Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Conditional Gradient Approach for Nonparametric Estimation of Mixing Distributions

Srikanth Jagabathula, Lakshminarayanan Subramanian, Ashwin Venkataraman

To cite this article:
Srikanth Jagabathula, Lakshminarayanan Subramanian, Ashwin Venkataraman (2020) A Conditional Gradient Approach for Nonparametric Estimation of Mixing Distributions. Management Science

Published online in Articles in Advance 20 May 2020

. https://doi.org/10.1287/mnsc.2019.3373 http://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Articles in Advance, pp. 1–22
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

A Conditional Gradient Approach for Nonparametric Estimation of Mixing Distributions

Srikanth Jagabathula,[a,b] Lakshminarayanan Subramanian,[c] Ashwin Venkataraman[c,d]

[a] Stern School of Business, New York University, New York, New York 10012; [b] Harvard Business School, Harvard University, Boston, Massachusetts 02163; [c] Courant Institute of Mathematical Sciences, New York University, New York, New York 10012; [d] Harvard Institute for Quantitative Social Science, Harvard University, Cambridge, Massachusetts 02138

Contact: sjagabat@stern.nyu.edu, http://orcid.org/0000-0002-4854-3181 (SJ); lakshmi@cs.nyu.edu (LS); ashwin@cs.nyu.edu (AV)

Received: May 4, 2017
Revised: May 29, 2018; October 20, 2018; January 30, 2019
Accepted: March 21, 2019
Published Online in Articles in Advance: May 20, 2020

https://doi.org/10.1287/mnsc.2019.3373

Abstract. Mixture models are versatile tools that are used extensively in many fields, including operations, marketing, and econometrics. The main challenge in estimating mixture models is that the mixing distribution is often unknown, and imposing a priori parametric assumptions can lead to model misspecification issues. In this paper, we propose a new methodology for nonparametric estimation of the mixing distribution of a mixture of logit models. We formulate the likelihood-based estimation problem as a constrained convex program and apply the conditional gradient (also known as Frank–Wolfe) algorithm to solve this convex program. We show that our method iteratively generates the support of the mixing distribution and the mixing proportions. Theoretically, we establish the sublinear convergence rate of our estimator and characterize the structure of the recovered mixing distribution. Empirically, we test our approach on real-world datasets. We show that it outperforms the standard expectation-maximization (EM) benchmark on speed (16 times faster), in-sample fit (up to 24% reduction in the log-likelihood loss), and predictive (average 28% reduction in standard error metrics) and decision accuracies (extracts around 23% more revenue). On synthetic data, we show that our estimator is robust to different ground-truth mixing distributions and can also account for endogeneity.

History: Accepted by Serguei Netessine, operations management.
Funding: S. Jagabathula's and A. Venkataraman's research was supported in part by the National Science Foundation [Grant CMMI-1454310].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/mnsc.2019.3373.

Keywords: nonparametric estimation • mixture of logit • convex optimization • consideration sets

1. Introduction

Mixture models are used for modeling a wide range of phenomena in many fields. Within operations, they have been used to model customer demand, which changes in response to the changes a firm makes to its product offerings. Predicting these changes allows firms to optimize their product and price offerings, such as discontinuing low-demand products or enforcing price changes to shift demand to specific products. Demand predictions also serve as key inputs to inventory control and price-optimization models that are used in retail operations and revenue-management (RM) systems. A typical prediction problem involves fitting a mixture model to historical sales transactions and inventory data. The most popular model that is fit is the mixture of multinomial logit (MNL) models, also known simply as the mixture of logit models. This model has received considerable attention in the literature and has also been successfully applied in practice. In addition, it has been shown to approximate a wide class of mixtures (McFadden and Train 2000).

Because of its significance for demand modeling, we focus on the problem of estimating the mixing distribution of a mixture of logit models, from sales transaction and inventory data. The main challenge in this problem is that the structure of the mixing distribution is not known in practice. A common workaround is to assume that the mixing distribution comes from a prespecified *parametric* family, such as the normal or the log-normal distribution, and then estimate the parameters via maximum-likelihood estimation (Train 2009). This approach is reasonable when there is some prior knowledge about the structure of the underlying mixing distribution. But when no such knowledge exists, as often happens in practice, the ground-truth mixing distribution may very well not conform to the imposed parametric form. This leads to model misspecification, which can result in biased parameter estimates (Train 2008) or low goodness-of-fit measures (Fox et al. 2011).

To avoid model misspecification, we take a nonparametric approach, in which we search for the best-fitting mixing distribution from the class of all possible mixing distributions. The challenge with this approach is a computational one. The class of all possible mixing distributions lacks sufficient structure to allow for tractable estimation methods. One approach in the literature has been to approximate the class of all possible mixing distributions with another (large) class and then search for the best-fitting mixing distribution in that approximate space. For instance, Train (2008) takes this approach, in which the space of all mixing distributions is approximated with the class of finite mixtures of normal distributions or the class of discrete distributions with a large support size. Such approximations allow the application of standard optimization techniques, such as the expectation-maximization (EM) framework. But the resulting optimization problems are still nonconvex and become difficult to solve, running into numerical issues, as the number of parameters increases.

