}^d$ is the appropriate convex cone that enforces the shape constraint for variable $v \in \mathcal{R}$, for example the cone of monotone increasing or convex sequences.

To summarize, for each coordinate $v \in \mathcal{R}$ and for each $i \in [n]$, we:

1. Take the i-th observed data point x^i as a test point.

2. Replace its v-th coordinate with the n observed v-th coordinates $x_v^1, ...x_v^n$ to produce $x^{i,1,v}, x^{i,2,v}, \ldots, x^{i,n,v}$.

3. Enforce the appropriate shape constraint on the vector of evaluated function values, $(f(x^{i,1,v}), f(x^{i,2,v}), \ldots, f(x^{i,n,v})) \in S_v$.

See Figure (1.1) for an illustration. This leads to the following relaxation of (1.3.1):

$$f^* = \underset{f \in \mathcal{F}_n}{\arg\min} \|f - \widehat{f}\|_2 \tag{1.3.3}$$

where $\mathcal{F}_n$ is the class of functions f such that $(f(x^{i,1,v}), f(x^{i,2,v}), \ldots, f(x^{i,n,v})) \in S_v \subset \mathbb{R}^n$

for each $v \in \mathcal{R}$ and each $i \in [n]$. In other words, we have relaxed the shape constraints on the function f, requiring the constraints to hold relative to the selected test points. However, this optimization is still infinite dimensional.

We make the final transition to finite dimensions by changing the objective function to only consider values of f on the test points. Letting $F_{i,k,v}$ denote the value of f evaluated on test point $x^{i,k,v}$, we relax (1.3.3) to obtain the solution $F^* = (F^*_{i,k,v})_{v\in\mathcal{R},i\in[n],k\in[n]}$ of the optimization:

$$\begin{aligned} \underset{F}{\arg\min} \quad & \sum_{i,k,v} (F_{i,k,v} - \widehat{f}(x^{i,k,v}))^2 \\ \text{subject to} \quad & (F_{i,1,v}, ..., F_{i,n,v}) \in (S_v)_{v\in\mathcal{R}},\ \forall\, i \in [n] \end{aligned} \tag{1.3.4}$$

However, this leads to ill-defined predictions on the original data points $x^1, ..., x^n$, since for each v, $x^{i,i,v} = x^i$, but we may obtain different values $F^*_{i,i,v}$ for various $v \in \mathcal{R}$.

We avoid this issue by adding a consistency constraint (1.3.7) to obtain our final black box reshaping optimization (`BBOPT`):

$$\underset{F}{\arg\min} \quad \sum_{i,k,v} (F_{i,k,v} - \widehat{f}(x^{i,k,v}))^2 \tag{1.3.5}$$

$$\text{subject to} \quad (F_{i,1,v}, ..., F_{i,n,v}) \in (S_v)_{v\in\mathcal{R}},\ \forall\, i \in [n] \tag{1.3.6}$$

$$\text{and} \quad F_{i,i,v} = F_{i,i,w} \quad \forall\, v, w \in \mathcal{R}, \forall\, i \in [n] \tag{1.3.7}$$

We then take the reshaped predictions to be

$$f^*(x^i) = F^*_{i,i,v}$$

for any $v \in \mathcal{R}$. Since the constraints depend on each x^i independently, `BBOPT` decomposes into n optimization problems, one for each observed value. Note that the true response values y^i are not used when reshaping. We could select optimal shape constraints on a held-out test set.

Intersecting Isotonic Regression

In this section, we present an efficient algorithm for solving `BBOPT` for the case when each $\mathcal{S}_v$ imposes monotonicity constraints. Let $R = |\mathcal{R}|$ denote the number of monotonicity constraints.

When reshaping with respect to only one predictor ($R = 1$), the consistency constraints (1.3.7) vanish, so the optimization decomposes into n isotonic regression problems. Each problem is efficiently solved in $\Theta(n)$ time with the pool adjacent violators algorithm (PAVA) (Ayer et al., 1955).

For $R > 1$ monotonicity constraints, `BBOPT` gives rise to n independent *intersecting isotonic regression* problems. The k-th problem corresponds to the k-th observed value x^k; the "intersection" is implied by the consistency constraints (1.3.7). For each independent problem, our algorithm takes $O(m \log R)$ time, where $m = n \times R$ is the number of variables in each problem.

We first state the general problem. Assume $v^1, v^2, \ldots, v^K$ are each real-valued vectors with dimensions $d_1, d_2, \ldots, d_K$, respectively. Let $i_j \in \{1, \ldots, d_j\}$ denote an index in the j-th vector v^j. The intersecting isotonic regression problem (`IISO`) is:

$$\begin{aligned} \underset{(\widehat{v}^k)_{k=1}^K}{\text{minimize}} \quad & \sum_{k=1}^{K} \|\widehat{v}^k - v^k\|^2 \\ \text{subject to} \quad & \widehat{v}_1^k \le \widehat{v}_2^k \le \cdots \le \widehat{v}_{d_k}^k, \ \forall\, k \in [K] \\ \text{and} \quad & \widehat{v}_{i_1}^1 = \widehat{v}_{i_2}^2 = \cdots = \widehat{v}_{i_K}^K \end{aligned} \tag{1.3.8}$$

First consider the simpler constrained isotonic regression problem with a single sequence

Algorithm 1 `IISO Algorithm`

1. Apply PAVA to each of the 2K tails.
2. Combine and sort the left and right tails separately.
3. Find segment s^* in between tail values where the derivative $g'(\eta)$ changes sign.
4. Compute c^*, the minimizer of $g(c)$ in segment s^*.

$v \in \mathbb{R}^d$, index $i \in [d]$, and fixed value $c \in \mathbb{R}$

$$\begin{aligned} \underset{\widehat{v}}{\text{minimize}} \quad & \|\widehat{v} - v\|^2 \\ \text{subject to} \quad & \widehat{v}_1 \leq \widehat{v}_2 \leq \cdots \leq \widehat{v}_d \\ \text{and} \quad & \widehat{v}_i = c \end{aligned} \tag{1.3.9}$$

Lemma 1.3.1. *The solution v^* to (1.3.9) can be computed by using index i as a pivot and splitting v into its left and right tails, so that $\ell = (v_1, v_2, \ldots, v_{i-1})$ and $r = (v_{i+1}, \ldots, v_d)$, then applying PAVA to obtain monotone tails $\widehat{\ell}$ and $\widehat{r}$. v^* is obtained by setting elements of $\widehat{\ell}$ and $\widehat{r}$ to*

$$\begin{aligned} \ell_k^* &= \min(\widehat{\ell}_k, c) \\ r_k^* &= \max(\widehat{r}_k, c) \end{aligned} \tag{1.3.10}$$

and concatenating the resulting tails so that $v^* = (\ell^*, c, r^*) \in \mathbb{R}^d$.

We now explain the `IISO Algorithm` presented in Algorithm (1). First divide each vector v^j into two tails, the left tail ℓ^j and the right tail r^j, using the intersection index i_j as a pivot,

$$v^j = (\underbrace{v_1^j, v_2^j, \ldots, v_{(i_j-1)}^j}_{\ell^j}, v_{i_j}^j, \underbrace{v_{(i_j+1)}^j, v_{(i_j+2)}^j, \ldots, v_{d_j}^j}_{r^j}).$$

resulting in $2K$ tails $\{\ell^1, \ldots, \ell^K, r^1, \ldots, r^K\}$.

Step 1 of Algorithm (1) performs an unconstrained isotonic regression on each tail using PAVA to obtain $2K$ monotone tails $\{\widehat{\ell}^1, \ldots, \widehat{\ell}^K, \widehat{r}^1, \ldots, \widehat{r}^K\}$. This can be done in $\Theta(n)$ time, where n is the total number of elements across all vectors so that $n = \sum_{i=1}^K d_i$.

Given the monotone tails, we can write a closed-form expression for the `IISO` objective function in terms of the value at the point of intersection.

Let c be the value of the vectors at the point of intersection so that $c = \widehat{v}^1_{i_1} = \widehat{v}^2_{i_2} = \cdots = \widehat{v}^K_{i_K}$. For a fixed c, we can solve `IISO` by applying Lemma (1.3.1) to each sequence separately. This yields the following expression for the squared error as a function of c:

$$g(c) = \sum_{k=1}^{K}(c - v^k_{i_k})^2 + \sum_{k=1}^{K}\sum_{i=1}^{i_k-1}(\ell^k_i - \min(\widehat{\ell}^k_i, c))^2 + \sum_{l=1}^{K}\sum_{j=i_l+1}^{d_l}(r^l_j - \max(\widehat{r}^l_j, c))^2 \quad (1.3.11)$$

which is piecewise quadratic with knots at each $\widehat{\ell}^k_i$ and $\widehat{r}^l_j$. Our goal is to find $c^\star = \min_c g(c)$. Note that $g(c)$ is convex and differentiable.

We proceed by computing the derivative of g at each knot, from smallest to largest, and finding the segment in which the sign of the derivative changes from negative to positive. The minimizer c^* will live in this segment.

Step 2 of Algorithm (1) merges the left and right sorted tails into two sorted lists. This can be done in $O(n \log K)$ time with a heap data structure. Step 3 computes the derivative of the objective function g at each knot, from smallest to largest, searching for the segment in which the derivative changes sign. Step 4 computes the minimizer of g in the corresponding segment. By updating the derivative incrementally and storing relevant side information, Steps 3 and 4 can be done in linear time.

The total time complexity is therefore $O(n \log(K))$.

1.3.2 Reshaping Random Forests

In this section, we describe a framework for reshaping a random forest to ensure monotonicity of its predictions in a subset of its predictor variables. A similar method can be applied to ensure convexity. For both regression and probability trees (Malley et al., 2012), the prediction of the forest is an average of the prediction of each tree; it is therefore sufficient to ensure monotonicity or convexity of the trees. For the rest of this section, we focus on

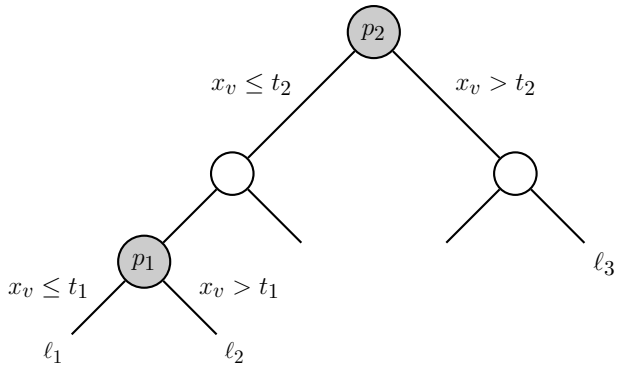


Figure 1.2: An illustration motivating the random forest reshaping method. The only nodes that split on the shape-constrained variable v are p_1 and p_2. Assume $x \in \mathbb{R}^d$ drops down to leaf ℓ_1 and that, holding all other variables constant, increasing x_v beyond t_1 and t_2, results in dropping to leaves ℓ_2 and ℓ_3, respectively. To enforce monotonicity in v for this point, we need to ensure leaf values μ_ℓ follow $\mu_{\ell_1} \leq \mu_{\ell_2} \leq \mu_{\ell_3}$.

reshaping individual trees to enforce monotonicity.

Our method is a two-step procedure. The first step is to grow a tree in the usual way. The second step is to *reshape* the leaf values to enforce monotonicity. We hope to explore the implications of combining these steps in future work.

Let $T(x)$ be a regression tree and $\mathcal{R} \subseteq [d]$ a set of predictor variables to be reshaped. Let $x \in \mathbb{R}^d$ be an input point and denote the k-th coordinate of x as x_k. Assume $v \in \mathcal{R}$ is a predictor variable to be reshaped. The following thought experiment, illustrated in Figure (1.2), will motivate our approach.

Imagine dropping x down T until it falls in its corresponding leaf, ℓ_1. Let p_1 be the closest ancestor node to ℓ_1 that splits on v and assume it has split rule $\{x_v \leq t_1\}$. Holding all other coordinates constant, increasing x_v until it is greater than t_1 would create a new point that drops down to a different leaf ℓ_2 in the right subtree of p_1.

If ℓ_1 and ℓ_2 both share another ancestor p_2 farther up the tree with split rule $\{x_v \leq t_2\}$, increasing x_v beyond t_2 would yield another leaf ℓ_3. Assume these leaves have no other shared ancestors that split on v. Denoting the value of leaf ℓ as μ_ℓ, in order to ensure monotonicity in v for this point x, we require $\mu_{\ell_1} \leq \mu_{\ell_2} \leq \mu_{\ell_3}$.

We use this line of thinking to propose a framework for estimating monotonic random

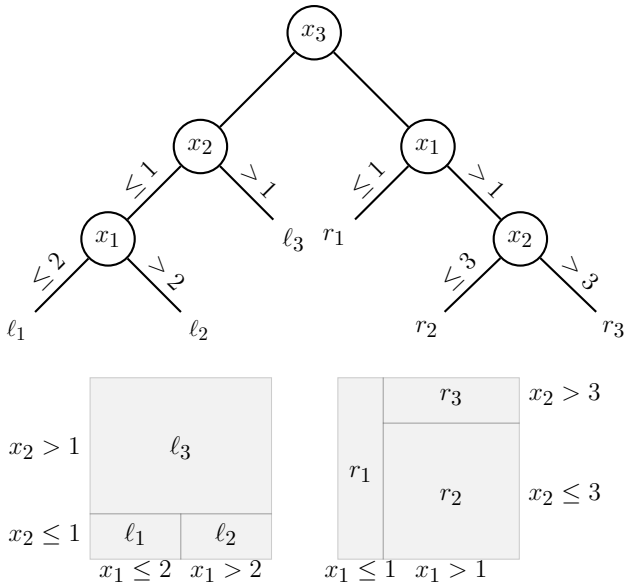


Figure 1.3: Suppose we have three variables (x_1, x_2, x_3) and when we split on reshaped variable x_3, the left and right subtrees and their corresponding cells are as shown above. By examination, any point that drops to ℓ_2 can only travel to r_2 when its x_3 coordinate is increased. By this logic, the exact estimator would use the six constraints , $\mu_{\ell_2} \leq \mu_{r_2}, \mu_{\ell_1} \leq \mu_{r_1}, \mu_{\ell_1} \leq \mu_{r_2}, \mu_{\ell_3} \leq \mu_{r_1}, \mu_{\ell_3} \leq \mu_{r_2}, \mu_{\ell_3} \leq \mu_{r_3}$, whereas the over-constrained estimator would use all nine pairwise constraints.

forests and describe two estimators that fall under this framework.

Exact Estimator

Each leaf ℓ in a decision tree is a cell (or hyperrectangle) C_ℓ which is an intersection of intervals

$$C_\ell = \bigcap_{j=1}^{d} \{x : x_j \in I_j^\ell\}$$

When we split on a shape-constrained variable v with split-value t, each cell in the left subtree is of the form $C_l = \bar{C}_l \cap \{x : x_v \leq t\}$ and each cell in the right subtree is of the form $C_r = \bar{C}_r \cap \{x : x_v > t\}$.

For cells l in the left subtree and r in the right subtree, our goal is to constrain the corresponding leaf values $\mu_l \leq \mu_r$ only when $\bar{C}_l \cap \bar{C}_r \neq \emptyset$. See Figure (1.3) for an illustration. We must devise an algorithm to find the intersecting cells (l, r), and add each to a constraint set E. This can be done efficiently with an interval tree data structure.

Assume there are n unique leaves appearing in E. The exact estimator is obtained by solving the following optimization:

$$\begin{aligned} \min_{(\widehat{\mu}_\ell)_{\ell=1}^n} \quad & \sum_{\ell=1}^{n} (\mu_\ell - \widehat{\mu}_\ell)^2 \\ \text{subject to} \quad & \widehat{\mu}_i \leq \widehat{\mu}_j \; \forall \, (i,j) \in E \end{aligned} \tag{1.3.12}$$

where μ_ℓ is the original value of leaf ℓ.

This is an instance of L_2 isotonic regression on a directed acyclic graph where each leaf value μ_ℓ is a node, and each constraint in E is an edge. With n vertices and m edges, the fastest known exact algorithm for this problem has time complexity $\Theta(n^4)$ (Spouge et al., 2003), and the fastest known δ-approximate algorithm has complexity $O(m^{1.5} \log^2 n \log \frac{n}{\delta})$ (Kyng et al., 2015).

With a corresponding change to the constraints in Equation (1.3.12), this approach extends naturally to convex regression trees. It can also be applied directly to probability trees for binary classification by reshaping the estimated probabilities in each leaf.

Over-constrained Estimator

In this section, we propose an alternative estimator that can be more efficient to compute, depending on the tree structure. In our experiments below, we find that computing this estimator is always faster.

Let E_p denote the set of constraints that arise between leaf values under a shape-constrained split node p. By adding additional constraints to E_p, we can solve (1.3.12) exactly for each shape-constrained split node in $O(n_p \log n_p)$ time, where n_p is the number of leaves under p.

In this setting, each shape-constrained split node gives rise to an independent optimization involving its leaves. Due to transitivity, we can solve these optimizations sequentially in reverse (bottom-up) level-order on the tree.