Our main contribution in this paper is to reformulate the nonparametric mixture-estimation problem as a constrained convex program, *without* resorting to any approximations to the space of all possible mixing distributions. We pose the mixture-estimation problem as the problem of searching for the distribution that minimizes a loss function from among the class of all possible mixing distributions. The standard log-likelihood loss (which results in the maximum-likelihood estimator) and squared loss are two example loss functions. Then, we use the insight that the mixing distribution affects the objective function only through the choice probabilities it predicts for the observed choices in the data; we call the vector of these choice probabilities the *data mixture-likelihood vector*. Now, instead of optimizing over the space of mixing distributions, the mixture-estimation problem can be solved by directly optimizing over the space of all possible data mixture-likelihood vectors. The constraints ensure that the mixture-likelihood vector is indeed consistent with a valid mixing distribution. We show that for the standard loss functions used in the literature, the objective function is convex in the mixture-likelihood vector. Furthermore, although not a priori clear, we show that the constraint set is also convex. Together, these two properties result in a constrained convex program formulation for the mixture-estimation problem. We emphasize here that, although we obtain a convex program, the constraint space lacks an efficient description. Therefore, the resulting program, though convex, may be theoretically hard to solve. Nevertheless, there is vast literature on solving such convex programs, which we leverage to obtain scalable and numerically stable methods that are efficient for special cases and result in good approximations more generally.

A more immediate concern is that simply solving the above program is not sufficient because the optimal solution will be expressed as the mixture-likelihood vector and not as the mixing distribution. Backing out the underlying mixing distribution from the mixture-likelihood vector may again be a computationally intensive exercise. To counter this issue, we apply the conditional gradient (also known as (a.k.a.) Frank–Wolfe) algorithm to solve the above convex program. We show that the special structure of the conditional-gradient (CG) algorithm allows it to simultaneously perform both the tasks of optimizing over the predicted choice probabilities *and* recovering the best-fitting mixing distribution. The CG algorithm is an iterative first-order method for constrained convex optimization. We show that, when applied to our method, each iteration of the CG algorithm yields a single mixture component. The CG algorithm has seen an impressive revival in the machine-learning literature recently because of its favorable properties compared with standard projected/proximal gradient methods, such as efficient handling of complex constraint sets. The vast literature on the CG algorithm in the machine-learning area confers two key advantages to our estimation technique: (a) availability of precise convergence guarantees (Lacoste-Julien and Jaggi 2015) and (b) scalability to large-scale and high-dimensional settings (Wang et al. 2016).

1.1. Summary of Key Results

Our work makes the following contributions:

1. *Novel mixture-estimation methodology.* Our estimator is (a) *general-purpose*: can be applied with little to no customization for a broad class of loss functions; (b) *fast*: order-of-magnitude faster than the benchmark expectation-maximization algorithm; and (c) *nonparametric*: makes no assumption on the mixing distribution and estimates customer types in the population in a data-driven fashion.

2. *Analytical results.* We obtain two key theoretical results:

(i) We provide a sublinear convergence guarantee—that is, $O(1/k)$ after k iterations—for our CG-based estimator, for both the log-likelihood and squared loss functions. Refer to Theorem 1 and Section 5.1 for details.

(ii) We characterize the structure of the mixing distribution recovered by our estimator. Our method recovers two types of mixture components: what we call (a) nonboundary and (b) boundary types. A nonboundary type is described by a standard logit model with a parameter vector $\boldsymbol{\omega}$. The boundary types, on the other hand, are limiting logit models that result from unbounded solutions in which the parameter vector $\boldsymbol{\omega}$ is pushed to infinity. We show that each boundary type can be described by two parameters, $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$. The parameter vector $\boldsymbol{\theta}$ induces a (weak) preference order over the set of products and determines a consideration set the customer forms, when given an offer set.

The parameter vector ω_0 then determines the logit choice probabilities from within the consideration set. Refer to Section 5.2 for details.

For the case of a single offer set (such as market-share data), we also identify conditions under which our estimator recovers boundary types and characterize the corresponding consideration sets of the recovered types. Our conditions depend on the geometry of the observed product features, viz. the (convex) polytope formed by the convex hull of the product feature vectors; see Section 5.3.1 for details. In addition, when some features are binary, we show that our estimator recovers boundary types in *each* iteration, with the consideration sets reflecting strong noncompensatory preferences in the population; refer to Section 5.3.2 for more details.

3. *Empirical results.* We conducted three numerical studies to validate our methodology:

(a) Using synthetic data, we show that our estimator is robust to several complex ground-truth mixing distributions and consistently recovers a good approximation to the underlying distribution. Note that this is despite the fact that our estimator has no knowledge of the true mixing distribution. In particular, its performance is significantly better than a standard benchmark method imposing a parametric assumption on the mixing distribution and highlights the potential impact of model misspecification in practice.

(b) On the SUSHI Preference Data Set (Kamishima et al. 2005), where customers rank different sushi varieties according to their preference, we show that our method achieves superior in-sample fit compared with fitting a latent-class MNL (LC-MNL) model using the EM algorithm (Bhat 1997) for both the log-likelihood (24% better) and squared loss (58% better), with 16× speedup in the estimation time. The CG algorithm iteratively adds customer types that explain the observed-choice data to the mixing distribution, which results in a much better fit as compared with the EM algorithm that updates all customer types together in each iteration. Our approach also achieves better predictive accuracy than EM, with an average 28% and 16% reduction in the root mean square error (RMSE) and mean absolute percentage error (MAPE) metrics, respectively, for predicting market shares on new assortments. In solving the assortment decision, we show that our method can extract up to 23% more revenue from the population than the EM benchmark.

(c) On real-world sales-transaction data from the IRI Academic Data Set (Bronnenberg et al. 2008), we show that our method achieves up to 8% (respectively (resp.), 7%) and 7% (resp., 5%) reduction, respectively, in the in-sample and out-of-sample log-likelihood loss (resp., squared loss), compared with the EM benchmark. In particular, we outperformed EM-based estimation in all five product categories that we considered.

2. Relevant Literature

Our work has connections to two broad areas.

2.1. Nonparametric Maximum-Likelihood Estimation (NPMLE)

Our estimation approach generalizes the NPMLE techniques—our method is applicable for any convex loss function, including the standard log-likelihood loss—which have a long and rich history in classical statistics (Robbins 1950, Kiefer and Wolfowitz 1956). These techniques search for a distribution that maximizes the likelihood function from a large class of mixing distributions. In the context of studying properties of the maximum-likelihood estimator (such as existence, uniqueness, support size, etc.) for the mixing distribution via the geometric structure of the constraint set, Lindsay (1983) shows that when the mixing distribution is unrestricted, the NPMLE can be formulated as a convex program. However, such a formulation is computationally difficult to solve when the underlying parameter space is high dimensional. To address this issue, existing work has taken two approaches. The first approach reduces the search space to a large (but finite) number of mixture components and uses the EM algorithm for estimation (Laird 1978). Although now the estimation problem is finite-dimensional, convexity is lost, and standard issues related to nonconvexity and finite mixture models become a significant obstacle (McLachlan and Peel 2000). The second approach retains convexity but gains tractability through a finite-dimensional convex approximation, where the support of the mixing distribution is assumed to be finite and prespecified (such as a uniform grid) and only the mixing weights need to be estimated. Fox et al. (2011) specialize this approach to estimating a mixture of logit models. However, it is unclear how to choose the support. When the dimensionality of the parameter space is small, Fox et al. (2011) demonstrate that a uniform grid is sufficient to reasonably capture the underlying distribution, but this approach quickly becomes intractable for even moderately large parameter dimensions.[1] Consequently, existing techniques have usually focused on simple models with univariate or low-dimensional (bivariate and trivariate) mixing distributions (Bohning et al. 1992, Jiang and Zhang 2009, Feng and Dicker 2018) to retain tractability.

In the context of the above, we avoid the issues resulting from the nonconvex formulation by retaining convexity, but at the same time, we do *not* need access to a prespecified support. We leverage the conditional gradient algorithm to directly solve the seemingly intractable convex program, which iteratively generates the support of the mixing distribution by searching over the underlying parameter space. This allows our method to scale to higher-dimensional settings: 5 in our SUSHI case study and 11 in the IRI case study; see Sections 7.1 and 7.2.

2.2. Conditional Gradient Algorithms

The conditional gradient algorithm is one of the earliest methods (Frank and Wolfe 1956) for constrained convex optimization and has recently seen an impressive revival for solving large-scale problems with structured constraint sets (see Clarkson 2010 and Jaggi 2011 for excellent overviews). The algorithm has been used in diverse domains, including computer vision (Joulin et al. 2014), submodular function optimization (Bach 2013), and collaborative filtering (Jaggi and Sulovsk 2010), as well as inference in graphical models (Krishnan et al. 2015). In addition, numerous related variants of the algorithm have been proposed, such as solving nonlinear subproblems to increase sparsity (Zhang 2003) and incorporating regularization to improve predictive performance (Harchaoui et al. 2015). In terms of theoretical performance, Jaggi (2013) gave a convergence analysis that guarantees an error of at most $O(1/k)$ (sublinear convergence) after k iterations for any compact convex constraint set. Recently, Lacoste-Julien and Jaggi (2015) proved that many variants of the classical Frank–Wolfe algorithm enjoy global linear convergence for any strongly convex function optimized over a polytope domain.

Our main contribution is leveraging the conditional gradient algorithm for estimating the mixing distribution, which also allows us to provide convergence guarantees for our estimator. For the squared loss function, the sublinear convergence rate of $O(1/k)$ after k iterations follows from existing results. But, for the log-likelihood loss, existing results don't apply because the gradient blows up at the boundary of the constraint region. We address this issue by showing that the iterates produced by the fully corrective variant of the CG algorithm (the one that we implement) are strictly bounded away from the boundary. We then adapt and extend existing arguments to establish the same $O(1/k)$ sublinear convergence guarantee, as for the squared loss. We also show that, under appropriate structure in the observed product features, our estimator converges to the optimal solution in a *finite* number of iterations (see Section 5.3).

3. Problem Setup and Formulation

We consider a universe $[n] \overset{\text{def}}{=} \{1, 2, \ldots, n\}$ of n products, which customers interact with over $T \geq 1$ discrete time periods.[2] We assume access to aggregate sales data for these n products in each time period.[3] In each time period $t \in [T]$, the firm offers a subset $S_t \subseteq [n]$ of products to the customers and collects sales counts for each of the products. We let N_{jt} denote the number of times product j was purchased in period t, $N_t \overset{\text{def}}{=} \sum_{j \in S_t} N_{jt}$ denote the total number of sales in period t, and $N \overset{\text{def}}{=} \sum_{t \in [T]} N_t$ denote the total number of sales over all the time periods. We suppose that we observe at least one sale in each period t, so that $N_t > 0$ for all periods $t \in [T]$; if there was no observed sale in a time period, then we assume that it was already dropped from the observation periods. Let $\mathsf{Data} \overset{\text{def}}{=} \{(N_{jt} : j \in S_t) \mid t \in [T]\}$ denote all the observations collected over the T discrete time periods. We assume that product $j \in S_t$ is represented by a D-dimensional feature vector $\mathbf{z}_{jt}$ in some feature space $\mathcal{Z} \subseteq \mathbb{R}^D$. Example features include price, brand, and color. Product features could vary over time; for instance, product prices may vary because of promotions, discounts, and so forth. In practice, these data are often available to firms in the form of purchase transactions, which provide sales information, and inventory data, which provide offer-set information.