Let n_p denote the number of leaves under node p. For each node p that is split on a shape-constrained variable, the over-constrained estimator solves the following max-min problem:

$$\begin{aligned} \min_{(\widehat{\mu}_\ell)_{\ell=1}^{n_p}} \quad & \sum_{\ell=1}^{n_p} (\mu_\ell - \widehat{\mu}_\ell)^2 \\ \text{subject to} \quad & \max_{\ell \in \text{left}(p)} \widehat{\mu}_\ell \leq \min_{r \in \text{right}(p)} \widehat{\mu}_r \end{aligned} \tag{1.3.13}$$

where left(p) denotes all leaves in the left subtree of p and right(p) denotes all leaves in the right subtree.

This is equivalent to adding an edge (ℓ, r) to E for every pair of leaves such that ℓ is in left(p) and r is in right(p). All such pairs do not necessarily exist in E for the exact estimator; see Figure (1.3). For each shape-constrained split, (1.3.13) is an instance of L_2 isotonic regression on a complete directed bipartite graph.

For a given shape-constrained split node p, let $\ell = (\ell_1, \ell_2, \ldots, \ell_{n_1})$ be the values of the leaves in its left subtree, and $r = (r_1, r_2, \ldots, r_{n_2})$ be the values of the leaves in its right subtree, indexed so that $\ell_1 \leq \cdots \leq \ell_{n_1}$ and $r_1 \leq \cdots \leq r_{n_2}$. Then the max-min problem (1.3.13) is equivalent to:

$$\begin{aligned} \min_{\widetilde{\ell}, \widetilde{r}} \quad & \sum_{i=1}^{n_1} (\ell_i - \widetilde{\ell}_i)^2 + \sum_{i=1}^{n_2} (r_i - \widetilde{r}_i)^2 \\ \text{subject to} \quad & \widetilde{\ell}_1 \leq \widetilde{\ell}_2 \leq \ldots \leq \widetilde{\ell}_{n_1} \leq \widetilde{r}_1 \leq \cdots \leq \widetilde{r}_{n_2} \end{aligned} \tag{1.3.14}$$

The solution to this optimization is of the form $\widetilde{\ell}_i = \min(c, \ell_i)$ and $\widetilde{r}_i = \max(c, r_i)$, for some constant c. Given the two sorted vectors ℓ and r, the optimization becomes:

$$\min_c \sum_{i=1}^{n_1} (\ell_i - \min(c, \ell_i))^2 + \sum_{i=1}^{n_2} (r_i - \max(c, r_i))^2$$

This objective is convex and differentiable in c. Similar to the black box reshaping method, we can compute the derivatives at the values of the data and find where it flips

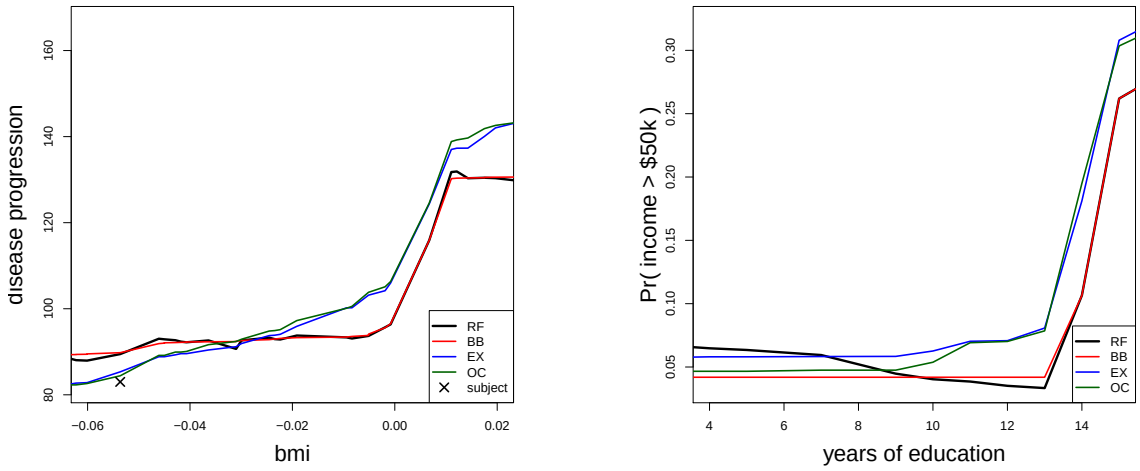
sign, then compute the minimizer in the corresponding segment. This takes $O(n)$ time where $n = n_1 + n_2$, the number of leaves under the shape-constrained split. With sorting, the over-constrained estimator can be computed in $O(n \log n)$ time for each shape-constrained split node.

We apply this procedure sequentially on the leaves of every shape-constrained node in reverse level-order on the tree.

1.4 Experiments

We apply the reshaping methods described above to two regression tasks and two binary classification tasks. We show that reshaping allows us to enforce shape constraints without compromising predictive accuracy. For convenience, we use the acronyms in Table (1.1) to refer to each method.

The BB method was implemented in R, and the OC and EX methods were implemented



(a) Diabetes dataset.

(b) Adult dataset.

Figure 1.4: Illustrating the result of reshaping. We choose a test point at random and make predictions with each model as we vary the predictor variable on the x-axis, holding all other variables constant. We see in both cases that the original RF predictions are not monotone-increasing. The Diabetes plot (1.4a) also shows the true value of the chosen data point with an X.

Table 1.1: Acronyms used for reshaping methods.

Method	Acronym	Sec
Random Forest	RF	-
Black Box Reshaping	BB	1.3.1
Exact Estimator	EX	1.3.2
Over-constrained Estimator	OC	1.3.2

Table 1.2: Experimental results of 5-fold cross-validation. Accuracy is measured by mean squared error (Diabetes), mean absolute percent error (Zillow), and classification accuracy (Adult, Spam). Standard errors are shows in parentheses.

Method	Diabetes	Zillow	Adult	Spam
RF	3209 (377)	2.594% (0.1%)	87.0% (1.7%)	95.4% (1.3%)
BB	3210 (390)	2.594% (0.1%)	87.1% (1.8%)	95.4% (1.3%)
EX	3154 (353)	-	86.8% (1.5%)	95.3% (1.2%)
OC	3155 (350)	2.588% (0.1%)	87.0% (1.5%)	95.3% (1.2%)

in R and C++, extending the R package `ranger` (Wright and Ziegler, 2017b); the forked repository can be found on github[1]. The exact estimator from Section (1.3.2) is computed using the MOSEK C++ package (ApS, 2017).

For binary classification, we use the probability tree implementation found in `ranger`, enforcing monotonicity of the probability of a positive classification with respect to the chosen predictors. For the purposes of these experiments, black box reshaping is applied to a traditional random forest. The random forest was fit with the default settings found in `ranger`.

We apply 5-fold cross validation on all four tasks and present the results under the relevant performance metrics in Table (1.2).

1.4.1 Diabetes Dataset

The diabetes dataset (Efron et al., 2004) consists of ten physiological baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements, for each of 442 patients. The response is a quantitative measure of disease progression measured one year after baseline.

Holding all other variables constant, we might expect disease progression to be monotonically increasing in body mass index (Ganz et al., 2014). We estimate a random forest and apply our reshaping techniques, then make predictions for a random test subject as we vary the body mass index predictor variable. The results shown in Figure (1.4a) illustrate the effect of reshaping on the predictions.

We use mean squared error to measure accuracy. The results in Table (1.2) indicate that the prediction accuracy of all four estimators is approximately the same.

1.4.2 Zillow Dataset

In this section, the regression task is to predict real estate sales prices using property information. The data were obtained from Zillow, an online real estate database company. For each of 206,820 properties, we are given the list price, number of bedrooms and bathrooms, square footage, build decade, sale year, major renovation year (if any), city, and metropolitan area. The response is the actual sale price of the home.

We reshape to enforce monotonicity of the sale price with respect to the list price. Due to the size of the constraint set, this problem becomes intractable for MOSEK; the results for the EX method are omitted. An interesting direction for future work is to investigate more efficient algorithms for this method.

Following reported results from Zillow, we use mean absolute percent error (MAPE) as our measure of accuracy. For an estimate $\widehat{y}$ of the true value y, the APE is $|\widehat{y} - y|/y$.

1. `https://github.com/mbonakda/ranger`

The results in Table (1.2) show that the performance across all estimators is indistinguishable.

1.4.3 Adult Dataset

We apply the reshaping techniques to the binary classification task found in the Adult dataset Lichman (2013). The task is to predict whether an individual's income is less than or greater than \$50,000. Following the experiments performed in Milani Fard et al. (2016) and You et al. (2017), we apply monotonic reshaping to four variables: capital gain, weekly hours of work, education level, and the gender wage gap.

We illustrate the effect of reshaping on the predictions in Figure (1.4b). The results in Table (1.2) show that we achieve similar test set accuracy before and after reshaping the random forest.

1.4.4 Spambase Dataset

Finally, we apply reshaping to classify whether an email is spam or not. The Spambase dataset (Lichman, 2013) contains 4,601 emails each with 57 predictors. There are 48 word frequency predictors, 6 character frequency predictors, and 3 predictors related to the number of capital letters appearing in the email.

That data were collected by Hewlett-Packard labs and donated by George Forman. One of the predictors is the frequency of the word "george", typically assumed to be an indicator of non-spam for this dataset. We reshape the predictions to enforce the probability of being classified as spam to be monotonically decreasing in the frequency of words "george" and "hp".

The results in Table (1.2) again show similar performance across all methods.

1.5 Discussion

We presented two strategies for prediction rule reshaping. We developed efficient algorithms to compute the reshaped estimators, and illustrated their properties on four datasets. Both approaches can be viewed as frameworks for developing more sophisticated reshaping techniques.

There are several ways that this work can be extended. Extensions to the black box method include adaptively combining rearrangements and isotonization (Chernozhukov et al., 2009), and considering a weighted objective function to account for the distance between test points.

In general, the random forest reshaping method can be viewed as operating on pre-trained parameters of a specific model. Applying this line of thinking to gradient boosted trees, deep learning methods, and other machine learning techniques could yield useful variants of this approach.

And finally, while practitioners might require certain shape-constraints on their predictions, many scientific applications also require inferential quantities, such as confidence intervals and confidence bands. Developing inferential procedures for reshaped predictions, similar to Chernozhukov et al. (2010) for rearrangements and Athey et al. (2018) for random forests, would yield interpretable predictions along with useful measures of uncertainty.

CHAPTER 2

DISCRETE CHOICE RANDOM FORESTS

2.1 Abstract

Discrete choice models (DCMs) are a popular probabilistic framework for analyzing consumer choice data, with widespread applications in areas such as economics, marketing, psychology, and finance. We propose a nonparametric DCM in which the utility of an item with respect to an agent is the sum of a fixed-effect nonlinear function, which we estimate with a random forest, and a random-effect linear function. For the fixed-effect component, we provide an estimation algorithm which is computationally scalable; with the random-effect component, our model avoids the undesirable Independence of Irrelevant Alternatives (IIA) property. The resulting framework, which we call Random Choice Forest (RCF), is shown in simulation to be accurate in both prediction and counterfactual analysis.

2.2 Introduction

Discrete choice models apply to situations where agents face a choice among a finite set of alternatives. For example, a customer chooses among competing products; a person chooses a transportation mode to take to work; a respondent chooses among a set of options on a survey. In the discrete choice setting, attributes are observed for both the agents and alternatives, and the goal is to understand the heterogeneity of preferences in the population in order to make accurate predictions about the future. Discrete choice models are typically derived under an assumption of utility-maximizing behavior; such models are called random utility models.

Random utility models assign a utility U_{hj} that agent h obtains from choosing alternative j. It is assumed that each agent chooses the item that maximizes utility. The utility is typically decomposed into two components so that $U_{hj} = f(X_{hj}) + \epsilon_{hj}$ where $X_{hj} \in \mathbb{R}^p$

is the observed feature vector containing features of both the item and the agent, and ϵ_{hj} represents latent aspects of the utility unknown to the observer. The easiest and most widely used discrete choice model is the multinomial logit (MNL). Under the MNL, f is assumed to be linear so that $f(X_{hj}) = X_{hj}^\top \beta$ and each ϵ_{hj} is independently, identically Gumbel distributed. Under this model, letting $Y_{hj} = 1$ if agent h chooses item j, it can be shown that the choice probabilities equal

$$p_{hj} = \mathbb{P}(Y_{hj} = 1 \,|\, \{X_{hj}\}_{j=1}^J) = \frac{\exp(f(X_{hj}))}{\sum_{k=1}^J \exp(f(X_{hk}))} = \frac{\exp(X_{hj}^\top \beta)}{\sum_{k=1}^J \exp(X_{hk}^\top \beta)}.$$

While random utility models were used much earlier, McFadden (1974) developed the foundations of the econometric analysis of discrete choice by deriving the MNL and using it to model transportation mode choices. The MNL has since been applied to a wide range of other fields such as marketing (Chintagunta and Nair, 2011), finance (Uhler and Cragg, 1971), and economic demand estimation (Revelt and Train, 1998). Despite the prevalence of the MNL, the assumption that utility is a linear function of the features is restrictive and unrealistic in many real world applications, and can lead to issues with interpretability and prediction. Kleinberg et al. (2015) discuss how improved predictions with machine learning techniques can have large social impact. In order to remove the linearity assumption, we define a nonparametric DCM in which we take $f(X_{hj})$ to be a nonlinear function of the features. We propose a method to estimate f as an aggregation of bootstrapped decision trees in the same style as the random forest algorithm for regression and classification.

Growing a single decision tree in the DCM setting is not straightforward because the estimated probability $\mathbb{P}(Y_{hj} = 1 \,|\, \{X_{hj}\}_{l=1}^J)$ is a function of all J feature vectors $X_{h1}, \ldots, X_{hJ}$. Thus, predicting the choice of a single agent h may involve up to J leaf nodes. Since the predictions depend on many leaf nodes, one cannot directly apply the greedy splitting procedures used in standard regression and classification trees. Instead, we propose a computationally efficient algorithm based on partial maximization of the likelihood. The resulting estimator

is called a Random Choice Forest (RCF).

In addition to linearity , the MNL model has other restrictive limitations for practitioners. One major limitation is that the MNL does not model heterogeneity of preferences across the population; if utilities from agent to agent vary randomly, or vary with latent variables, the MNL is misspecified, and can lead to nonsensical conclusions. Relatedly, the MNL implies proportional substitution patterns across all alternatives. This implication is implausible for alternative sets containing choices that are close substitutes – we would expect these items to be interchanged more frequently than disparate goods . We describe these limitations in detail in Section (2.5.2).

One popular approach for addressing these issues in the MNL model is the so-called mixed multinomial logit model (MML), where $U_{hj} = \beta^\top X_{hj} + \gamma_h^\top Z_{hj} + \epsilon_{hj}$, γ_h is a random vector with a distribution of, e.g. $N(0, \Sigma)$, and X_{hj} and Z_{hj} are (possibly equal) feature vectors. One estimates the random effects distribution from data and the predicted probability p_{hj} is then

$$\int_{\mathbb{R}^p} \frac{\exp(\beta^\top X_{hj} + \gamma_h^\top Z_{hj})}{\sum_{l=1}^{J} \exp(\beta^\top X_{hl} + \gamma_h^\top Z_{hl}))} \, p(\gamma_h) d\gamma_h.$$

We develop a mixed-effects variant of the Random Choice Forest where we let $U_{hj} = g(X_{hj}) + \gamma_h^\top Z_{hj} + \epsilon_{hj}$; the function $g(X_{hj})$ is a nonlinear term that we estimate via aggregation of decision trees and γ_h is a random vector with a specified distribution.

2.3 Related Work

In this section, we describe related work applying both nonparametric methods and machine learning tools to the problem of discrete choice modeling. One line of work called the BLP model (Berry et al., 1995) came out of the Industrial Organization (IO) literature. BLP uses random effects in addition to latent characteristics to model choice. This work set the stage for much of the later discrete choice work in IO. This work emphasized modeling choices under a large number of alternatives in addition to potential endogeneity when the

assumption of iid residuals is violated. In the BLP model, utility is modeled as

$$U_{hj} = \beta_h^\top X_j + \gamma_j + \epsilon_{hj}$$

where β_h are random coefficients and γ_j is a latent product characteristic. This component allows the model to describe unexplained patterns of demand.

Another line of work draws inspiration from embedding methods for arbitrary high-dimensional objects (Rudolph et al., 2016, 2017); instead of assuming one latent characteristic like Berry et al. (1995), these methods estimate high-dimensional latent vectors. These methods assume embeddings are parameters (or latent variables) of a probabilistic model that can be estimated through posterior inference. The resulting embeddings are then used to represent objects in the dataset of interest. These embedding methods inspired a class of discrete choice models with large numbers of latent variables. Advances in variational inference were leveraged to conduct large-scale posterior inference. We first briefly outline the embedding methodology before moving on to explain the methods for discrete choice.

Exponential family embeddings (Rudolph et al., 2016) are a class of embedding methods for arbitrary high-dimensional objects. The exposition here assumes that we are interested in constructing embeddings for items in a grocery store but these methods can be applied to arbitrary objects. For each item i, let c_i denote the set of indices representing the other items in the shopping bakset. The conditional distribution of the quantity of item i in a shopping basket, denoted x_i, given its "context" — the other items in the shopping basket — x_{c_i} is modeled as an appropriate exponential family distribution (here, Poisson) with natural parameters $\eta_i(x_{c_i})$ and sufficient statistics $t(x_i)$. The distribution is parameterized with embedding vectors associated with the data point and its context so that

$$\eta_i(x_{c_i}) = f_i\left(\rho_i^\top \sum_{j \in c_i} \alpha_j x_j\right)$$

where $\rho_i \in \mathbb{R}^k$ denotes the embedding of the ith data point, and $\alpha_i \in \mathbb{R}^k$ denotes the context vector governing the distribution of data that appears in i's context. In the parlance of generalized lienar models, $f_i(\dot{})$ is called the link function. Inference is done by maximum likelihood estimation using stochastic gradients. Rudolph et al. (2016) use a conditional Poisson distribution to model count data from shopping basket data from a grocery store. This results in embedding representations of items from the grocery store and the authors show that these embeddings can be used to compare items using the resulting vector representations. Rudolph et al. (2017) extends this work by placing the exponential family embedding model within a hierarchical modeling framework to appropriately model grouped data. The authors show how this can be used to construct time-dependent embeddings so that the notion of similar goods is a function of the time of year. This embedding paradigm was used as inspiration for a number of discrete choice models.

Ruiz et al. (2017) develop a sequential probabilistic model of shopping data called `SHOPPER` that leverages the idea of representing items with latent vectors. This approach pays close attention to how items in a shopping basket interact and can be used to answer counterfactual questions about changes in price. A core component of the `SHOPPER` model is that items are representated as latent vector representations estimated from shopping basket data. While the embedding models described above model the conditional distribution of each item given the others, `SHOPPER` directly models the joint distribution of the items, allowing the practitioner to leverage important information such as price and customer heterogeneity.

The `SHOPPER` model is described in terms of its generative process — each customer chooses an item to put in her basket given the items chosen so far. The first item chosen is the one that maximizes the agents utility unconditionally, and the second item maximizes her utility given the item she just chose, etc. The model allows for interaction effects to account for substitutes and complements. Consider the choice of the ith item and let y_{i-1} denote the items chosen so far. Then, the customer chooses the item c that maximizes the utility

$U_c(y_{i-1}) = V(c, y_{i-1}) + \epsilon_c$, where ϵ_c is assumed to be sampled iid from a Gumbel distribution as in the standard multinomial logit model. The fundamental modeling assumption is that the the representative utility $V(c, y_{i-1})$ has the following form,

$$V(c, y_{i-1}) = \psi_c + \rho_c^\top \left(\frac{1}{i-1} \sum_{j=1}^{i-1} \alpha_{y_j} \right)$$

where the second term is the familiar expression taken from the exponential family embedding methods described above. The first term ψ_c is a latent variable that varies by item and is used to model preference heterogeneity, seasonality, and price.

In the second term, ρ_c represents the per-item interaction coefficients, and α_c represents item attributes. When $\rho_c^\top \alpha_{c'}$ is large, then having c' in the basket increases the utility of choosing c (the items are complements); conversely, a large negative value would signify substitutes.

The ψ_c term allows the practitioner to use domain expertise to enrich the model. In the paper, they give an example where $\psi_c = \lambda_c + \theta_u^\top \alpha_c$ where λ_c is the overall item popularity. θ_u is a latent vector associated with each customer so that $\theta_u^\top \alpha_c$ increases for items c that the customer tends to purchase. The paper develops a variational inference algorithm to perform posterior inference on this model. This methodology was extended to other datasets in a similar manner in Athey et al. (2018). These methods were both shown to outperform the standard multinomial logit model on various prediction tasks. Instead of focusing on parametric models with latent variables, the methodology we propose in this chapter aims to provide a nonparametric approach to estimating the utility function. The hope is to provide a method that requires minimal modeling or feature engineering, providing practitioners with a strong baseline method for discrete choice prediction. Next we we describe several projects that align more closely with our goals.

Paredes et al. (2017) compare standard multinomial logit models of discrete choice to machine learning methods. They apply these methods to a stated-preference discrete choice

dataset obtained from Signapore's Land Transport Authority aimed at understanding the traveling behaviors of Singapore residents. The goal was to predict the number of cars a household owned based on various demographic variables. They were able to apply machine learning methods like random forests because their dataset did not contain any item-specific attributes such as price. This results in a simple classification problem where the goal is to classify each household into one of three categories: 0, 1, or 2 cars. This is not the standard discrete choice setting, and the machine learning methods they use would not be appropriate once item-specific attributes are considered. The work in this chapter provides a principled way of fitting random forests in the standard discrete choice setting with both item- and agent-specific characteristics.

A natural approach for applying neural networks to discrete choice was proposed by Hruschka et al. (2001), where the representative utility is modeled as a k-layer neural network. Another recent line of work applied to the travel industry considers the application of neural networks to discrete choice when a sequence of alternatives is presented to a customer (Mottini and Acuna-Agost, 2017). This work focused on the particular problem of airline itinerary choice. Similar to the `SHOPPER` setting, the approach in Mottini and Acuna-Agost (2017) assumes a sequential decision making task. Their model combines recurrent neural networks with the attention mechanism in an encoder-decoder architecture. The input to the model is a sequence corresponding to the choice set, and the output is a pointer to the most probable alternative. The authors note that this model presents significant computational constraints thereby limiting the scale and number of tests they could perform. This is a significant issue for neural networks since hyperparameter tuning has been shown to be a crucial component of the fitting procedure.

There are also a number of existing nonparametric approaches to discrete choice. Matzkin (1993) studies the fully nonparametric specification where the deterministic component V_{hj} is nonparametric and the random component ϵ_{hj} is distribution-free. Berry and Haile (2009) considers the identification of nested discrete choice models with nonparametric deterministic

components. Another class of nonparametric methods treat the deterministic component as linear, as in the MNL, but focus on a distribution-free stochastic component (Horowitz et al., 1994). Abe (1999) propose an application of generalized additive models (Hastie and Tibshirani, 1986) to discrete choice, treating the deterministic component as nonparametric and the stochastic component as Gumbel as in the MNL. The derived algorithm in Abe (1999) reduces to the backfitting procedure used to fit GAMs in Hastie and Tibshirani (1986). This approach reduces the amount of subjectivity required for the standard MNL and enables exploratory analysis of the fitted functions. It can be used as a tool for specifying parametric models. However, this approach does not allow for a data-driven approach to modeling interaction effects. One benefit of modern machine learning methods is that they require minimal legwork to arrive at a strong baseline model. In Section (2.5), we describe our approach to providing a general purpose machine learning method for discrete choice.

2.4 Alternative Approaches

First we describe several out-of-the-box approaches that we did not pursue in this work. One approach is to fit a standard random forest for binary classification for each item. This would treat each Y_{hj} as an independent response, making this a one-vs-all method. The resulting random forests could be used at prediction time on each alternative, choosing the alternative with largest predicted probability as the predicted choice. However, this approach ignores the data-generating process and the comparative aspect of the alternatives. In particular, it ignores the fact that choices are made from a finite set of objects, and that those objects are known at decision time. This discrete choice approach we suggest in this work directly models this choice process.

If the choice dataset does not contain alternative-specific predictor variables, we can simply apply a standard random forest used for classification. In this setting, the random forest approach would classify individuals into classes that correspond to different alternatives as done in Paredes et al. (2017). However, the addition of alternative-specific predictor vari-

ables makes the application of random forests non-trivial. This question[1] on StackExchange was posted by a researcher looking for a method to fill this gap.

Another out-of-the-box approach to using random forests for discrete choice is something we called the *stacked model*. Let $\mathbf{X}_h$ be the stacked covariates so that $\mathbf{X}_h = [X_{h1}; X_{h2}; ...; X_{hJ}]$ where $X_{hj} \in \mathbb{R}^p$ and J is the number of items. We associate each leaf node b with a vector $(V_{b1}, ..., V_{bJ}) \in \mathbb{R}^J$. Then, we can define a vector valued function such that

$$\mathbf{V}(\mathbf{X}_h) = (V_{b(\mathbf{X}_h),1}, ..., V_{b(\mathbf{X}_h),J}).$$

We then define the model

$$P(Y_{hj} = 1 \,|\, \mathbf{X}_h) = \frac{\exp(V_{b(X_h)j})}{\sum_{l \in A_h} \exp(V_{b(X_h)l})}.$$

To estimate the model via decision tree, we greedily split on the feature space until we get to leaf nodes and we estimate $\mathbf{V}_b$ for each leaf node b by empirical counts.

The problem here is that it is impossible to split on item features to distinguish the different items. For simplicity, suppose $j \in \{\text{car}, \text{train}, \text{bus}\}$ and suppose $\mathbf{X}_h$ does not contain any consumer-specific features. Then, for every h, $\mathbf{X}_h$ is of the form

$$\mathbf{X}_h = \left[\; X_{\text{car, price}}, X_{\text{car, time}}, X_{\text{train, price}}, X_{\text{train, time}}, X_{\text{bus, price}}, X_{\text{bus, time}} \; \right]$$

Splitting on the first (or any) coordinate of $\mathbf{X}_h$ is futile because $X_{\text{car, price}}$ is the same for all h. In Section (2.7), we describe a final alternative approach that could be thought of as a simpler method than the one we describe next.

1. https://stats.stackexchange.com/questions/340457/using-random-forests-for-modeling-discrete-choice-problems

2.5 Model and Method

For agent $h \in \{1, \ldots, H\}$ and item $j \in \{1, \ldots, J\}$, we let $X_{hj} \in \mathbb{R}^{p_1}$ and $Z_{hj} \in \mathbb{R}^{p_2}$ be feature vectors that might share some or all attributes of both the item and the agent. We suppose that the agents make choices independent of each other. Our model for the choice of agent h is then:

$$p_{hj} = \int_{\mathbb{R}^{p_2}} \frac{\exp(g(X_{hj}) + \gamma_h^\top Z_{hj})}{\sum_{l=1}^{J} \exp(g(X_{hl}) + \gamma_h^\top Z_{hl})} \, p_\theta(\gamma_h) d\gamma_h.$$

where p_θ is the probability density function of the random effects γ_h. Our model is parametrized by a nonlinear $g : \mathbb{R}^{p_1} \to \mathbb{R}$ and θ.

We propose a two-stage estimation procedure similar in spirit to Hajjem et al. (2014). In the first stage, the fixed effects component, g, is estimated with a Random Choice Forest as described in Section (2.5.1). With the estimate of g fixed, the second stage estimates the posterior of the random effects.

2.5.1 Random Choice Forest

Let $\mathcal{L}$ denote a set of axis-aligned hyperrectangles defined by a decision tree. Each hyperrectangle in $\mathcal{L}$ corresponds to a leaf in the decision tree. Denote $b : \mathbb{R} \to \mathcal{L}$ as the function that maps a point in the input space to the hyperrectangle in $\mathcal{L}$ that contains it. Under this setting, we assume the following random utility model:

$$U_{hj} = V_{b(X_{hj})} + \epsilon_{hj}$$

with ϵ_{hj} drawn iid from a Gumbel distribution. The choice probabilities under this model are

$$p_{hj} = P(Y_{hj} = 1 \,|\, \{X_{hj}\}_{l=1}^{J}) = \frac{\exp(V_{b(X_{hj})})}{\sum_{l=1}^{J} \exp(V_{b(X_{hl})})}.$$

Let $\mathcal{D} = \{(\{X_{hj}\}_{j=1}^{J}, \mathbf{Y_h})\}_{h=1}^{H}$ denote observed choice data where $\mathbf{Y_h}$ is a one-hot vector representing consumer h's choice. We grow the tree by partial maximum log-likelihood; we say "partial" because we fix the values of all leaves except the leaves we are creating by performing the current split.

Assume we are considering a split of parent node $b^\star$ into left and right children b^L and b^R. For each possible split such that $b^L \cup b^R = b^\star$, we consider the MLE score

$$s(b^\star,\{b^L,b^R\}) = \sup_{\{V_L,V_R\}} \sum_{h\in b^\star} \sum_{j=1}^{J} Y_{hj} \left(V_{b(X_{hj})} - \log \left(\sum_{l=1}^{J} \exp(V_{b(X_{hl})}) \right) \right)$$

where $h \in b^\star$ is the set $\{h : \exists j \text{ such that } X_{hj} \in b^\star\}$ of all agents in the training data who have at least one feature vector in $b^\star$. We aim to choose the split that maximizes the increase in the overall likelihood of the tree, while fixing all leaf values except V_L and V_R.

Unlike standard decision trees, the parameter estimates for this method are tied across leaves. The score for splitting $b^\star$ might depend on other leaf values. For computational efficiency and determinism, we split the leaves in level-order on the tree. The bootstrapped choice data used for each tree are sampled at the agent level.

The maximum likelihood optimization is performed with damped Newton's method (Newton's method with backtracking line search). However, performing this optimization for every possible split option is computationally intractable for large-scale problems.

Approximate Split-Finding Algorithm

We propose an approximate split-finding algorithm which obviates the need to perform an expensive optimization for each potential split. Assume we are splitting node $b^\star$ into left and right children b^L and b^R. Let V_L and V_R denote the leaf values of b^L and b^R respectively, and

let $\widetilde{V}$ denote the set of estimated utility values at all other leaves. Then the log-likelihood is

$$\ell(V_L, V_R, \widetilde{V}) = \sum_{h \in b^\star} \sum_{j=1}^{J} Y_{hj} \left(V_{b(X_{hj})} - \log \left(\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})}) \right) \right)$$

The partial derivative with respect to V_L is

$$\frac{\partial \ell}{\partial V_L} = C_L - \sum_{h \in b^\star} p_h(V_L) n_h(L)$$

where $C_L = |\{(h, j) : X_{hj} \in b^L \text{ and } Y_{hj} = 1\}|$ is the number of positive choices in leaf b^L, $n_h(L)$ is the number of samples associated with agent h in leaf b^L, and $p_h(V_L)$ is the estimated model probability that agent h will choose an item in leaf b^L.

Let V denote the current estimated utility assigned to $b^\star$, the node we are splitting. Evaluating the partial derivative at this current value so that $V_L = V$, we have

$$\frac{\partial \ell}{\partial V_L}\Big|_{V_L = V} = C_L - \sum_{h \in b^\star} p_h(V) n_h(L)$$

This is the difference between the observed and expected counts in leaf b^L. The same is true for the right leaf.

Intuitively, these partial derivatives will be large for splits under which the model can be improved. We therefore score each split with the norm of the gradient, $\|\nabla \ell\|$. We choose the split with largest gradient norm and perform damped Newton's method for the chosen split to obtain the estimated leaf values V_L and V_R. Similar to approaches for scanning potential splits in regression and classification trees, this gradient can be quickly updated online as we scan through the potential split points. This approach is far more efficient than performing damped Newton's method on each potential split, and we find that it yields similar out-of-sample performance.

We compare out-of-sample log-likelihood vs. wall clock time for the full split-finding

method and the proposed approximate split-finding method in Figure (2.1). These simulations were done on data obtained from the synthetic experiment in Section (2.6.1) for varying tree heights. We find that the approximate method maintains similar out-of-sample performance while obtaining up to a 500x improvement in wall clock time. Care was taken to optimize the full method.

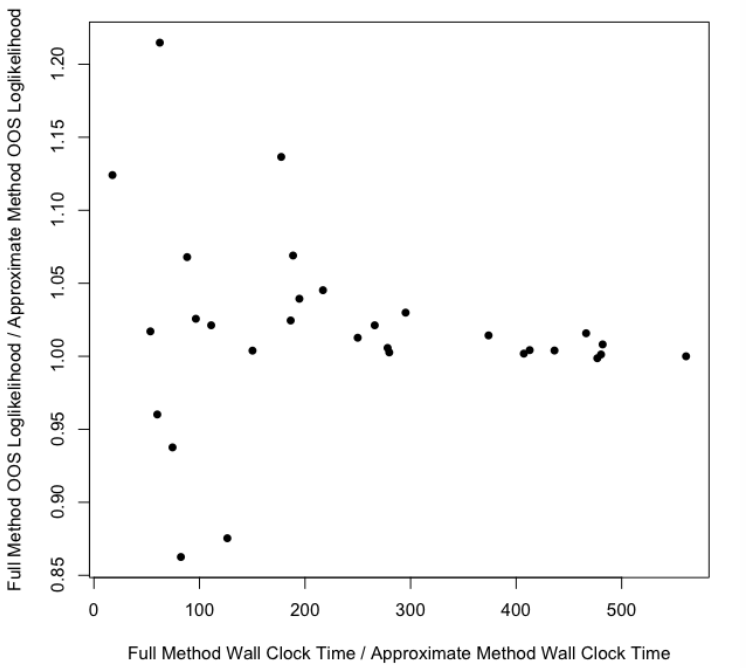


Figure 2.1: Full vs Approximate Split-Finding Methods. The approximate split finding method maintains similar out-of-sample performance while obtaining up to a 500x improvement in wall clock time.

Random Choice Forest Algorithm

In this section, we describe the full RCF algorithm in detail. First consider the choice probability for agent h and item j,

$$p_{hj} = \frac{e^{V_{b(X_{hj})}}}{\sum_k e^{V_{b(X_{hk})}}}$$

Adding a constant value c to the values of each leaf will leave this probability unchanged and therefore will leave the likelihood unchanged. We considered two solutions to this problem of unidentifiability. The first approach was to perform maximum likelihood with the additional constraint that all leaf values sum to zero. The problem with this approach is that the value of the likelihood makes discontinuous jumps as we split leaves. For example, assume we are splitting a leaf with value V into left and right leaves with values V_L and V_R. Since we are doing partial maximum likelihood maximization where all other leaf values are fixed, this imposes the constraints that $V_L + V_R = V$. As a result, we must choose initial values for V_L and V_R in our optimization procedure to satisfy this constraint – these values would result in a discontinuous change in the likelihood function. Instead, if we were to let the initial values of the leaves be $V_L = V_R = V$, then the initial value of the likelihood would match the current likelihood of the tree, and the optimization would be guaranteed to maintain the current likelihood or increase it. That leads to the second solution – the solution we choose to adopt. On each split, we perform unconstrained partial maximum likelihood to estimate V_L and V_R. Then, we compute the mean value of all leaves in the tree and subtract it from each leaf value. This maintains the likelihood value and ensures that it is at least as good as the constrained version of the maximization. We also found that enforcing this sum-to-zero constraint has numerical advantages, prohibiting leaf values from growing large in magnitude.