We assume that each customer makes choices according to an MNL (a.k.a. logit) model, which specifies that a customer purchases product j from offer-set S with probability

$$f_{j,S}(\boldsymbol{\omega}) = \frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_{jS}\right)}{\sum_{\ell \in S} \exp\left(\boldsymbol{\omega}^\top \mathbf{z}_{\ell S}\right)}, \tag{1}$$

where $\mathbf{z}_{\ell S}$ is the feature vector of product ℓ when offered as part of offer-set S and $\boldsymbol{\omega}$ is the parameter or "taste" vector. This taste vector specifies the "value" that a customer places on each of the product features in deciding which product to purchase. Customers often have heterogeneous preferences over product features. To capture this heterogeneity, we assume that the population of customers is described by a mixture of MNL models, where in each choice instance, a customer samples a vector $\boldsymbol{\omega}$ according to some distribution Q (over the parameter space $\mathbb{R}^D$) and then makes choices according to the MNL model with parameter vector $\boldsymbol{\omega}$.

Our goal is to estimate the best-fitting mixing distribution to the collection Data of sales observations, from the class of all possible mixing distributions $\mathcal{Q} \overset{\text{def}}{=} \{Q : Q \text{ is a distribution over } \mathbb{R}^D\}$. The fit to the data is measured via a *loss function* that quantifies the mismatch between the observed sales fractions in Data and those predicted by the mixture of logit model. In order to state the problem formally, we need to introduce additional notation. For each (product, offer-set) pair, define the mapping $\mathrm{g}_{jt} : \mathcal{Q} \to [0, 1]$ as

$$\mathrm{g}_{jt}(Q) = \int f_{jt}(\boldsymbol{\omega})\, \mathrm{d}Q(\boldsymbol{\omega}),$$

where for brevity of notation, we let $f_{jt}(\boldsymbol{\omega})$ denote $f_{j,S_t}(\boldsymbol{\omega})$. In other words, $\mathrm{g}_{jt}(Q)$ is the probability of choosing product j from offer-set S_t under the mixing distribution Q. Let $M \overset{\text{def}}{=} |S_1| + \cdots + |S_T|$ and $\mathbf{g} : \mathcal{Q} \to [0, 1]^M$

denote the vector-valued mapping, defined as $\mathbf{g}(Q) = (g_{jt}(Q) : t \in [T],\ j \in S_t)$. We call $\mathbf{g}(Q)$ the *data mixture-likelihood vector*, or simply the *mixture-likelihood vector*, under mixing distribution Q. We let $\mathcal{G} = \{\mathbf{g}(Q) : Q \in \mathcal{Q}\}$ denote the set of all mixture-likelihood vectors.

Given the above, our goal is to solve the following problem:

$$\min_{Q \in \mathcal{Q}} \ \mathsf{loss}(\mathbf{g}(Q); \mathsf{Data}), \qquad (\text{Mixture Estimation})$$

where $\mathsf{loss}(\cdot; \mathsf{Data}) : \mathcal{G} \to \mathbb{R}_{\geq 0} \cup \{+\infty\}$ is a nonnegative convex function. We make the standard assumption that $\mathsf{loss}(\cdot)$ is continuously differentiable on the relative interior of $\mathcal{G}$. Two example functions include:

• **Negative log-likelihood (NLL) loss**: This loss function is by far the most widely used in practice (Train 2008):

$$\mathsf{NLL}(\mathbf{g}(Q); \mathsf{Data}) = -\frac{1}{N}\sum_{t=1}^{T}\sum_{j \in S_t} N_{jt} \log\big(g_{jt}(Q)\big).$$

• **Squared (SQ) loss**: This loss function was employed by Fox et al. (2011):

$$\mathsf{SQ}(\mathbf{g}(Q); \mathsf{Data}) = \frac{1}{2 \cdot N}\sum_{t=1}^{T} N_t \cdot \sum_{j \in S_t}\big(g_{jt}(Q) - y_{jt}\big)^2,$$

where $y_{jt} \overset{\text{def}}{=} N_{jt}/N_t$ denotes the fraction of sales for product j in offer-set S_t.

We first describe traditional approaches to solving the Mixture Estimation problem, and their limitations, to motivate the need for our approach.

3.1. Traditional Approaches to Mixture Estimation

Directly solving the Mixture Estimation problem is challenging due to the complexity of searching over all possible mixing distributions. Consequently, traditional approaches assume that the mixing distribution belongs to a family $\mathcal{Q}(\Theta)$ of distributions parametrized via parameter space Θ, where $\mathcal{Q}(\Theta) \overset{\text{def}}{=} \{Q_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$ and $Q_{\boldsymbol{\theta}}$ is the mixing distribution corresponding to the parameter vector $\boldsymbol{\theta} \in \Theta$. The best-fitting distribution is then obtained by solving the following likelihood problem:[4]

$$\min_{\boldsymbol{\theta} \in \Theta} \ -\frac{1}{N}\sum_{t=1}^{T}\sum_{j \in S_t} N_{jt} \log\left(\int f_{jt}(\boldsymbol{\omega})\, dQ_{\boldsymbol{\theta}}(\boldsymbol{\omega})\right). \qquad (2)$$

Different assumptions for the family $\mathcal{Q}(\Theta)$ result in different estimation techniques.

The most common assumption is that the mixing distribution follows a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, parametrized by $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\Sigma}$ is the variance-covariance matrix of the distribution. The resulting model is referred to as the random parameters logit (RPL) model (Train 2009), and the corresponding likelihood problem is given by

$$\min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} -\frac{1}{N}\sum_{t=1}^{T}\sum_{j \in S_t} N_{jt} \log\Bigg(\int f_{jt}(\boldsymbol{\omega}) \cdot \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} \cdot \exp\left(-\frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\omega} - \boldsymbol{\mu})\right) d\boldsymbol{\omega}\Bigg). \qquad (3)$$

The integral in the above problem is often approximated through a Monte Carlo simulation.