For each candidate predictor variable, the maximization procedure is performed on the split chosen by the approximate split-finding procedure described in Section (2.5.1). Our experiments showed that performing Newton's method far outperformed gradient descent on the optimization in both wall-clock time and number of iterations. The first split in the tree is a special case. Because of the unidentifiability described above, the Hessian is singular, precluding the use of Newton's method. Instead, for the first split, we perform maximum likelihood with the constraint that $V_L + V_R = 0$, applying a univariate Newton's method to estimate for V_L and V_R. After the first split, damped Newton's method can be used because

we are doing *partial* maximum likelihood – this results in a nonsingular Hessian.

Assume we are splitting node $b^\star$ into left and right children b^L and b^R. We will estimate the utilities in these leaves, V_L and V_R, by maximizing the likelihood corresponding to the leaves. Let $\widetilde{V}$ denote the current estimated utility values at all other leaves. Then,

$$\ell(V_L, V_R, \widetilde{V}) = \sum_{h \in b^\star} \sum_{j=1}^{J} Y_{hj} \left(V_{b(X_{hj})} - \log \left(\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})}) \right) \right)$$

The partial derivative with respect to V_L is therefore

$$\begin{aligned} \frac{\partial \ell}{\partial V_L} &= \sum_{(h,j) \in b^L} Y_{hj} - \sum_{h \in b^\star} \sum_{j=1}^{J} Y_{hj} \frac{\sum_{X_{hl} \in b^L} \exp(V_L)}{\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})})} \\ &= C_L - \sum_{h \in b^\star} \frac{n_h(L) \exp(V_L)}{\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})})} \end{aligned}$$

where $C_L = |\{(h,j) : X_{hj} \in b^L \text{and} Y_{hj} = 1\}|$ is the number of positive choices in leaf b^L, and $n_h(L)$ is the number of samples associated with agent h in leaf b^L. Similarly,

$$\frac{\partial \ell}{\partial V_R} = C_R - \sum_{h \in b^\star} \frac{n_h(R) \exp(V_R)}{\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})})}$$

Let $\nabla \ell$ denote the column vector $\left[\frac{\partial \ell}{\partial V_L} \frac{\partial \ell}{\partial V_R} \right]^\top$. The second partials are,

$$\begin{aligned} \frac{\partial^2 \ell}{\partial V_L^2} &= -\sum_{h \in b^\star} \left(\frac{n_h(L) \exp(V_L) \left(\left(\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})}) \right) - n_h(L) \exp(V_L) \right)}{\left(\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})}) \right)^2} \right) \\ \frac{\partial^2 \ell}{\partial V_R^2} &= -\sum_{h \in b^\star} \left(\frac{n_h(R) \exp(V_R) \left(\left(\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})}) \right) - n_h(R) \exp(V_R) \right)}{\left(\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})}) \right)^2} \right) \\ \frac{\partial^2 \ell}{\partial V_L \partial V_R} &= -\sum_{h \in b^\star} \left(\frac{-n_h(L) \exp(V_L) \times n_h(R) \exp(V_R)}{\left(\sum_{\ell=1}^{J} \exp(V_{b(X_{h,\ell})}) \right)^2} \right) \end{aligned}$$

And let $\nabla^2 \ell$ denote

$$\nabla^2 \ell = \begin{bmatrix} \frac{\partial^2 \ell}{\partial V_L^2} & \frac{\partial^2 \ell}{\partial V_L \partial V_R} \\ \frac{\partial^2 \ell}{\partial V_R \partial V_L} & \frac{\partial^2 \ell}{\partial V_R^2} \end{bmatrix}$$

Estimating $V = [V_L \quad V_R]^\top$ with Newton's method would involve the following iterative update rule:

$$V^{k+1} = V^k + \eta(-\nabla^2 \ell)^{-1} \nabla \ell$$

with $\eta = 1$. However, in practice, we use damped Newton's method (i.e. Newton's method with line search). We choose parameters $0 < \alpha \leq \frac{1}{2}, 0 < \beta < 1$ and at each iteration, starting with $\eta = 1$, and while

$$\ell(V + \eta v) < \ell(V) + \alpha \eta \nabla \ell(V)^\top v$$

we shrink $\eta = \beta \eta$. Here, $v = -(\nabla^2 \ell(V))^{-1} \nabla \ell(V)$. After each split, we adjust all leaf values to ensure they sum to zero. After the tree growing process is complete, we perform one final likelihood maximization over all leaf values using gradient descent.

The number of optimizations required to grow a single tree can be enormous, so careful attention to the implementation is necessary to ensure tractability. For example, careful use of hash maps to keep track of which leaves each agent's samples are in greatly improves the efficiency of the likelihood and gradient calculations.

2.5.2 Random Effects

In addition to linearity, the MNL has a number of other restrictive limitations (Train (2009), Chapter 3); in this section, we highlight two of these limitations and discuss how to handle them by introducing random effects. First, the MNL model captures taste variations in the population that vary with observed variables, but is unable to handle preferences that change based on unobserved variables. And while the MNL may still be able to predict average behavior well, it does not provide inferences about the distribution of tastes. This

kind of information can be useful to decision makers implementing policies that affect only part of the population.

Additionally, when the features of an alternative or agent changes, the MNL model implies a restrictive substitution pattern across alternatives. This limitation can be seen by inspecting the ratios of probabilities between alternatives. The ratio of choice probabilities between alternatives j and k, $\frac{\mathbb{P}(Y_{hj}=1 \mid \{X_{hj}\}_{l=1}^{J})}{\mathbb{P}(Y_{hk}=1 \mid \{X_{hj}\}_{l=1}^{J})}$ represents the relative odds of choosing j over k. Under the MNL, this ratio is $\exp(\beta^{\top}(X_{hj} - X_{hk}))$ and does not depend on X_{hl} for $l \neq j, k$. Because this ratio does not depend on other (irrelevant) alternatives, the MNL is said to exhibit *independence from irrelevant alternatives*, or IIA. This is problematic because discrete choice models are often used for counterfactual analysis where one adds, removes, or modifies an item and uses the model to predict the resulting change in consumer preferences. This issue can be seen through a famous example called the red-bus-blue-bus problem.

This particular example is taken from Train (2009). Assume a decision maker has the choice of traveling to work with either a blue bus or a car. For simplicity, assume that the choice probabilities of the two modes of transportation are equal so that $p_{\text{car}} = p_{\text{blue bus}} = \frac{1}{2}$ so that the ratio of the choice probabilities is 1. Suppose that suddenly a red bus is introduced into the choice set, and assume this red bus matches the blue bus on all characteristics except color. The choice probabilities of the two buses are therefore equal so that their ratio $p_{\text{blue bus}}/p_{\text{red bus}} = 1$. However, due to IIA, $p_{\text{car}}/p_{\text{blue bus}}$ will remain at 1, since this ratio does not depend on other alternatives (namely, the red bus) due to IIA. The only probabilities that satisfy $p_{\text{blue bus}}/p_{\text{red bus}} = p_{\text{car}}/p_{\text{blue bus}} = 1$ are $p_{\text{blue bus}} = p_{\text{red bus}} = p_{\text{car}} = \frac{1}{3}$. However, realistically, the probability of taking a car should be unchanged with the addition of a new bus that matches the old bus exactly. This is an extreme example, but a similar story holds if the new bus was an approximate substitute for the old bus. One would expect that the new bus would reduce the probability of choosing the old bus by a greater amount than it reduces the probability of choosing car.

Introducing random effects into the model specification allows us to at once solve both

limitations described above. The mixed multinomial logit (MML) model allows agent preferences to vary across the population. It also allows the modeler to explicitly specify the correlation of utilities across alternatives. In the linear setting, the MML utility accrued by agent h choosing item j is $U_{hj} = \beta^\top X_{hj} + \gamma_h^\top Z_{hj} + \epsilon_{hj}$ where X_{hj} and Z_{hj} are observed features. The γ_h coefficients are modeled as independent samples from a prior distribution $p(\gamma_h|\phi)$, where ϕ is a hyperparameter. A fully Bayesian procedure would also specify a hyperprior distribution for ϕ. The specification of Z_{hj} allows the practitioner to specify the correlation structure across alternatives. In the standard MNL, $Z_{hj} = 0$ so that there is no correlation between alternatives; this leads to the IIA property. If Σ is the covariance matrix of γ_h, then the covariance between the utility for items j and k is $Z_{hj}^T \Sigma Z_{hk}$.

Incorporating random effects naively into the leaves of a random forest can lead to identifiability issues. Consider the model

$$U_{hj} = V_{b(X_{hj})} + \nu_{b(X_{hj})} + \epsilon_{hj}$$

where the $\nu_{b(X_{hj})}$'s are independent random effects and ϵ_{hj} are the stochastic component drawn as iid Gumbel. Let B be the total number of leaf nodes. The conditional probability determined by this model is

$$\mathbb{P}(Y_{hj} = 1 \,|\, \{X_{hj}\}_{j=1}^J) = \int_{\mathbb{R}^B} \frac{e^{V_{b(X_{hj})} + \nu_{b(X_{hj})}}}{\sum_{l=1}^J e^{V_{b(X_{hl})} + \nu_{b(X_{hl})}}} \, d\nu_1 \, d\nu_2 \ldots \, d\nu_B$$

Identifiability Problem: Let $\widetilde{V}_b$ be another set of real numbers for each leaf node b. Define $\widetilde{\mathbb{P}}(Y_{hj} \,|\, \{X_{hj}\}_{j=1}^J) = \dfrac{e^{\widetilde{V}_{b(X_{hj})}}}{\sum_{l=1}^J e^{\widetilde{V}_{b(X_{hl})}}}$. It may be that, for all values of $\{X_{hj}\}_{j=1}^J$, and for all h, j, we have that

$$\widetilde{\mathbb{P}}(Y_{hj} = 1 \,|\, \{X_{hj}\}_j) = \mathbb{P}(Y_{hj} = 1 \,|\, \{X_{hj}\}_j)$$

This is a problem because based on the data $(\mathbf{Y}_h, \{X_{hj}\}_{j=1}^J)$, we cannot determine whether

$\widetilde{\mathbb{P}}$ or $\mathbb{P}$ is the right model. This is illustrated in the following example:

Case 1 1/3 of the people are high income individuals, 2/3 are low income. All high income people prefer cars, all low income people prefer buses.

Case 2 Everyone prefer cars 1/3 of the time and buses 2/3 of the time.

We do not observe whether a consumer is high or low income. Define $V^h_{car} = 100$, $V^h_{bus} = 0$, $V^l_{car} = 0$, and $V^l_{bus} = 100$, where h stands for high income and l stands for low income. Define also $\widetilde{V}_{car} = 1$ and $\widetilde{V}_{bus} = 1 + \log 2$.

Then, we have that

$$\frac{1}{3}\frac{e^{V^h_{car}}}{e^{V^h_{car}} + e^{V^h_{bus}}} + \frac{2}{3}\frac{e^{V^l_{car}}}{e^{V^l_{car}} + e^{V^l_{bus}}} = \frac{e^{\widetilde{V}_{car}}}{e^{\widetilde{V}_{car}} + e^{\widetilde{V}_{bus}}} = \frac{1}{3}.$$

Therefore, based on the data alone, it is impossible to distinguish Case 1 from Case 2. Instead, we propose the following model:

$$U_{hj} = V_{b(X_{hj})} + f(X_{hj}) + \epsilon_{hj}$$

where $f(X_{hj})$ is a random function and ϵ_{hj} is still iid Gumbel. In our experiments below, we assume f is linear so that $U_{hj} = g(X_{hj}) + \gamma_h^\top Z_{hj} + \epsilon_{hj}$ where γ_h is random. g is first estimated with a Random Choice Forest described in Section (2.5.1), then, with g fixed, we perform posterior inference to estimate the random effects distribution.

2.6 Experiments

For stability and efficiency, the Random Choice Forest described in Section (2.5.1) was implemented in `C++` by extending the `ranger` package (Wright and Ziegler, 2017a); the forked repository can be found on github[2]. Posterior inference for the random effects component

2. `https://github.com/mbonakda/ranger`

was performed using `stan` (Stan Development Team, 2018), a probabilistic programming language.

The intention of the experiments below is to illustrate the advantage of using RCFs over of the standard multinomial logit model (MNL). We show that the RCF outperforms the MNL without feature engineering or careful attention to specifying prior distributions.

2.6.1 Friedman Model

In this section, we compare Random Choice Forests to the standard multinomial logit model on a synthetic dataset. We compare a purely fixed effects synthetic model in addition to a mixed effects synthetic model. We use a model first introduced by Friedman (1991) as the true utility function:

$$U_{hj} = 10\sin(\pi X_{1,hj} X_{2,hj}) + 20(X_{3,hj} - 0.5)^2 + 10X_{4,hj} + 5X_{5,hj} \tag{2.6.1}$$

where each covariate is iid $U[0,1]$. We also add five irrelevant predictors, also distributed as $U[0,1]$. This model has become a popular benchmark for evaluating nonparametric methods (Friedberg et al., 2018; Taddy et al., 2015). For interpretability, we assume that each agent is choosing between three modes of transportation so that $j \in \{\text{blue bus, train, car}\}$.

For each agent and item, we sample ten predictor variables from $U[0,1]$ and compute the corresponding utilities according to Equation (2.6.1). Each agent's choice is then sampled according to the implied choice probabilities $p_{hj} = \exp(U_{hj})/\sum_k \exp(U_{hk})$. We fit a Random Choice Forest and a multinomial logit model on the resulting synthetic choice data, and compare them with out-of-sample log-likelihood assigned by each model.

We first generate data for $H = 3000$ agents and perform 5-fold cross-validation to tune the following parameters of the RCF: `mtry` (the number of variables to sample at each split), `num.trees`, and `height`. We then compare out-of-sample performance on a held-out test set. With traditional random forests, the effects of these parameters are well understood,

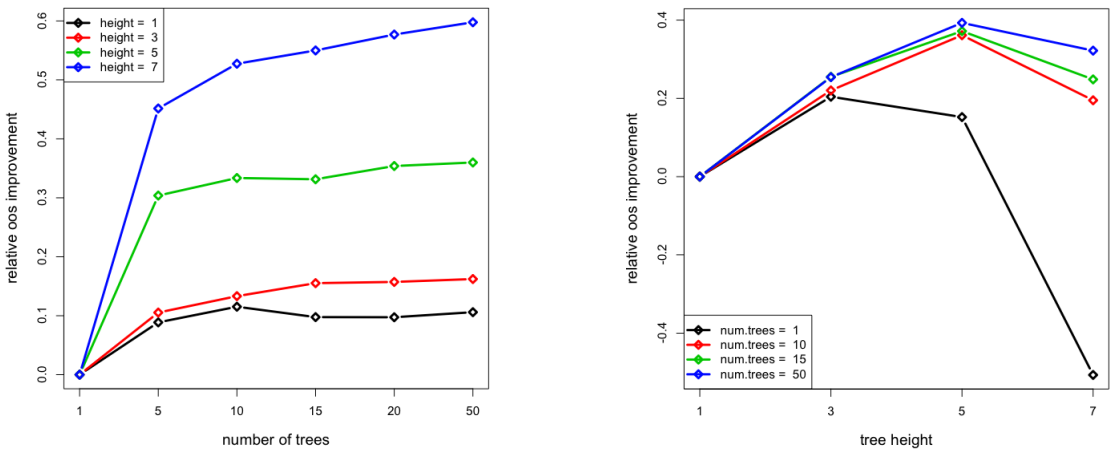
and tuning them out-of-sample allows the practitioner to explore the bias-variance trade-off. Since the parameter estimates of an RCF are tied across leaves, it is worthwhile to confirm that RCFs maintain the same behavior as traditonal random forests. We illustrate the effect of varying these hyperparameters in Figure (2.2) and find that RCFs exhibit the expected behavior.

Table (2.1) shows that the Random Choice Forest outperforms the multinomial logit model. Here we show the performance of the RCF with hyperparameters chosen by cross-validation.

method	average oos log-likelihood (se)
MNL	-617 (11)
RCF	-522 (49)

Table 2.1: Held-out log-likelihood. (Fixed Effects)

Next, we construct a dataset from the same model as before, except now we add a random effect on a bus indicator variable. This model is equivalent to a nested logit model where utilities are correlated among buses. This particular synthetic model allows us to illustrate



(a) Number of Trees.

(b) Tree Height.

Figure 2.2: We show that RCFs exhibit the expected random forest behavior. In Figure (2.2a) we see improved out-of-sample improvement as we increase the number of trees, with an eventual plateau. In Figure (2.2b), we see improved out-of-sample improvement with increase tree hights, followed by expected overfitting behavior.

the red-bus-blue-bus problem. The generative model now becomes

$$U_{hj} = 10\sin(\pi X_{1,hj} X_{2,hj}) + 20(X_{3,hj} - 0.5)^2 + 10X_{4,hj} + 5X_{5,hj} + \gamma_h X_{h,\text{bus}_j}$$
$$\gamma_h \sim \text{N}(0, 10)$$

where X_{h,bus_j} is equal to 1 if item j is a bus, and 0 otherwise. In this experiment, we follow our approach outlined in Section (2.5.2) by first fitting a Random Choice Forest, and then performing posterior inference to estimate the posterior of the random effects distribution. We compare this method with a mixed multinomial logit model.

After fitting the Random Choice Forest, we assume the following generative model and perform posterior inference with the probabilistic programming language `stan`.

$$\begin{aligned}
\mu &\sim N(0, 1) \\
\sigma &\sim \text{half-cauchy}(0, 5) \\
\gamma_h &\sim \text{N}(\mu, \sigma^2) \\
U_{hj} &= \beta \times g(X_{hj}) + \gamma_h X_{h,\text{bus}_j} \\
Y_{hj} &\sim \text{Mult}(\exp(U_{hj} / \sum_k U_{hk}))
\end{aligned}$$

where g is fixed and β is treated as a fixed effect. These hyperprior distributions are "standard" distributions suggested by the `stan` manual for such problems. Of course, in real-world settings, more care would be taken when specifying these priors. The `stan` framework performs posterior inference using Hamiltonian Monte Carlo (HMC). In our experiments, we found that using a non-centered parameterization of the priors was crucial for Monte Carlo chain convergence; for details, see Betancourt and Girolami (2013).

Out of sample analysis is done using the posterior predictive choice probabilities. We use 5,000 in-sample agents and 5,000 out-of-sample agents. We repeat the process 10 times.

Using the best model from the fixed effects simulation above, Table (2.2) shows that the RCF with the random effect outperforms the MML.

method	average oos log-likelihood (se)
MML	-4101 (10)
RCF	-3899 (14)

Table 2.2: Held-out log-likelihood. (Mixed Effects)

Next we introduce a new alternative to the choice set – a red bus – and predict how aggregate demand will change. We compute the resulting decrease in demand for each of the original items: blue bus, train, and car after the red bus is added to the choice set. This experiment is done out-of-sample.

For each agent h, we first compute the probability of choosing item j under the estimated model, once for each choice set. In `stan`, this is computed using the posterior predictive distribution. Then, for each item, we average the choice probabilities over the population of agents to estimate the population demand for that item. If p_j^r is the population demand for item j when the red bus is in the choice set, and p_j^{nr} is the population demand for item j without the red bus, the decrease in demand for item j is computed as:

$$\frac{p_j^{nr} - p_j^r}{p_j^{nr}}$$

We compare a purely fixed effects RCF to a mixed effects RCF in Table (2.3) and find that the addition of the random effect produces the desired result. The fixed effects model draws approximately the same demand away from each mode of transportation; but more demand is pulled away from the blue bus when using the random effect. The magnitude of this change can be controlled by incorporating domain knowledge into the choice of the prior distributions.

Finally, we mention a few methodological pieces of advice for the practitioner. In our experiments, we find that with smaller datasets, it is best to split the dataset in half for the two stages of estimation. The first half is used to estimate the RCF, and the second half

model	blue bus	train	car
RCF fixed	29%	29%	31%
RCF mixed	38%	15%	12%

Table 2.3: Percent decrease in demand after addition of red bus.

is used for posterior inference. The improvement in out-of-sample performance disappears as the dataset size increases. Additionally, for massive datasets, it is best to fit a new coefficient on each tree in the second stage of estimation. Recall that a forest is a sum (or average) of trees so that $g(X_{hj}) = \sum_{\text{trees}} g_t(X_{hj})$; in some cases, with enough data, it is better to perform posterior inference under a generative model that assumes $U_{hj} = \sum_{\text{trees}} \beta_t g_t(X_{hj}) + \gamma_h X_{h,\text{bus}_j}$, where each β_t is a fixed effect.

2.6.2 Dunnhumby Beverage Dataset

In this section, we compare a multinomial logit model with fixed effects to the Random Choice Forest on a dataset of supermarket transactions. The dataset is *The Complete Journey* dataset published by Dunnhumby[3] and consists of supermarket receipts from households collected over two years. Each receipt corresponds to a basket of goods. Demographic information is provided on each household, along with three levels of category information about each commodity: department, commodity description, and sub-commodity description.

The dataset consists of 801 households, 140,339 unique shopping baskets, and 68,253 unique items. The average basket contains 10 unique items. Items are categorized into 308 unique "commodity" levels. Since the data are provided as transactional receipts, we must transform the dataset into one that is appropriate for models of discrete choice. This presents several difficulties which we describe next.

The number of unique items presents computational and statistical challenges so we first filter the data so that we only consider beverages. Our simplifying assumption is that each household that bought a beverage made a choice between different kinds of beverages in

3. https://www.dunnhumby.com/sourcefiles

the choice set provided by the supermarket. We constrain the choice set to the ten most popular beverages across all receipts that fall into one of following commodity descriptions: soft drinks, fluid milk products, refrigerated juices and drinks, and beers. We only consider household receipts that contain a *single* item that falls into one of these categories. This results in 8,473 unique shopping baskets.

The final complication of transforming a transactional dataset into a discrete choice dataset is the lack of item-specific measurements on items not chosen by the household. For example, if the household chose milk, the receipt will not contain item-specific information about the other 9 beverages we consider in the choice set. The two item-specific covariates are the price of the item and the coupon value. To impute these missing values in the choice dataset, we smooth a weekly time series of price and coupon values and use the resulting price in each week for all items in the choice dataset, whether or not they were chosen. Using the actual price for the chosen items results in information leakage.

The resulting discrete choice dataset contains ten beverage choices for each household with the guarantee that each household chose only one of the ten beverages in their trip. The agent-specific characteristics include estimated age range, marital status, household income, household composition, household size, and number of children present at time of purchase.

Similar to Section (2.6.1), we use 5-fold cross-validation to compare the MNL model with the RCF on held-out log-likelihood. The specified MNL allows each predictor to enter the model linearly. Initial experimental results suggested that the probabilities from the RCF were miscalibrated. For example, after performing 5-fold cross-validation and choosing the random forest hyperparameters with largest held-out log-likelihood, the range of the resulting predicted choice probabilities were between 0.03 and 0.28. This phenomenon was also observed for a range of different random forest hyperparameters.

To calibrate these probabilities, we use a popular nonparametric calibration method introduced by Zadrozny and Elkan (2001) and successfully applied in Guo et al. (2017) and Niculescu-Mizil and Caruana (2005). The calibration procedure applies a function f to each

raw probability to obtain a calibrated probability. f is estimated with a validation set and the resulting calibrated model is applied to a held-out test set. This procedure is applied to each fold in cross-validation. The method begins with raw estimated probabilities $\widehat{p}_{hj}$ for each agent h and item j. The calibration is applied to each item separately, followed by a normalization step.

Given H agents in the validation set, we take the raw probabilities across all agents for a specific item k $\{\widehat{p}_{1k}, \widehat{p}_{2k}, \ldots, \widehat{p}_{Hk}\}$, order them, and assign them to M bins so that an equal number of raw probabilities are assigned to each bin. For each bin, we also compute $\bar{y}_i$, the fraction of agents in that bin that chose item k. This results in the sequence $\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_M$. The final step of the calibration method performs an isotonic regression on this sequence. The resulting estimated function is then applied to the raw probabilities obtained from the test set. For each agent in the test set, we obtain calibrated probabilities for each item, but these probabilities do not sum to one. The final step is to normalize these K calibrated probabilities as done in Guo et al. (2017).

method	avg held-out log-likelihood
MNL	-3231
RCF	-3392
calibrated RCF	-3253

Table 2.4: Average held-out log-likelihoods across 5 folds.

We present the results of the best performing RCF across all parameters in Table (2.4). We find that while the calibration step improves the performance of the RCF, it still falls short of the MNL performance. One possibility is that this dataset is not rich enough for a random forest because of the signal-to-noise ratio. This hypothesis is supported by the fact that the best performing tree-height in terms of out-of-sample accuracy was a tree height of 2. We performed two additional experiments to test this hypothesis.

In the first experiment, we removed the item-specific attributes – price and coupon value – and fit a standard random forest for classification on the resulting dataset. The task was to classify households into one of ten categories, one category for each beverage. We computed

the out-of-sample accuracy for varying tree heights and found that the best performing tree height was 2, mirroring our observations with the Random Choice Forest. This test suggests that the underperformance is not due to an implementation shortfall of the methodology but rather a feature of the dataset.

Our second experiment explored whether a more sophisticated linear model would outperform the naive MNL where each predictor enters the utility linearly. Much like the first experiment, this approach was yet another search for signal in the data. In our model specification, we included all first and second order interactions between predictor variables. While we found that several interaction coefficients were estimated to be non-zero, the out-of-sample performance of this model did not outperform the simple MNL with only first order interactions. This strongly suggests that this dataset is not rich enough to benefit from more sophisticated modeling techniques.

2.6.3 Cars Dataset

In this section, we apply the RCF and the calibration method discussed in Section (2.6.2) to the *Cars Dataset* used in McFadden and Train (2000) and compare it to the MNL. The dataset was obtained from a study with 4,654 respondents, each of whom was asked to choose among six alternative vehicle options. The alternatives were described based on variables shown in Table (2.5).

The results in Table (2.6) reflect results simlar to those in Section (2.6.2) – the held-out log-likelihood falls short of outperforming the MNL.

2.7 Discussion

We presented new methodology for estimating random forests for discrete choice problems. Our methodology is paired with a computationally efficient algorithm for growing each of the decisions trees. We also proposed an approach for incorporating random effects in order

Variables
Price
Mileage Range
Acceleration
Top Speed
Pollution
Size of Vehicle
Luggage Space
Operating Cost
Station availability
Sport utility vehicle
Sports Car
Truck
Van
Electric Vehicle
College Eduction
Compressed Natural Gas Vehicle
Methanol Vehicle

Table 2.5: Predictor variables for Cars Dataset

method	avg held-out log-likelihood
MNL	-949
calibrated RCF	-960

Table 2.6: Average held-out log-likelihoods across 5 folds.

to avoid the IIA property inherent in fixed effects models and illustrated how they can be useful for counterfactual analysis.

We applied this method to a synthetic dataset and showed that it can outperform the standard multinomial logit model. Past work has also explored fitting multinomial logit models with constrained coefficients (Revelt and Train, 2001) so that, for example, the estimated coefficient for "price" is negative, implying worse utility for higher prices. We could apply an analogous constraint to satisfy this prior assumption by using the methodology described in the first chapter of this thesis for reshaping decision trees.

The synthetic experiments suggest that if the standard discrete choice model is well-specified, then our method will outperform the multinomial logit model in terms of predictive accuracy. However, the Random Choice Forest failed to outperform the multinomial logit

model on real datasets. We discuss several theories for this behavior that can be further explored in future work.

One possibility is that these datasets simply do not have a rich enough structure to be appropriate for nonparametric methods like random forests. We explored this hypothesis in Section (2.6.2) by showing how both a standard random forest for classification and a more sophisticated linear model both fail to outperform the simplest linear model in out-of-sample accuracy. The dearth of large-scale open-source discrete choice datasets makes proving this methodology difficult. Some of the most interesting large-scale datasets in this area are owned by companies that will never release such data. However, future work could explore additional regularizing procedures to improve the accuracy of the random forest; for example, adding a constraint to the sum of all leaf values.

A second possibility is that the random utility approach to modeling choice, together with utility maximization, is a poor model of human behavior. Utility maximization provides just one approach to modeling choices among a set of alternatives, but it comes with many assumptions, some of which refuted experimentally. If the discrete choice model inspired by random utility is a poor representation of actual choices, then we would not expect to see the relaxation of a linear utility function improving out-of-sample prediction. But, perhaps a happy medium can be attained by borrowing some of the ideas of the Random Choice Forest, and combining them with the simple linear model approach. We describe one possibility next.

The method proposed above resulted in optimal leaf values being tied to the values of other leaves in the tree. This complication arises from splitting based on item-specific attributes because it allows for the choice set of an individual to trickle down to different leaves in the tree, tying the leaves together because of the form of discrete choice likelihood.

An alternative approach is to only consider splits of agent-specific attributes and, instead of fitting a constant function in each leaf, fitting a linear multinomial logit model that depends on item-specific attributes. For each potential split option, two multinomial logit

models would be estimated in the leaves. Each agent in the choice set would be mapped to a specific leaf so that the function b that maps data to leaves would only depend on the agent h. Each leaf would be associated with an estimated coefficient vector so that the log-likelihood would now be,

$$\ell(V_L, V_R, \widetilde{V}) = \sum_{h \in b^\star} \sum_{j=1}^{J} Y_{hj} \left(\beta_{b(h)}^\top X_{hj} - \log \left(\sum_{\ell=1}^{J} \exp(\beta_{b(h)}^\top X_{h\ell}) \right) \right)$$

The splits would still implicitly depend on item-specific characteristics since the score for each split would depend on the model in each leaf, and the model in each leaf depends on the item-specific attributes. This approach offers a compromise between the multinomial logit and the random forest, and would be interesting to explore in future work.

CHAPTER 3

TRANSLATION EMBEDDINGS

3.1 Abstract

We propose a method for constructing dense vector representations of objects like words, sentences, and mathematical equations. Our approach blends ideas from well-known embedding methods like `word2vec` (Mikolov et al., 2013) and `GloVe` (Pennington et al., 2014) with a generative model inspired by work in statistical machine translation (Brown et al., 1993). The parameters of the generative model are estimated using the expectation-maximization algorithm, and the final embeddings are obtained by projecting the resulting parameter estimates to a lower-dimensional space. Our approach allows the practitioner to capture the different kinds of similarity based on the specification of the probabilistic model. Unlike standard methods for word embeddings, this approach can also be applied to variable-length objects. We use this approach to construct word embeddings and our experiments show that this method obtains results competitive with `GloVe` on multiple word similarity tasks. Additionally, we apply this method to mathematical texts to obtain embeddings for mathematical equations. We formulate qualitative and quantitative tasks for assessing the validity of these equation embeddings and show that this approach yields reasonable results.

3.2 Introduction

This work is part of a larger effort called the Hopper Project which aims is to develop natural language processing tools for mathematical texts. One goal of the Hopper Project is to develop algorithms and methods to extract representations of text and mathematical equations to improve downstream tasks such as document similarity and search over corpora of technical documents. Leveraging the mathematical equations in addition to natural language will enable a rich, unified view of each document, allowing comparisons across subject areas that

use different language to discuss similar mathematical concepts.

This work tackles the problem of comparing equations as a first step towards these goals. The objective of this project is to construct vector representations (i.e. embeddings) of equations so that two nearby vectors correspond to two equations with similar syntax or meaning. Embedding methods have become popular in natural language processing. Word embeddings are feature engineering tools that construct mappings from words to real-valued vectors. These methods transform words from a one-hot vector representation to a dense, continuous vector such that nearby words in the resulting vector space have similar meaning (Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014). The resulting word embeddings can then be used in downstream NLP tasks and have been shown to improve tasks such as parsing (Socher et al., 2013a) and sentiment analysis (Socher et al., 2013b).

In this chapter, we propose a general framework for constructing embeddings for objects. Unlike most word embedding methods, this method can also be applied to variable-length objects like sentences or equations. The framework requires the practitioner to specify a generative model of the objects of interest; careful attention must be given to the specification so that appropriate similarities are captured. The benefit of this approach is that the practitioner has control over the kind of similarity captured by the embeddings, providing flexibility for different kinds of downstream tasks.

The method presented here was inspired by several word embedding methods. One such class of word embedding methods specify a language model for assigning a probability to a sequence of tokens. Mathematically, a language model assigns a probability to a sequence of n words,

$$P(w_1, w_2, \ldots, w_n)$$

A good language model assigns high probability to sequences of words that are valid (i.e. syntatically and semantically correct sentences), and a low probability to random sequences of words. The most popular word embedding method, `word2vec`, paramaterizes a language model with word embeddings, and estimates these word embeddings using natural language

text.

Instead of fitting a language model, we borrow ideas from statistical machine translation (Brown et al., 1993) and propose modeling object generation in terms of "translation probabilities" – the probability of one object given another. These translation probabilities are then used to represent objects in the collection. Translation models are motivated as follows. Assume we are translating a French sentence f to an English sentence e. Let $p(e|f)$ be the probability assigned to e being a translation of f. Using Bayes' theorem, we can write $p(e|f) \propto p(e) \times p(f|e)$. Then, given a French sentence f, we find the most likely translation by performing the following optimization:

$$\widehat{e} = \arg\max_{e} \; p(e) \times p(f|e)$$

In the expression above, $p(e)$ is a language model as described above and $p(f|e)$ is the translation model. In our work, instead of focusing on the language model like `word2vec`, we focus on the translation model.

3.3 Related Work

Embeddings are dense vector representations of objects. They are typically used as feature inputs to machine learning methods for various natural language processing tasks and are an example of a successful application of unsupervised learning. Given a vocabulary of V words, the simplest word embeddings would represent each word as a $\{0,1\}^V$ one-hot vector.

For example,

$$v_{\text{chicago}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \qquad v_{\text{philadelphia}} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

However, this word representation does not give us any notion of similarity. For example, $v_{\text{chicago}}^\top v_{\text{philadelphia}} = 0$, despite Chicago and Philadelphia both being major cities.