The other common assumption is that the mixing distribution has a finite support of size K. The distribution is then parametrized by $\boldsymbol{\theta} = (\alpha_1, \ldots, \alpha_K, \boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K)$, where $(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K)$ denotes the support of the distribution and $(\alpha_1, \ldots, \alpha_K)$ denote the corresponding mixture proportions with $\sum_{k \in [K]} \alpha_k = 1$ and $\alpha_k \geq 0$ for all $k \in [K]$. The resulting model is referred to as the latent-class MNL model (Bhat 1997), and the corresponding likelihood problem is given by

$$\min_{\substack{\alpha_1, \alpha_2, \ldots, \alpha_K \\ \boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_K}} -\frac{1}{N}\sum_{t=1}^{T}\sum_{j \in S_t} N_{jt} \log\left(\sum_{k=1}^{K} \alpha_k f_{jt}(\boldsymbol{\omega}_k)\right)$$
$$\text{subject to} \sum_{k \in [K]} \alpha_k = 1, \alpha_k \geq 0 \ \forall\ k \in [K]. \qquad (4)$$

Although commonly used, these traditional approaches suffer from two key limitations:

• *Model misspecification*: The most significant issue with traditional approaches is model misspecification, which occurs when the ground-truth mixing distribution is not contained in the search space $\mathcal{Q}(\Theta)$. In practice, such misspecification is common because the selection of the search space $\mathcal{Q}(\Theta)$ is often driven by tractability considerations as opposed to knowledge of the structure of the ground-truth mixing distribution. Model misspecification can result in biased parameter estimates (Train 2008) and low goodness-of-fit measures (Fox et al. 2011).

• *Computational issues*: Another practical issue is that, even if the model is not misspecified, the resulting likelihood problems are nonconvex and therefore hard to solve in general.

3.2. Our Approach: Mixture Estimation by Solving a Convex Program

Our approach is designed to address the challenges described above. We avoid the model misspecification issue, as we directly search over all possible mixing distributions, instead of restricting our search to specific parametric families.[5] However, this can introduce computational concerns, given the complexity of the search space $\mathcal{Q}$. To address the computational issue, we formulate the Mixture Estimation problem as a constrained convex program, as described next.

This formulation allows us to tap into the vast existing literature on solving convex programs efficiently.

We now describe the steps in our formulation. We first observe that the objective function only depends on Q through the corresponding mixture-likelihood vector $\mathbf{g}(Q)$. Therefore, instead of searching over the mixing distributions, we directly search over the mixture-likelihood vectors, obtaining the following equivalence:

$$\min_{Q\in\mathcal{Q}} \mathsf{loss}(\mathbf{g}(Q)) \quad\equiv\quad \min_{\mathbf{g}\in\mathcal{G}} \mathsf{loss}(\mathbf{g}), \tag{5}$$

where we have dropped the explicit dependence of loss on Data for simplicity of notation. Recall that $\mathcal{G} = \{\mathbf{g}(Q) : Q \in \mathcal{Q}\}$ is the set of all possible mixture-likelihood vectors.

With the above equivalence, our ability to solve the Mixture Estimation problem depends on our ability to describe the constraint set $\mathcal{G}$. We show next that this constraint set can indeed be expressed as a convex set. For that, analogous to the mixture-likelihood vector earlier, define the *atomic-likelihood vector* $\boldsymbol{f}(\boldsymbol{\omega}) \overset{\text{def}}{=} (f_{jt}(\boldsymbol{\omega}) :\ t \in [T], j \in S_t)$. We let $\mathcal{P} = \{\boldsymbol{f}(\boldsymbol{\omega}) : \boldsymbol{\omega} \in \mathbb{R}^D\}$ denote the set of all possible atomic-likelihood vectors and $\overline{\mathcal{P}}$ denote its closure.[6] Now, it is clear that if $Q \in \mathcal{Q}$ is a discrete distribution with finite support $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_K$ and corresponding mixing weights $\alpha_1, \alpha_2, \ldots, \alpha_K$, then $\mathbf{g}(Q) = \sum_{k=1}^{K} \alpha_k \boldsymbol{f}(\boldsymbol{\omega}_k)$, and so $\mathbf{g}(Q)$ belongs to the convex hull, $\operatorname{conv}(\overline{\mathcal{P}})$, of vectors in $\overline{\mathcal{P}}$, defined as

$$\operatorname{conv}(\overline{\mathcal{P}}) = \Bigg\{ \sum_{\boldsymbol{f}\in\mathcal{F}} \alpha_f \boldsymbol{f} \ :\ \mathcal{F} \subset \overline{\mathcal{P}} \text{ is finite and } \sum_{\boldsymbol{f}\in\mathcal{F}} \alpha_f = 1, \alpha_f \geq 0 \ \ \forall \boldsymbol{f} \in \mathcal{F} \Bigg\}.$$

It can be verified that $\operatorname{conv}(\overline{\mathcal{P}})$ is a convex set in $\mathbb{R}^M$. In other words, for any discrete mixing distribution Q with finite support, we can express $\mathbf{g}(Q)$ as a convex combination of atomic-likelihood vectors $\{\boldsymbol{f}\}_{\boldsymbol{f}\in\mathcal{F}}$ for some finite subset $\mathcal{F} \subset \overline{\mathcal{P}}$. More generally, it can be shown (Lindsay 1983) that $\mathcal{G} = \operatorname{conv}(\overline{\mathcal{P}})$—that is, the set of all possible mixture-likelihood vectors coincides with the convex hull of all atomic-likelihood vectors. This fact, combined with the equivalence in (5), implies that instead of solving Mixture Estimation, we can equivalently solve the following problem:

$$\min_{\mathbf{g}\in\operatorname{conv}(\overline{\mathcal{P}})} \mathsf{loss}(\mathbf{g}). \tag{Convex Mixture}$$

We can show that the above is a constrained convex program (the proof is given in Online Appendix A.1):

Lemma 1. *For any convex function* $\mathsf{loss}(\cdot)$, *Convex Mixture is a convex program with a compact constraint set in the Euclidean space.*

So, our task now is to solve the Convex Mixture problem. However, solving it alone does not provide the mixing distribution—it only provides the optimal mixture-likelihood vector. We show next that the conditional gradient algorithm is the ideal candidate to not only obtain the optimal mixture-likelihood vector, but also the optimal mixing distribution.