More sophisticated word embeddings were popularized by the famous `word2vec` method (Mikolov et al., 2013). The `word2vec` method offers two approaches for constructing word embeddings. One approach, the continuous bag-of-words (CBOW) model, uses the context of each word (surrounding words) to predict the left-out center word. For example, in the sentence "Chicago gets cold in the winter", we would treat the words (chicago, gets, in, the, winter) as the context and, from these words, a model is estimated to predict the center word "cold". These (context, word) training examples can be constructed from a corpus of text. The model is parameterized with word embeddings and estimated by minimizing an objective function.

The data consist of (context, word) pairs obtained from a corpus of text. Each word is represented as a one-hot V-dimensional vector, where V is the number of words in the vocabulary. Let $x^{(c)}$ denote a one-hot vector representing a word in the context and let y denote the center word. In the CBOW `word2vec` method, the $x^{(c)}$ and y can be considered our data; let $V \in \mathbb{R}^{n \times V}$ and $U \in \mathbb{R}^{V \times n}$ denote our parameters, where n is the size of our embedding space. For a given context of size $2m$, we obtain the one-hot word vectors $(x^{(c-m)}, \ldots, x^{(c-1)}, x^{(c+1)}, \ldots, x^{(c+m)})$. The corresponding embeddings vectors are then $(Vx^{(c-m)}, \ldots, Vx^{(c-1)}, Vx^{(c+1)}, \ldots, Vx^{(c+m)})$. The CBOW method then averages these vectors to obtain $\widehat{v}$, which represents the context. Let u_j denote the row in U that represents

the center word y. A score z is then generated so that $z = u_j^\top \widehat{v}$; the goal is to generate a score that is higher for similar words. The score is then transformed into a probability $\widehat{y} = \text{softmax}(z)$ and compared to the true center word y using a cross-entropy loss function. The loss function can be viewed as the pseudo-likelihood of a language model. The cross-entropy is maximized as a function of the U and V to obtain the word embedding vectors. This method learns two vectors for each word in the vocabulary – a word embedding and a context embedding; typically the word embeddings are used in downstream tasks. An alternative to the CBOW model is the skip-gram model which conditions on the center word and predicts (or generates) the surrounding words.

`GloVe` (Pennington et al., 2014), another popular word embedding method, takes a different approach. Instead of using local context information like `word2vec`, `GloVe` is trained on global word-word co-occurence counts. This method models the ratios of cooccurence probabilities. Let p_{ij} denote the probability that word i and word j cooccur within a certain window size. `GloVe` aims to model the ratio p_{ij}/p_{jk} in terms of the word embeddings for words i, j, and k. The derivation of the model results in an objective function that fits embeddings vectors for words i and j such that their inner product $v_i^\top v_j$ approximates the log coccurence count between i and j, $\log(X_{ij})$. Adding additional bias terms b_i and b_j, the objective function becomes the following weighted lease squares model:

$$\sum w_{ij} \left(v_i^\top v_j + b_i + b_j - \log X_{ij} \right)^2$$

Other methods for learning word embeddings leverage matrix factorization methods. These methods utilize low-rank approximations of large matrices that contain word statistics computed from a corpus of text. One such factorization method is latent semantic analysis (LSA) (Deerwester et al., 1990). The first step in LSA is to construct a term-document matrix where the rows correspond to words and the columns correspond to documents. Each element of the matrix represents the number of times a particular word appears in a particular

document. A more recent method (Levy and Goldberg, 2014) factorizes a term-term matrix M where each element is the pointwise mutual information between the two corresponding words. The resulting matrix is factorized with a k-SVD so that $M = U_k \Sigma_k V_k^\top$ where the subscript k corresponds to the first k singular vectors or values. The word embeddings are then taken to be $W = U_k \Sigma_k^{1/2}$. Levy and Goldberg (2014) show that, under certain assumptions, this approach is equivalent to the skip-gram `word2vec` method described above. The method presented in this chapter applies matrix factorization to a matrix of translation probabilities as described in the sections that follow.

The word embedding methods described so far are limited in that they are not easily extended to variable-length objects like sentences. Several approaches to sentence embeddings have been proposed to capture the semantics of word sequences. One of the earliest methods in this space is the Paragraph Vector method (Le and Mikolov, 2014). The Paragraph Vector method is similar in spirit to `word2vec` except that it ties together several (context,word) pairs with an additional synthetic token representing the sentence (or paragraph) each pair came from. In addition to learning word embeddings to predict the held-out word, this method also learns a vector corresponding to the token for each sentence. While the results in this paper were encouraging, other research groups were unable to reproduce their results[1].

Since the Paragraph Vector method, recent work has applied methods ranging from simple averaging of word embeddings to complex convolutional or recurrent neural networks (Iyyer et al., 2015; Kiros et al., 2015; Wang et al., 2016). More recent work (Wieting et al., 2016), learned sentence embeddings by starting with word embeddings and modifying them in a supervised approach using a paraphrase pairs dataset and training a model that averages word embeddings. This method led to strong performance on several standard text similarity tasks. However, it relies on supervision which is not widely available, especially in the context of mathematical equations.

1. `https://groups.google.com/d/msg/word2vec-toolkit/Q49FIrNOQRo/4Z_oDZMyQvUJ`

The goal of this project was to construct embeddings with variable-length objects without appealing to black box deep learning methodology. Our proposed probabilistic approach to constructing embeddings allows the practitioner to target the appropriate kind of similarity needed for his or her task. Drawing inspiration from `word2vec` and matrix factorization approaches we described above, we leverage parameter estimates from a translation model to construct embeddings for words and sentences. The translation model is inspired by the IBM translation models described in the next section.

3.4 IBM Translation Models

The probabilistic models used in our embedding method were inspired by the seminal statistical machine translation systems developed at IBM in the late 1980s (Brown et al., 1993). In this section, we describe the simplest version of these models to illustrate the assumptions used in the following sections. The convention is to assume that the task is to translate from French, the source language, into English, the target language.

We will use f to refer to a French sentence so that f is a sequence of words $f_1, f_2, \ldots, f_m$, where m is the length of the French sentence. Similarly, e denotes the English sentence with length l so that $e = e_1, e_2, \ldots, e_l$. We assume that we have training data in the form of "parallel text" containing example translations $(f^{(k)}, e^{(k)})$ for $k \in [n]$, where $e^{(k)}$ is a translation of $f^{(k)}$. We use this data to estimate the parameters of our translation model.

As described in Section (3.2), the IBM models have two components: a language model $p(e)$ that assigns a probability to a sequence of English words, and a translation model $p(f|e)$ that assigns a conditional probability to pairs of French and English sentences. Together, these models are used to translate a French sentence by performing the optimization:

$$\widehat{e} = \arg\max_{e} \; p(e) \times p(f|e)$$

The embedding method in this chapter focuses on estimating the translation component

of the right-hand side. Since it is difficult to model $p(f_1, f_2, \ldots, f_m|e_1, e_2, \ldots, e_l)$ directly, a key idea in the IBM models is to introduce additional *alignment* variables $a_1, \ldots, a_m$. The alignment variable $a_i \in \{0, 1, 2, \ldots, l\}$ specifies the English word e_{a_i} that was translates to (or generates) the French word f_i. We assume e_0 is a special `NULL` token that indicates no direct translation of the corresponding French word. With the alignment variables defined, we can write the translation model as

$$p(f_1, f_2 \ldots, f_m|e_1, e_2, \ldots, e_l) = \sum_{a_1} \sum_{a_2} \cdots \sum_{a_m} p(f_1, f_2 \ldots, f_m, a_1, \ldots, a_m|e_1, e_2, \ldots, e_l)$$

Let f_1^m denote $f_1, f_2 \ldots, f_m$, then we have:

$$\begin{aligned} p(f_1, f_2 \ldots, f_m|e_1, e_2, \ldots, e_l) &= \sum_{a_1} \sum_{a_2} \cdots \sum_{a_m} p(f_1^m, a_1^m|e_1^l) \\ &= \sum_{a_1} \sum_{a_2} \cdots \sum_{a_m} p(a_1^m|e_1^l, m) \times p(f_1^m, |a_1^m, e_1^l) \\ &= \sum_{a_1} \sum_{a_2} \cdots \sum_{a_m} \prod_i p(a_i|a_1^{i-1}, e_1^l, m) \times p(f_i|a_1^i, f_1^{i-1}, e_1^l) \end{aligned}$$

where each step here is equality, applying the definition of conditional probability and the chain rule. The last step is one of many ways to write the probability. We can interpret this last expression as a generative model. We generate a French sentence from an English sentence by first choosing which position of the English sentence generates the first word in the French sentence, a_1, given the English sentence. Then we choose the actual French word given the knowledge of the English string and the position to which the first French word is connected by the alignment variable. This idea can be applied to each word in the French sentence, fully describing a generative model for the French sentence.

Two simplifying independence assumptions are made in IBM Model 1 that result in a computationally efficient estimation method. The first simplifying assumption is that the distribution of a_i depends only on the lengths of the French and English sentences. The

second assumption is that the value f_i depends only on e_{a_i} – the identify of the English word to which f_i is aligned. These assumptions lead to

$$\begin{aligned} p(f_1, f_2 \ldots, f_m | e_1, e_2, \ldots, e_l) &= \sum_{a_1} \sum_{a_2} \cdots \sum_{a_m} \prod_i p(a_i | l, m) p(f_i | e_{a_i}) \\ &= \frac{1}{(l+1)^m} \sum_{a_1} \sum_{a_2} \cdots \sum_{a_m} \prod_i p(f_i | e_{a_i}) \end{aligned}$$

Given a set of training examples, and with the above model defined, parameter estimation boils down to estimating the translation probabilities $p(f_i|e_j)$ — the probability of e_j being translated to (or generating) f_i. We apply this framework in the following sections to estimate "translation probabilities" and construct word and equation embeddings.

3.5 Translation Embeddings

In this section, we describe our translation embeddings method which leverages ideas from the IBM translation models described in Section (3.4). The word "translation" is used loosely here and stems from the relation to these IBM models and not because we are attempting to translate from one language to another.

3.5.1 Translation Probabilities

Assume we want to construct an embedding for object $e = \{e_1, e_2, \ldots, e_\ell\}$. If e is an individual unit (e.g. a word), then $\ell = 1$. Drawing inspiration from IBM Model 1 (Brown et al., 1993), the probability of target unit f_i conditioned on object e is given by

$$p(f_i|e) = \frac{1}{\ell + 1} \sum_{j=0}^{\ell} p(f_i|e_j) \tag{3.5.1}$$

where the f_i, e_i are individual units like words or mathematical symbols. The motivation for Equation (3.5.1) will be described in the sections that follow. For now, we will assume that this generative model is given.

Under these assumptions, the probabilistic model associated with object e is characterized by the V-vector $v_e = (v_1(e), v_2(e), \ldots, v_V(e)) \in \mathbb{R}^V$ with components $v_i(e) = p(f_i|e)$, where V is the size of the "vocabulary" and f_i denotes the i-th word or symbol in the vocabulary.

The translation probabilities for the units $p(f_i|e_j)$ can be estimated given some form of parallel text and additional assumptions on the data generating process. We show some specific examples in the experiments. The translation probability vectors $\{v_e\}$ are then used to fit embeddings.

3.5.2 Translation Probabilities → Embeddings

Given the vector of translation probabilities for object e, $v_e \in \mathbb{R}^V$, we fit embedding vectors $\phi_e \in \mathbb{R}^m$ where $m \ll V$. We could appeal to the Johnston-Lindenstrauss lemma and apply a random projection to v_e, but our experiments show that the following approach yields better results.

We construct a dictionary $D \in \mathbb{R}^{V \times m}$ and representations $\phi_e \in \mathbb{R}^m$ to minimize the loss function

$$\mathcal{L}(D, \phi) = \sum_e \|v_e - D\phi_e\|^2 = \|V - D\Phi\|_F^2$$

subject to $D^\intercal D = I$, where $V \in \mathbb{R}^{V \times N}$ is the matrix whose columns are the translation probability vectors v_e and the columns of $\Phi \in \mathbb{R}^{m \times N}$ are the fitted embeddings.

The solution to this minimization can be found by taking the SVD of $V = \underbrace{U}_{D} \underbrace{\Sigma V^\intercal}_{\Phi}$ and truncating to the top m singular vectors and values. An embedding vector $\phi_e = U^\intercal v_e$ can be calculated for a new object e, assuming we have the vector of translation probabilities v_e.

3.6 Word Embeddings

Our method requires the practitioner to specify a generative model parameterized by translation probabilities – the probability of generating one object given another. In the next section, we propose a simple generative model for text and show how it can be used to construct word embeddings.

Model

Let $\mathcal{C} = w_1 w_2 w_3 \dots w_N$ denote a corpus of N words. We propose a model parameterized by "translation probabilities" $p(w_i|w_j)$, the probability of word w_j generating word w_i, and apply the embedding method described above.

Assume the t-th word in the corpus w_t is generated by one of the k words in its immediate history $\{w_{t-k}, \dots, w_{t-1}\}$. That is, w_t is generated by w_{a_t} where a_t is a latent variable and $a_t \in \{t-k, \dots, t-1\}$.

Denote the history as $w_{t-k}^{t-1} = \{w_{t-k}, \dots, w_{t-1}\}$. Under the model, we have

$$\begin{aligned} p(w_t|w_{t-k}^{t-1}) &= \sum_{a_t=t-k}^{t-1} p(w_t|a_t, w_{t-k}^{t-1}) p(a_t|w_{t-k}^{t-1}) \\ &= \sum_{a_t=t-k}^{t-1} p(w_t|w_{a_t}) p(a_t|k) \end{aligned}$$

where in the last step we have employed independence assumptions similar to those found in IBM Models 1 and 2 described in Section (3.4). Assuming each word in the history generates w_t with equal probability, we have

$$p(w_t|w_{t-k}^{t-1}) = \frac{1}{k} \sum_{j=t-k}^{t-1} p(w_t|w_j)$$

Given the the corpus, the log-likelihood under this model is

$$\sum_{t=k+1}^{N} \log \left(\sum_{j=t-k}^{t-1} p(w_t|w_j) \right)$$

We maximize the likelihood with EM and use the resulting translation probabilities to construct word embeddings.

3.6.1 EM for Words

We first initialize the translation probabilities $p(w_i|w_j) = \frac{1}{V}$ for all i, j, where V is the number of words in the vocabulary.

E-step: Compute the expected counts $\mathbb{E}[c(w'|w)]$, the expected number of times word w generates word w' under the distribution $p(a_t|w_{t-k}^t)$. Note that when the parameters $p(w'|w)$ are known, we have

$$p(a_t|w_{t-k}^t) = \frac{p(a_t, w_t|w_{t-1}^{t-k})}{\sum_{a_t} p(a_t, w_t|w_{t-1}^{t-k})} = \frac{p(w_t|w_{a_t})}{\sum_{a_t} p(w_t|w_{a_t})}$$

For a given sequence of words w_{t-k}^t, the evidence that word w generates word w' is

$$\begin{aligned} c(w'|w; w_{t-k}^t) &= \sum_{a_t=t-k}^{t-1} p(a_t|w_{t-k}^t)\delta(w', w_t)\delta(w, w_{a_t}) \\ &= \frac{p(w'|w)}{\sum_{j=t-k}^{t-1} p(w'|w_r)}\delta(w', w_t) \sum_{r=t-k}^{t-1} \delta(w, w_r) \end{aligned}$$

where $\delta(i, j) = 1$ if $i = j$, and 0 otherwise. We compute these counts over the entire corpus.

M-step: Compute the MLEs of each $p(w_i|w_j)$ with the expected counts above:

$$p(w'|w) = \frac{c(w'|w)}{\sum_{w_i} c(w_i|w)}$$

3.6.2 Wikipedia Experiments

We use a 700MB snippet of Wikipedia for conducting experiments with word embeddings. Words that occur less than 400 times are removed, yielding a vocabulary of approximately $15,000$ words.

Translation probabilities are estimated by the EM algorithm described above and projected to 100 dimensions by taking the SVD, as described in Section (3.5.2).

Nearest Neighbor Examples

We first do a qualitative study of our word embeddings. For each word in a set of query words, we find the 5 nearest neighbors in the embedding space and report the results in Table (3.6.1). We obtain reasonable results for this nearest neighbors exercise, with the nearest neighbor words being similar in topic or meaning.

republican	physics	baseball	fish
federalist	theoretical	hockey	freshwater
democrat	mechanics	soccer	aquarium
senator	particle	basketball	prey
libertarian	relativity	player	feeding

Table 3.1: Nearest neighbor examples.

The examples shown in Table (3.6.1) illustrate the kinds of similarity we are capturing with these embeddings. And we can describe why this kind of similarity is captured by appealing to the specification of our generative model.

The first kind of similarity is between words that appear with similar words in their future (words that appear after the word of interest). For example, the words "baseball" and "hockey" appear with similar sports-related terms in their future context. Since the generative model uses past words to predict future words, the model associated with baseball, $p(\cdot|\text{baseball})$ will be similar to the model associated with hockey, $p(\cdot|\text{hockey})$.

The second kind of similarity is between words that co-occur with each other; for example the words "theoretical" and "physics", and also the words "freshwater" and "fish". Again,

since these words co-occur with each other, their future contexts will be similar across the corpus, yielding similar models. A different specification of the model could reduce one kind of similarity over another as needed by the practitioner.