4. Conditional Gradient Algorithm for Estimating the Mixing Distribution

We now apply the conditional gradient (hereafter, CG) algorithm to solve the (Convex Mixture) problem. The CG algorithm (Frank and Wolfe 1956, Jaggi 2013) is an iterative method for solving constrained convex programs. It has seen an impressive revival in the machine-learning literature recently because of its favorable properties compared with standard projected/proximal gradient methods, such as efficient handling of complex constraint sets. Online Appendix C provides an overview of the general CG algorithm. Here, we describe how it applies to solving $\min_{\mathbf{g}\in\operatorname{conv}(\overline{\mathcal{P}})} \mathsf{loss}(\mathbf{g})$.

The CG algorithm is an iterative first-order method that starts from an initial feasible solution, say, $\mathbf{g}^{(0)} \in \operatorname{conv}(\overline{\mathcal{P}})$, and generates a sequence of feasible solutions $\mathbf{g}^{(1)}, \mathbf{g}^{(2)}, \ldots$ that converge to the optimal solution. Letting $\nabla\mathsf{loss}(\cdot)$ denote the gradient of the loss function and $\langle\cdot,\cdot\rangle$ the standard inner product in the Euclidean space, the algorithm computes a *descent direction* $\mathbf{d}$ such that $\langle \nabla\mathsf{loss}(\mathbf{g}^{(k-1)}), \mathbf{d}\rangle < 0$ in iteration $k \geq 1$ and takes a suitable step in that direction (see, e.g. Nocedal and Wright 2006). The main distinction of the CG algorithm is that it always chooses *feasible* descent steps, where by a feasible step, we mean a step from the current solution toward the next solution such that the next solution remains feasible as long as the current is feasible. By contrast, other classical algorithms may take infeasible steps, which are then projected back onto the feasible region after each step; such projection steps are usually computationally expensive. To find a feasible step, the CG algorithm first obtains a descent direction by optimizing a linear approximation of the convex loss function at the current iterate $\mathbf{g}^{(k-1)}$:

$$\min_{\mathbf{v}\in\operatorname{conv}(\overline{\mathcal{P}})} \mathsf{loss}\left(\mathbf{g}^{(k-1)}\right) + \left\langle \nabla\mathsf{loss}\left(\mathbf{g}^{(k-1)}\right), \mathbf{v} - \mathbf{g}^{(k-1)} \right\rangle, \tag{6}$$

where the objective function in the above subproblem describes a supporting hyperplane to the convex loss function $\mathsf{loss}(\cdot)$ at the current iterate $\mathbf{g}^{(k-1)}$. The optimal solution, say, $\mathbf{v}^*$, provides the optimal direction $\mathbf{d}^* = \mathbf{v}^* - \mathbf{g}^{(k-1)}$. It can be shown that $\mathbf{d}^*$ is a descent direction if $\mathbf{g}^{(k-1)}$ is not already an optimal solution to the Convex Mixture problem. The next solution $\mathbf{g}^{(k)}$ is obtained by taking a step $\alpha \in [0, 1]$ in the direction of $\mathbf{d}^*$, so that $\mathbf{g}^{(k)} = \mathbf{g}^{(k-1)} + \alpha\mathbf{d}^* = \alpha\mathbf{v}^* + (1-\alpha)\mathbf{g}^{(k-1)}$. Because $\mathbf{v}^*$ and $\mathbf{g}^{(k-1)}$ both belong to $\text{conv}(\overline{\mathcal{P}})$ and $\text{conv}(\overline{\mathcal{P}})$ is convex, it follows that $\mathbf{g}^{(k)} \in \text{conv}(\overline{\mathcal{P}})$ for any $\alpha \in [0,1]$.

Solving the above subproblem is the most computationally challenging component in each iteration of the CG algorithm. In our context, we have additional structure that we can exploit to solve this subproblem. Specifically, the objective function is linear in the decision variable $\mathbf{v}$. And, linear functions always achieve optimal solutions at extreme points when optimized over a convex set. Our constraint set $\text{conv}(\overline{\mathcal{P}})$ is the convex hull of all the atomic-likelihood vectors in $\overline{\mathcal{P}}$. Therefore, the set of extreme points of the constraint set is a subset of $\overline{\mathcal{P}}$. It thus follows that it is sufficient to search over the set of all atomic-likelihood vectors in $\overline{\mathcal{P}}$, resulting in the following optimization problem:

$$\min_{\mathbf{v}\in\overline{\mathcal{P}}} \left\langle \nabla \text{loss}\left(\mathbf{g}^{(k-1)}\right), \mathbf{v} - \mathbf{g}^{(k-1)} \right\rangle. \tag{7}$$

Our ability to solve (7) efficiently depends on the structure of the set $\overline{\mathcal{P}}$. We discuss this aspect in more detail in Section 4.1. For now, we suppose that we have access to an oracle that returns an optimal solution, say, $\boldsymbol{f}^{(k)}$, to (7) in each iteration k.