Word Similarity

Word similarity tasks are a standard quantitative approach for evaluating word embeddings. The assumption is that word embedding methods and humans should agree on what words are "similar." For each word similarity task, human subjects were given a list of word pairs and asked to rate the similarity of the words in each pair on a $1 - 10$ scale. Scores were averaged across subjects for each word pair. We compare similarity scores based on our word embeddings to these human scores on four word similarity tasks: WordSim-353 (WS) (Finkelstein et al., 2001), MC (Miller and Charles, 1991), SCWS (Huang et al., 2012), and RG (Rubenstein and Goodenough, 1965).

The Hellinger distance is a common metric used to compare two probability distributions. The Hellinger distance between distributions p and q is $\sum_i (\sqrt{p_i} - \sqrt{q_i})^2$. In these tasks, we first take the square root of our translation probabilities and then project to 100-dimensional space. Intuitively, the square root is applied to up-weight lower probabilities before projection.

Cosine similarity scores are computed using the embedding vectors for each pair of words. We then compute the Spearman rank correlation between embedding similarity scores and the human similarity scores across the entire collection of words.

We compare our embeddings to `GloVe` embeddings fit on the same Wikipedia dataset in Table (3.2). The **tr-embed** row refers to our method. We find that the translation embedding approach yields competitive results to the popular `GloVe` method on this task.

	WS	MC	SWCS	RG
GloVe	0.70	0.69	0.64	0.70
tr-embed	0.72	0.80	0.62	0.80

Table 3.2: Word similarity correlation across four different tasks.

3.7 Equation Embeddings

We construct equation embeddings using the Digital Library of Mathematical Functions (DLMF) (Olver et al.) as our corpus. Equations are tokenized at the mathematical symbol level and then represented as a bag of symbols. In the sections that follow, we specify our generative model for the DLMF, derive the EM steps, and show qualitative and quantitative results. Broadly speaking, there are two kinds of similarity we could measure between mathematical equations: syntactic and semantic similarity. The DLMF allows us to construct embeddings that model syntactic similarity – we start here as a proving grounds for the methodology. In Section (3.8) we propose a different approach for future work that aims at modeling semantic similarity.

Model

The DLMF contains sections of mathematical text, each containing several equations. Each section is about a specific mathematical topic so we make the assumption that the equations in each section are similar. This approach could also be applied to scientific publications, where we assume equations in the same paper are similar.

We impose a two-level hidden alignment model similar to our one-level hidden alignment model for words. We assume each equation in a section is the translation of some other "source" equation in that same section. The identity of the source equation is latent. The second level of hidden alignment is at the character level, where each character in the target equation is assumed to be a translation of a token in the source equation.

We use subscripts to identify individual symbols in an equation so that the r-th symbol of the i-th equation in section s is $e_r^{s,i}$. When it is unnecessary, we drop the section superscript

so that we simply have e^i_r. Assume that following notational conventions:

- Each section s consists of m_s equations $\{e^{s,1}, \ldots, e^{s,m_s}\}$.
- The i-th equation in section s is represented as a bag of $n_{s,i}$ symbols $e^{s,i} = \{e^{s,i}_1, \ldots, e^{s,i}_{n_{s,i}}\}$.

We assume that each equation in a section is generated by (translated from) some other equation in the same section. The pseudo-likelihood of section s with equations $\{e^{s,1}, \ldots, e^{s,m_s}\}$ is

$$p(s) = p(e^{s,1}, \ldots, e^{s,m_s}) = \prod_{i=1}^{m_s} p(e^{s,i}|e^{s,-i})$$

Each alignment at the section level is assumed to have equal probability. Marginalizing over the latent variable representing the section-level alignment yields

$$p(e^{s,i}|e^{s,-i}) = \frac{1}{m_s - 1} \sum_{j \neq i} p(e^{s,i}|e^{s,j})$$

Given two equations in the same section with the following symbols: $e^i = \{e^i_1, \ldots e^i_{n_i}\}$ and $e^j = \{e^j_1, \ldots, e^j_{n_j}\}$, we impose a second level of hidden alignment at the symbol level (including a NULL symbol), which yields

$$p(e^i|e^j) = \frac{1}{(n_j + 1)^{n_i}} \prod_{k=1}^{n_i} \sum_{r=1}^{n_j} p(e^i_k|e^j_r)$$

We aim to estimate the parameters

$$\Theta = \{p(e_i|e_j) : \text{for } e_i, e_j \text{ in our symbol vocabulary}\}$$

by maximum likelihood.

Given data $\mathcal{D}$, the incomplete-data log-likelihood is

$$\ell(\Theta|\mathcal{D}) = \sum_s \sum_{i=1}^{m_s} \log \left(\frac{1}{m_s - 1} \sum_{j \neq i} p(e^{s,i}|e^{s,j}) \right)$$

Plugging in the equation-level translation model gives us our final expression

$$\ell(\Theta|\mathcal{D}) = \sum_s \sum_{i=1}^{m_s} \log \left(\sum_{j \neq i} \left(\frac{1}{(n_j+1)^{n_i}} \prod_{k=1}^{n_i} \sum_{r=1}^{n_j} p(e_k^{s,i}|e_r^{s,j}) \right) \right)$$

We estimate the parameters Θ using EM.

3.7.1 EM for Equations

In this section, we focus on maximizing $\ell(\Theta|\mathcal{D}) = p(e^i|e^{-i})$ as a function of symbol-level translation probabilities $p(e_r|e_q)$. We inspect the variation in the likelihood estimate of updating our parameter estimate from Θ' to Θ:

$$\begin{aligned}
\ell(\Theta|\mathcal{D}) - \ell(\Theta'|\mathcal{D}) &= \log \left(\frac{p_\Theta(e^i|e^{-i})}{p_{\Theta'}(e^i|e^{-i})} \right) \\
&= \log \left(\sum_{a,\widetilde{a}} \frac{p_\Theta(e^i, a, \widetilde{a}|e^{-i})}{p_{\Theta'}(e^i|e^{-i})} \right) \\
&= \log \left(\sum_{a,\widetilde{a}} \frac{p_\Theta(e^i, a, \widetilde{a}|e^{-i})}{p_{\Theta'}(e^i, a, \widetilde{a}|e^{-i})} p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i}) \right)
\end{aligned}$$

where we sum over equation-level alignments $\widetilde{a}$ and symbol-level alignments a and employ Bayes' rule in the last step. Note that

$$\begin{aligned} p_\Theta(e^i, a, \widetilde{a}|e^{-i}) &= p_\Theta(e^i, a|e^{\widetilde{a}})p(\widetilde{a}|e^{-i}) \\ &= p_\Theta(e^i|e^{\widetilde{a}}, a)\underbrace{p(a|e^{\widetilde{a}})p(\widetilde{a}|e^{-i})}_{\text{constant}} \end{aligned}$$

Plugging this in we get

$$\begin{aligned} \ell(\Theta|\mathcal{D}) - \ell(\Theta'|\mathcal{D}) &= \log\left(\sum_{a,\widetilde{a}} \frac{p_\Theta(e^i|e^{\widetilde{a}}, a)}{p_{\Theta'}(e^i|e^{\widetilde{a}}, a)} p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i})\right) \\ &\geq \sum_{a,\widetilde{a}} \log\left(\frac{p_\Theta(e^i|e^{\widetilde{a}}, a)}{p_{\Theta'}(e^i|e^{\widetilde{a}}, a)}\right) p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i}) \\ &= \underbrace{\sum_{a,\widetilde{a}} \log\Big(p_\Theta(e^i|e^{\widetilde{a}}, a)\Big) p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i})}_{Q(\Theta|\Theta')} - \\ &\quad \underbrace{\sum_{a,\widetilde{a}} \log\Big(p_{\Theta'}(e^i|e^{\widetilde{a}}, a)\Big) p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i})}_{Q(\Theta'|\Theta')} \end{aligned}$$

We ignore $Q(\Theta'|\Theta')$ since it doesn't depend on Θ, and aim to maximize $Q(\Theta|\Theta')$. The symbol-level alignment variable a represents alignments a_k, one for each symbol in the gen-

erated equation e^i.We have that

$$\begin{aligned}
Q(\Theta|\Theta') &= \sum_{\widetilde{a}} \sum_{a} \log\Big(p_{\Theta}(e^i|e^{\widetilde{a}}, a)\Big) p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i}) \\
&= \sum_{\widetilde{a}} \sum_{a} \log\left(\prod_{k=1}^{n} p_{\Theta}(e^i_k|e^{\widetilde{a}}_{a_k})\right) p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i}) \\
&= \sum_{\widetilde{a}} \sum_{a} \sum_{k=1}^{n} \log\Big(p_{\Theta}(e^i_k|e^{\widetilde{a}}_{a_k})\Big) p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i})
\end{aligned}$$

We take the derivative of $Q(\Theta|\Theta')$ with respect to an arbitrary parameter $p(e_r|e_q)$, the probability of symbol e_q generating symbol e_r, and include the Lagrange multiplier λ to ensure probabilities sum to one. The following KKT condition must be met at the optimum:

$$\begin{aligned}
\sum_{\widetilde{a}} \sum_{a} \sum_{k=1}^{n} \frac{p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i})}{p(e_r|e_q)} \delta(e^i_k, e_r)\delta(e^{\widetilde{a}}_{a_k}, e_q) &= \lambda \\
\Rightarrow p(e_r|e_q) = \frac{1}{\lambda} \sum_{\widetilde{a}} \sum_{a} \sum_{k=1}^{n} p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i}) \delta(e^i_k, e_r)\delta(e^{\widetilde{a}}_{a_k}, e_q)
\end{aligned}$$

Given Θ', we can compute $p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i})$:

$$\begin{aligned}
p_{\Theta'}(a, \widetilde{a}|e^i, e^{-i}) &= p_{\Theta'}(a|e^i, e^{\widetilde{a}}) p_{\Theta'}(\widetilde{a}|e^i, e^{-i}) \\
&= \left(\frac{p(e^i|e^{\widetilde{a}}, a)}{\sum_{a'} p(e^i|e^{\widetilde{a}}, a')}\right)\left(\frac{p(e^i|e^{\widetilde{a}})}{\sum_{\widetilde{a}'} p(e^i|e^{\widetilde{a}})}\right) \\
&= \left(\frac{\prod_{k=1}^{n} p(e^i_k|e^{\widetilde{a}}_{a_k})}{\sum_{a'} \prod_{k=1}^{n} p(e^i_k|e^{\widetilde{a}}_{a'_k})}\right)\left(\frac{p(e^i|e^{\widetilde{a}})}{\sum_{\widetilde{a}'} p(e^i|e^{\widetilde{a}})}\right)
\end{aligned}$$

All terms in this expression can be written in terms of the current parameters Θ'. This

gives us

$$p(e_r|e_q) = \frac{1}{\lambda} \sum_{\widetilde{a}} \sum_{a} \sum_{k=1}^{n} \left(\frac{\prod_{k=1}^{n} p(e_k^i|e_{a_k}^{\widetilde{a}})}{\sum_{a'} \prod_{k=1}^{n} p(e_k^i|e_{a'_k}^{\widetilde{a}})} \right) \left(\frac{p(e^i|e^{\widetilde{a}})}{\sum_{\widetilde{a}'} p(e^i|e^{\widetilde{a}})} \right) \delta(e_k^i, e_r) \delta(e_{a_k}^{\widetilde{a}}, e_q)$$
$$= \frac{1}{\lambda} \sum_{\widetilde{a}} \left(\frac{p(e_r|e_q)}{\sum_b p(e_r|e_b^{\widetilde{a}})} \right) \left(\frac{p(e^i|e^{\widetilde{a}})}{\sum_{\widetilde{a}'} p(e^i|e^{\widetilde{a}})} \right) \sum_{k=1}^{n} \delta(e_k^i, e_r) \sum_{j=1} \delta(e_j^{\widetilde{a}}, e_q)$$

where λ is a normalization constant which ensure $p(e_r|e_q)$ is a valid probability distribution.

The E-step therefore consists of computing the expected counts by adding the following term every time symbols e_r and e_q are aligned over section- and equation-level alignments:

$$c(e_r, e_q) += \left(\frac{p(e_r|e_q)}{\sum_b p(e_r|e_b^{\widetilde{a}})} \right) \left(\frac{p(e^i|e^{\widetilde{a}})}{\sum_{\widetilde{a}'} p(e^i|e^{\widetilde{a}})} \right)$$

The M-step normalizes these counts.

We encounter underflow when computing $p(e^i|e^j)$ in the E-step. This is one approach to solving this problem; there may be other (better) solutions.

We compute equation-level translation probabilities in log space.

$$z_{ij} = \log(p(e^i|e^j) = \sum_k \log \left(\sum_r p(e_k^i|e_r^j) \right) - n_i \log(n_j + 1)$$

Then we have that

$$\frac{p(e^i|e^j)}{\sum_k p(e^i|e^k)} = \frac{e^{z_{ij}}}{\sum_k e^{z_{ik}}}$$

We find that the z_{ij} are negative numbers with large magnitude so that we get underflow, meaning $e^{z_{ij}} = 0$. To solve this problem we can multiply the numerator and denominator by e^{-m} where $m = \max_j z_{ij}$. This still results in underflow for values of z_{ij} that are orders of magnitude smaller than m, but we take the stance that these values are negligible compared to the one associated with m, so treating them as zero will give us reasonable results. So we

have,

$$\frac{p(e^i|e^j)}{\sum_k p(e^i|e^k)} = \frac{e^{z_{ij}-m}}{\sum_k e^{z_{ik}-m}}$$

3.7.2 DLMF Experiments

Nearest Neighbor Examples

After estimating the translation probabilities as discussed above, we project each equations' probability vector to 100 dimensions using SVD. In this section, we draw three random query equations and show that their nearest neighbor equation has similar syntax. Since our model does not incorporate surrounding text or mathematical domain knowledge, the method captures similarity based on syntax since syntactically similar equations appear in the same DLMF section. In Section (3.8), we propose a different approach that would potentially capture semantically similar equations.

query equation	nearest neighbor
$\zeta(s) = \frac{2^{s-1}}{1-2^{1-s}} \int_0^\infty \frac{\cos(s \arctan x)}{(1+x^2)^{s/2} \cosh(\frac{1}{2}\pi x)} dx.$	$\zeta(s) = \frac{1}{2(1-2^{-s})\,\Gamma(s)} \int_0^\infty \frac{x^{s-1}}{\sinh x} dx$
$\gamma(a,x) \sim -x^a e^{-x} \sum_{k=0}^\infty \frac{(-a)^k b_k(\lambda)}{(x-a)^{2k+1}}$	$\gamma(a,\lambda x) = \lambda^a \sum_{k=0}^\infty \gamma(a+k,x)\, \frac{(1-\lambda)^k}{k!}$
$P(a+1,z) = P(a,z) - \frac{z^a e^{-z}}{\Gamma(a+1)}$	$P(a+n,z) = P(a,z) - z^a e^{-z} \sum_{k=0}^{n-1} \frac{z^k}{\Gamma(a+k+1)}$

Table 3.3: Nearest neighbor examples.

Leave One Section Out

We evaluate our embeddings on the DLMF with leave-one-section-out cross-validation. We fit embeddings by performing EM on all data except for a single DLMF section. For each equation in the held-out section, we find its 5 nearest neighbors in the embedding space and compute a precision-at-5 score as the fraction of its 5 nearest neighbors that appear in the held-out section. We obtain a precision-at-5 score for each equation across the DLMF.

This metric is simply a proxy for performance and should be taken with a grain of salt since similar equations can be found in different sections and dissimilar equations can be

found in the same section.

For each held-out section s, we first stack the translation probability vectors as rows in matrices X_s and X_{-s}, where X_{-s} is the matrix containing all in-sample equations and X_s contains the out-of-sample equations. We then compute the k-SVD of the in-sample matrix

$$X_{-s} = U_{-s}\Sigma_{-s}V_{-s}^{\mathsf{T}}$$

For these experiments, $k = 100$. The embeddings for the in-sample equations are the rows of $E_{-s} = U_{-s}\Sigma_{-s}$. The embeddings for out-of-sample equations are obtained by projecting the translation probability vectors onto the right singular vectors $E_s = X_s V_{-s}$.

The translation probabilities were estimated using the entire dataset. Another study can be done in which the held-out section data is not used when estimating translation probabilities; this would truly make them out-of-sample embeddings.

We compare our method against embeddings obtained with TF-IDF. The IDF scores for the out-of-sample TF-IDF embeddings are computed using only in-sample data. The translation embeddings and TF-IDF vectors are normalized to unit norm before the nearest neighbors are found; this is equivalent to doing nearest neighbors by cosine similarity. We can think of TF-IDF as the "gold standard" for this task since our embeddings aim to capture syntactically similar equations while the TF-IDF approach directly leverages the syntax itself.

The distribution of the precision-at-5 scores are presented in Table (3.4) through their percentiles. We see that the median precision score for TF-IDF is 40% while the median precision score for the embedding method is 20%. Overall, TF-IDF does better on this task, but the translation embeddings give reasonable results as well given that syntax isn't directly modeled with the translation model.

percentile	tfidf	tr-embed
10%	0.0	0.0
20%	0.2	0.0
30%	0.2	0.2
40%	0.2	0.2
50%	0.4	0.2
60%	0.4	0.4
70%	0.6	0.6
80%	0.8	0.6
90%	1.0	1.0

Table 3.4: Distributions of precision-at-5 scores.

3.8 Discussion

We proposed a method for constructing embeddings for arbitrary objects. The method requires the practioner to specify a generative model for each object where the parameters of the model are "translation probabilities". Given parameter estimates for the generative model, the final stage of the method projects these translation probabilities down to a lower dimensional space. Our experiments show that this method yields embeddings competitive with standard methods, and also provides a principled way of constructing embeddings for variable-length objects like mathematical equations.

The translation embedding method consists of two components: the projection method and the probabilistic model. We suggest potential avenues of future work in both of these components.

3.8.1 Other Projection Methods

Let $v_e \in \mathbb{R}^V$ be a vector of translation probabilties for term/equation e. We wish to find embedding vectors $\phi_e \in \mathbb{R}^m$ for e, where $m \ll V$. Let $w_{e,f}$ be weights between items with high weights corresponding to similar items. These weights could be computed from coocccurence statistics. We want to preserve information in v while encouraging nearby

embeddings for similar equations. We might consider the objective function

$$\mathcal{L}(\phi, X; \lambda) = \sum_{e,f} w_{e,f} \|\phi_e - \phi_f\|^2 + \lambda \sum_e \|\phi_e - X v_e\|^2 \tag{3.8.1}$$

This is jointly convex in ϕ and X. This allows an embedding vector $\phi_e = X\beta_e$ to be calculated for a new item e as long as we have the representation v_e.

Another approach is to apply multidimensional scaling where we let d_{ij} be the Hellinger distance between the translation probability vectors for words i and j. We find embedding vectors $\phi_e \in \mathbb{R}^m$ to minimize the strain/stress:

$$S(\phi_1, \ldots, \phi_V) = \sum_{i \neq j} w_{ij} (d_{ij} - \|\phi_i - \phi_j\|)^2 \tag{3.8.2}$$

where w_{ij} is a similarity measure (e.g. d_{ij}^{-1}).

Finally, a last possible direction is to use canonical components analysis; this idea is a departure from considering the translation probability vectors as defined above. It is inspired by Dhillon et al. (2015).

Assume our corpus contains n tokens $\{t_1, \ldots, t_n\}$ each drawn from vocabulary of size V, where the vocabulary is $\{w_1, \ldots, w_V\}$.

Let $W \in \mathbb{R}^{n \times V}$ be the matrix of tokens ($W_{ij} = 1$ if the $t_i = w_j$ and zero otherwise). Let $T \in \mathbb{R}^{n \times V}$ where each entry is based on our generative model so that, for word embeddings, we have

$$T_{ij} = p(t_i = w_j | t_{i-k}^{i-1}) = \frac{1}{k} \sum_{a_t = i-k}^{i-1} t(t_i | t_{a_t}) \tag{3.8.3}$$

where $t(t_i|t_{a_t})$ are the estimated translation probabilities. We call each row of T a predictive probability vector. We seek an embedding for each word w_i such that words that have similar predictive probability vectors throughout the corpus also have similar embeddings. We use CCA(W, T) to find linear transformations $\phi_W \in \mathbb{R}^{V \times k}$ and $\phi_T \in \mathbb{R}^{V \times k}$. The embedding vectors would then be the rows of ϕ_W (we could also consider ϕ_T, $W\phi_W$, or $T\phi_T$).

In the next section, we briefly describe different kinds of similarity captured by the models proposed above and propose several ideas for future work.

3.8.2 Other Translation Models

We showed how the model specification together with the parallel text allows us to construct embeddings that capture different kinds of similarity. As discussed in Section (3.6), the word embedding model captured similarity associated with words appearing in similar contexts and words cooccurring frequently together. The former is the typical notion of similarity measured by popular word embedding methods. To preclude the latter form of similarity, we could specify a probabilistic model that dampens the probability of generating a word if it appears nearby. One way of doing this is to ignore words in the immediate history of the target word.

In Section (3.7), we were able to capture similar equations by modeling equation translation within sections of the Digital Library of Mathematical Functions. This approach captured syntactically similar equations because equations in the same section tend to have similar syntax. This approach was used as a testing ground for the embedding method since performing quantitative evaluations was simpler when measuring syntactic similarity. In our evaluations, we compared to TF-IDF, which directly measures syntactic similarity. We showed how our approach yields results similar to TF-IDF despite not explicitly computing statistics on symbol occurrence like the TF-IDF method.

While the syntactic similarity captured by the DLMF experiments provided an appropriate test case for the equation embeddings, the ultimate goal of the Hopper Project described in Section (3.2) is to capture semantic similarity of equations. This would allow equations to be considered similar whether or not they have the same syntax, and would be a more appropriate input for downstream tasks like document similarity and search.

One possible direction for future work would be to specify a translation model that translates between natural language and equations, similar in spirit to Yasunaga and Lafferty

(2019). The translation probabilities would be of the form $p(w_i|e_j)$ — the probability of word i being generated by mathematical symbol j. Parallel text would be of the form (text, equation) and could constructed from technical documents using the surrounding text of each equation as the "target" text, and the equation itself as the "source". We might expect to see, for example, a high translation probability for the word "probability" given the symbol Ω, since Ω is typically used as a symbol for a probability space. In this way, if two communities use the same natural language to describe a mathematical concept while using disparate mathematical symbols, this kind of approach would capture equation similarity because of the surrounding language. This line of thinking could be applied to more sophisticated model specifications to capture different kinds of similarity depending on the use case.

REFERENCES

Makoto Abe. A generalized additive model for discrete-choice data. *Journal of Business and Economic Statistics*, 17(3):271–284, 1999. ISSN 07350015. URL http://www.jstor.org/stable/1392286.

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

MOSEK ApS. *MOSEK Fusion API for C++ 8.0.0.94*, 2017. URL https://docs.mosek.com/8.0/cxxfusion/index.html.

S. Athey, J. Tibshirani, and S. Wager. Generalized Random Forests. *Forthcoming in Annals of Statistics*, 2018. URL https://arxiv.org/abs/1610.01271.

Susan Athey, David Blei, Robert Donnelly, Francisco Ruiz, and Tobias Schmidt. Estimating heterogeneous consumer preferences for restaurants and travel time using mobile location data. *AEA Papers and Proceedings*, 108:64–67, 2018. doi: 10.1257/pandp.20181031. URL http://www.aeaweb.org/articles?id=10.1257/pandp.20181031.

Miriam Ayer, H.D. Brunk, G.M. Ewing, W.T. Reid, and Edward Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 26:641–648, 1955.

Arie Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Mach. Learn.*, 19(1):29–43, April 1995. ISSN 0885-6125. doi: 10.1023/A:1022655006810.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944966.

Steven Berry, James Levinsohn, and Ariel Pakes. Automobile prices in market equilibrium. *Econometrica*, 63(4):841–90, 1995. URL https://EconPapers.repec.org/RePEc:ecm:emetrp:v:63:y:1995:i:4:p:841-90.

Steven T Berry and Philip A Haile. Nonparametric identification of multinomial choice demand models with heterogeneous consumers. (15276), August 2009. doi: 10.3386/w15276. URL http://www.nber.org/papers/w15276.

M. J. Betancourt and Mark Girolami. Hamiltonian Monte Carlo for Hierarchical Models. *arXiv e-prints*, art. arXiv:1312.0906, December 2013.

Matt Bonakdarpour, Sabyasachi Chatterjee, Rina Foygel Barber, and John Lafferty. Prediction rule reshaping. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 629–637, 2018. URL http://proceedings.mlr.press/v80/bonakdarpour18a.html.

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June 1993. ISSN 0891-2017.

Yining Chen and Richard J Samworth. Generalized additive and index models with shape constraints. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(4):729–754, 2016.

V. Chernozhukov, I. Fernández-Val, and A. Galichon. Improving point and interval estimators of monotone functions by rearrangement. *Biometrika*, 96(3):559–575, 2009.

V. Chernozhukov, I. Fernández-Val, and A. Galichon. Quantile and probability curves without crossing. *Econometrica*, 78(3):1093–1125, 2010. ISSN 1468-0262.

Pradeep K. Chintagunta and Harikesh S. Nair. Discrete-choice models of consumer demand in marketing. *Marketing Science*, 30(6):977–996, 2011. ISSN 07322399, 1526548X.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

Paramveer S Dhillon, Dean P Foster, and Lyle H Ungar. Eigenwords: spectral word embeddings. *Journal of Machine Learning Research*, 16:3035–3078, 2015. URL `http://www.jmlr.org/papers/volume16/dhillon15a/dhillon15a.pdf`.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

Ad Feelders and Martijn Pardoel. Pruning for monotone classification trees. In Michael R. Berthold, Hans-Joachim Lenz, Elizabeth Bradley, Rudolf Kruse, and Christian Borgelt, editors, *Advances in Intelligent Data Analysis V*, pages 1–12, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 406–414, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: 10.1145/371920.372094.

Rina Friedberg, Julie Tibshirani, Susan Athey, and Stefan Wager. Local Linear Forests. *arXiv e-prints*, art. arXiv:1807.11408, July 2018.

Jerome H. Friedman. Multivariate adaptive regression splines. *Ann. Statist.*, 19(1):1–67, 03 1991. doi: 10.1214/aos/1176347963.

Michael L. Ganz, Neil Wintfeld, Qian Li, Veronica Alas, Jakob Langer, and Mette Hammer. The association of body mass index with the risk of type 2 diabetes: a case–control study nested in an electronic health records system in the united states. *Diabetology & Metabolic Syndrome*, 6(1):50, Apr 2014. ISSN 1758-5996.

Sergio González, Francisco Herrera, and Salvador Garc´(i)a. Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. 33:367–388, 07 2015.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1321–1330, 2017. URL `http://proceedings.mlr.press/v70/guo17a.html`.

Maya Gupta, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, and Alexander van Esbroeck. Monotonic calibrated interpolated look-up tables. *Journal of Machine Learning Research*, 17(109):1–47, 2016.

Ahlem Hajjem, Francois Bellavance, and Denis Larocque. Mixed-effects random forest for clustered data. *Journal of Statistical Computation and Simulation*, 84:1313–1328, 06 2014.

Qiyang Han and Jon A Wellner. Multivariate convex regression: global risk bounds and adaptation. *arXiv preprint arXiv:1601.06844*, 2016.

Qiyang Han, Tengyao Wang, Sabyasachi Chatterjee, and Richard J Samworth. Isotonic regression in general dimensions. *arXiv preprint arXiv:1708.09468*, 2017.

Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statist. Sci.*, 1(3): 297–310, 08 1986. doi: 10.1214/ss/1177013604. URL `https://doi.org/10.1214/ss/1177013604`.

Joel L. Horowitz, Denis Bolduc, Suresh Divakar, John Geweke, Füsun Gönül, Vassilis Hajivassiliou, Frank S. Koppelman, Michael Keane, Rosa Matzkin, Peter Rossi, and Paul Ruud. Advances in random utility models report of the workshop on advances in random utility models duke invitational symposium on choice modeling behavior. *Marketing Letters*, 5(4):311–322, Oct 1994. ISSN 1573-059X. doi: 10.1007/BF00999207. URL `https://doi.org/10.1007/BF00999207`.

Harald Hruschka, Werner Fettes, and Markus Probst. Analyzing purchase data by a neural net extension of the multinomial logit model. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Artificial Neural Networks — ICANN 2001*, pages 790–795, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44668-2.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=2390524.2390645`.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691. Association for Computational Linguistics, 2015. doi: 10.3115/v1/P15-1162. URL http://aclweb.org/anthology/P15-1162.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5950-skip-thought-vectors.pdf.

Jon Kleinberg, Jens Ludwig, Sendhil Mullainathan, and Ziad Obermeyer. Prediction policy problems. *American Economic Review*, 105(5):491–95, May 2015. doi: 10.1257/aer.p20151023. URL http://www.aeaweb.org/articles?id=10.1257/aer.p20151023.

Rasmus Kyng, Anup Rao, and Sushant Sachdeva. Fast, provable algorithms for isotonic regression in all l_p-norms. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2719–2727. Curran Associates, Inc., 2015.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org, 2014. URL http://dl.acm.org/citation.cfm?id=3044805.3045025.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. pages 2177–2185, 2014. URL http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Kazuhisa Makino, Takashi Suda, Kojin Yano, and Toshihide Ibaraki. Data analysis by positive decision trees. In *CODAS*, pages 257–264, 1996.

James D Malley, Jochen Kruppa, Abhijit Dasgupta, Karen G Malley, and Andreas Ziegler. Probability machines: consistent probability estimation using nonparametric learning machines. *Methods of Information in Medicine*, 51(1):74, 2012.

Rosa L. Matzkin. Nonparametric identification and estimation of polychotomous choice models. *Journal of Econometrics*, 58(1):137 – 168, 1993. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(93)90116-M. URL http://www.sciencedirect.com/science/article/pii/030440769390116M.

Daniel McFadden. Conditional logit analysis of qualitiative choice behavior. In P. Zarembka, editor, *Frontiers in Econometrics*, pages 105–142. 1974.

Daniel McFadden and Kenneth Train. Mixed mnl models for discrete response. *Journal of Applied Econometrics*, 15(5):447–470, 2000. doi: 10.1002/1099-1255(200009/10)15:5⟨447::AID-JAE570⟩3.0.CO;2-1. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/1099-1255%28200009/10%2915%3A5%3C447%3A%3AAID-JAE570%3E3.0.CO%3B2-1.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

Mahdi Milani Fard, Kevin Canini, Andrew Cotter, Jan Pfeifer, and Maya Gupta. Fast and flexible monotonic functions with ensembles of lattices. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2919–2927. Curran Associates, Inc., 2016.

George A Miller and Walter G Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991. URL http://eric.ed.gov/ERICWebPortal/recordDetail?accno=EJ431389.

Alejandro Mottini and Rodrigo Acuna-Agost. Deep choice model using pointer networks for airline itinerary prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1575–1583, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098005. URL http://doi.acm.org/10.1145/3097983.3098005.

Alexandru Niculescu-Mizil and Rich Caruana. Obtaining calibrated probabilities from boosting. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 413–420, Arlington, Virginia, United States, 2005. AUAI Press. ISBN 0-9749039-1-4. URL http://dl.acm.org/citation.cfm?id=3020336.3020388.

F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, and B. V. Saunders. *NIST Digital Library of Mathematical Functions*. URL http://dlmf.nist.gov/.

Miguel Paredes, Erik Hemberg, Una-May O'Reilly, and Chris Zegras. Machine learning or discrete choice models for car ownership demand estimation and prediction? In *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2017 5th IEEE International Conference on*, pages 780–785. IEEE, 2017.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

R. Potharst and A. J. Feelders. Classification trees for problems with monotonicity constraints. *SIGKDD Explor. Newsl.*, 4(1):1–10, June 2002. ISSN 1931-0145.

Natalya Pya and Simon N. Wood. Shape constrained additive models. *Statistics and Computing*, 25(3):543–559, May 2015. ISSN 1573-1375.

David Revelt and Kenneth Train. Mixed Logit With Repeated Choices: Households' Choices Of Appliance Efficiency Level. *The Review of Economics and Statistics*, 80(4):647–657, November 1998.

David Revelt and Kenneth Train. Customer-Specific Taste Parameters and Mixed Logit: Households' Choice of Electricity Supplier. (0012001), January 2001. URL `https://ideas.repec.org/p/wpa/wuwpem/0012001.html`.

Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October 1965. ISSN 0001-0782.

Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. Exponential family embeddings. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 478–486. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6571-exponential-family-embeddings.pdf`.

Maja Rudolph, Francisco Ruiz, Susan Athey, and David Blei. Structured embedding models for grouped data. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 251–261. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/6629-structured-embedding-models-for-grouped-data.pdf`.

Francisco J. R. Ruiz, Susan Athey, and David M. Blei. SHOPPER: A Probabilistic Model of Consumer Choice with Substitutes and Complements. *arXiv e-prints*, art. arXiv:1711.03560, Nov 2017.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*, 2013a.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics, 2013b.

J. Spouge, H. Wan, and W.J. Wilbur. Least squares isotonic regression in two dimensions. *Journal of Optimization Theory and Applications*, 117(3):585–605, Jun 2003.

Stan Development Team. RStan: the R interface to Stan, 2018. URL `http://mc-stan.org/`. R package version 2.18.2.

Matthew Taddy, Chun-Sheng Chen, Jun Yu, and Mitch Wyle. Bayesian and empirical bayesian forests. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 967–976, Lille, France, 07–09 Jul 2015. PMLR.

Kenneth Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2009.

R. S. Uhler and J. G. Cragg. The structure of the asset portfolios of households. *The Review of Economic Studies*, 38(3):341–357, 1971. ISSN 00346527, 1467937X.

Yashen Wang, Heyan Huang, Chong Feng, Qiang Zhou, Jiahui Gu, and Xiong Gao. Cse: Conceptual sentence embeddings based on attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 505–515. Association for Computational Linguistics, 2016. doi: 10.18653/v1/P16-1048. URL `http://aclweb.org/anthology/P16-1048`.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. 2016.

Marvin N. Wright and Andreas Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017a. doi: 10.18637/jss.v077.i01.

Marvin N. Wright and Andreas Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017b.

Min Xu, Minhua Chen, and John Lafferty. Faithful variable screening for high-dimensional convex regression. *Ann. Statist.*, 44(6):2624–2660, 12 2016. doi: 10.1214/15-AOS1425.

Michihiro Yasunaga and John Lafferty. TopicEq: A Joint Topic and Mathematical Equation Model for Scientific Texts. In *Proceedings of AAAI 2019*, 2019.

Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. Deep lattice networks and partial monotonic functions. In *NIPS*, 2017.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 609–616, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL `http://dl.acm.org/citation.cfm?id=645530.655658`.