In summary, in each iteration, the CG algorithm finds a new customer type (or atomic-likelihood vector) $\boldsymbol{f}^{(k)} \in \overline{\mathcal{P}}$ and obtains the new solution $\mathbf{g}^{(k)} = \alpha\boldsymbol{f}^{(k)} + (1-\alpha)\mathbf{g}^{(k-1)}$ by putting a probability mass α on the new customer type $\boldsymbol{f}^{(k)}$ and the remaining probability mass $1-\alpha$ on the previous solution $\mathbf{g}^{(k-1)}$, for some $\alpha \in [0,1]$. In other words, the CG algorithm is iteratively adding customer types $\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, \ldots$ to the support of the mixing distribution. This aspect of the CG algorithm makes it most attractive for estimating mixing distributions. In particular, it has two implications: (a) By maintaining the individual customer types and the step sizes, we can maintain the entire mixing distribution along with the current solution $\mathbf{g}^{(k)}$ in each iteration k (see below for details); and (b) because each iteration adds (at most) one new customer type to the support, terminating the program at iteration K results in a distribution with at most K mixture components. We use the latter property to control the complexity, as measured in terms of the number of mixture components, of the recovered mixing distribution (see the discussion on "stopping conditions" in Section 4.1.4).

We now discuss the choice of the step size α. The standard variant of the CG algorithm does a line-search to compute the optimal step size that results in the maximum improvement in the objective value. Instead, we use the "fully corrective" Frank–Wolfe (FCFW) variant (Shalev-Shwartz et al. 2010), which, after finding $\boldsymbol{f}^{(k)}$ at iteration k, reoptimizes the loss function $\text{loss}(\mathbf{g})$ over the convex hull of the initial solution $\mathbf{g}^{(0)}$ and the atomic-likelihood vectors $\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, \ldots, \boldsymbol{f}^{(k)}$ found so far. More precisely, the algorithm computes weights $\boldsymbol{\alpha}^{(k)}$ from the $(k+1)$-dimensional simplex Δ_k that minimize the loss function and obtains the next iterate $\mathbf{g}^{(k)} := \alpha_0^{(k)}\mathbf{g}^{(0)} + \sum_{s=1}^{k} \alpha_s^{(k)}\boldsymbol{f}^{(s)}$. The weights $\boldsymbol{\alpha}^{(k)} = (\alpha_0^{(k)}, \alpha_1^{(k)}, \alpha_2^{(k)}, \ldots, \alpha_k^{(k)})$ represent the proportions of each of the mixture components. This variant of the CG algorithm makes more progress in each iteration and is therefore most suited when the subproblems in (7) are hard to solve. It also promotes sparser solutions (Jaggi 2013) containing fewer mixture components. Algorithm 1 summarizes the entire procedure.

Algorithm 1 (CG Algorithm for Estimating the Mixing Distribution)

```
1: Initialize: $k = 0$; $\mathbf{g}^{(0)} \in \overline{\mathcal{P}}$ such that both $\text{loss}(\mathbf{g}^{(0)})$, $\nabla\text{loss}(\mathbf{g}^{(0)})$ are bounded, and $\boldsymbol{\alpha}^{(0)} = (1)$
2: while stopping condition is not met do
3:     $k \leftarrow k + 1$
4:     Compute $\boldsymbol{f}^{(k)} \in \arg\min_{\mathbf{v}\in\overline{\mathcal{P}}} \langle \nabla\text{loss}(\mathbf{g}^{(k-1)}), \mathbf{v} - \mathbf{g}^{(k-1)} \rangle$   (support-finding step)
5:     Compute $\boldsymbol{\alpha}^{(k)} \in \arg\min_{\boldsymbol{\alpha}\in\Delta_k} \text{loss}(\alpha_0\mathbf{g}^{(0)} + \sum_{s=1}^{k} \alpha_s\boldsymbol{f}^{(s)})$   (proportions-update step)
6:     Update $\mathbf{g}^{(k)} := \alpha_0^{(k)}\mathbf{g}^{(0)} + \sum_{s=1}^{k} \alpha_s^{(k)}\boldsymbol{f}^{(s)}$
7: end while
8: Output: mixture proportions $\alpha_0^{(k)}, \alpha_1^{(k)}, \alpha_2^{(k)}, \ldots, \alpha_k^{(k)}$ and customer types $\mathbf{g}^{(0)}, \boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, \ldots, \boldsymbol{f}^{(k)}$.
```

We discuss a few key features of the algorithm. First, the algorithm outputs both the support and the mixture proportions of the mixing distribution, as desired. Second, the proportions-update step is also a constrained convex optimization problem, but over a much smaller domain compared with $\text{conv}(\overline{\mathcal{P}})$. We show below that this step can be solved efficiently. Third, Algorithm 1 is agnostic to the choice of the loss function loss (so long as it is convex and differentiable) and readily applies to both the NLL and SQ loss functions. Finally, although not the focus in this work, standard errors for the parameters of the recovered mixing distribution (or relevant summary statistics such as the price elasticity) can be computed via bootstrapping, as is commonly done in the literature; see, for instance, Train (2008). Furthermore, as discussed in Section 2, for the negative log-likelihood loss, our method reduces to classical nonparametric maximum-likelihood estimation of the mixing distribution, and therefore inherits its statistical properties. For a detailed discussion on NPMLE, we refer the reader to Lindsay (1995) and references therein.

4.1. Implementation Details

Here, we discuss a few key implementation details for Algorithm 1.

4.1.1. Solving the Support-Finding Step. For each loss function introduced in Section 3, the support-finding step in iteration k can be written as (by plugging in the gradients and dropping constant terms):

$$\text{NLL:}\quad \min_{\boldsymbol{\omega}\in\mathbb{R}^D} -\frac{1}{N}\sum_{t=1}^{T}\sum_{j\in S_t}\left(\frac{N_{jt}}{g_{jt}^{(k-1)}}\right)\cdot\frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_{jt}\right)}{\sum_{\ell\in S_t}\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_{\ell t}\right)}$$

$$\text{SQ:}\quad \min_{\boldsymbol{\omega}\in\mathbb{R}^D} \frac{1}{N}\sum_{t=1}^{T}\sum_{j\in S_t}\left(N_t\cdot g_{jt}^{(k-1)} - N_{jt}\right)\cdot\frac{\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_{jt}\right)}{\sum_{\ell\in S_t}\exp\left(\boldsymbol{\omega}^\top \mathbf{z}_{\ell t}\right)}. \tag{8}$$

The optimal solutions to the problems above may be unbounded. These unbounded solutions correspond to the atomic-likelihood vectors in $\overline{\mathcal{P}}\setminus\mathcal{P}$, as shown in Section 5.2. We solve the problems in (8) approximately using a general-purpose nonlinear program solver (Nocedal and Wright 2006). Solving these optimization problems exactly is computationally hard because they are nonconvex, as shown in Online Appendix D. However, we only need to generate an improving solution—that is, find a feasible descent direction (see description above Equation (6))—to ensure convergence of the algorithm. In our numerical experiments, we found that the standard Broyden–Fletcher–Goldfarb–Shanno (BFGS) method was sufficient to obtain improving solutions.

4.1.2. Solving the Proportions-Update Step. This step is itself a constrained convex program, so we use the CG algorithm to solve it. Instead of the variant described above, we adopt the approach of Krishnan et al. (2015), who recently proposed a modified Frank–Wolfe algorithm to approximately solve the proportions-update step. In contrast to the standard CG algorithm described above, this variant performs two kinds of steps to update the support of the mixing distribution in each iteration: a support-finding step that finds a customer type to be added to the mixture and an "away" step (Guélat and Marcotte 1986) that reduces probability mass (possibly to zero) from a customer type in the existing mixing distribution. Moreover, the support-finding step can be solved exactly by searching over the $k+1$ extreme points of the $(k+1)$-dimensional simplex Δ_k. The next iterate is then computed based on which step—support-finding step or away step—results in higher improvement in the objective value (see algorithm 3 in appendix B of Krishnan et al. 2015 for the details). The presence of away steps means that we can (sometimes) "drop" existing customer types from the mixing distribution, thereby resulting in solutions with fewer numbers of mixture components.

4.1.3. Initialization. We can start with any $\mathbf{g}^{(0)}\in\overline{\mathcal{P}}$ as the initial solution, such that the starting objective $\mathsf{loss}(\mathbf{g}^{(0)})$ and gradient $\nabla\mathsf{loss}(\mathbf{g}^{(0)})$ are bounded. Because we are fitting a mixture of logit models, a natural choice is to fit an LC-MNL model with a "small" number of classes (or even a single-class MNL model) to the data and use that as the initialization. In particular, the MNL log-likelihood objective is globally concave in the parameter $\boldsymbol{\omega}$, and there exists efficient algorithms (Hunter 2004) for its estimation that converge quickly in practice. In our empirical case studies, we initialize by fitting a two-class LC-MNL model to the data.

4.1.4. Stopping Conditions. We can use many stopping conditions to terminate the algorithm: (1) Jaggi (2013) showed that if the subproblem can be solved optimally in each iteration, then we can compute an upper bound on the "optimality gap" of the current solution $\mathbf{g}^{(k)}$—that is, $\mathsf{loss}(\mathbf{g}^{(k)})-\mathsf{loss}(\mathbf{g}^{*})$, where $\mathbf{g}^{*}$ denotes the optimal solution to the Convex Mixture problem. In this case, we can choose an arbitrarily small $\delta>0$ and choose to terminate the algorithm when $\mathsf{loss}(\mathbf{g}^{(k)})-\mathsf{loss}(\mathbf{g}^{*})\le\delta$. However, this might result in overfitting—because of the presence of a large number of mixture components—and, consequently, perform poorly in out-of-sample predictions. (2) We can utilize standard information-theoretic measures proposed in the mixture-modeling literature (McLachlan and Peel 2000), such as the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), and so forth, that capture model complexity as a function of the number of mixture components and prevent overfitting. (3) Finally, a simple way to control for model complexity is to just limit the number of iterations of the algorithm[7] at some $k=K_{\max}$ according to the maximum number of customer types that we may be interested in finding. This ensures that the estimated mixture is composed of at most $K_{\max}$ types, and we use this stopping condition in our empirical case studies.

5. Theoretical Analysis of the Estimator

In this section, we derive the convergence rate of our estimator and also theoretically characterize the customer types recovered by our method.

5.1. Convergence Rate of the Estimator

To state our result on the convergence rate, we need the following notation. For each offer-set S_t, let $\boldsymbol{y}_t \overset{\text{def}}{=} (y_{jt})_{j\in S_t}$ denote the vector of sales fractions for offer-set S_t. Let $H(\boldsymbol{y}_t)\overset{\text{def}}{=}-\sum_{j\in S_t} y_{jt}\log y_{jt}$ denote the entropy of the vector $\boldsymbol{y}_t$. As $0\le y_{jt}\le 1$, $H(\boldsymbol{y}_t)\ge 0$ for all $t\in[T]$.[8] Moreover, let $D_{KL}(p\|q)\overset{\text{def}}{=}p\log(p/q)+(1-p)\log((1-p)/(1-q))$ denote the relative entropy (a.k.a. KL divergence) between p and q for any $0 \le p, q \le 1$. It is a known fact that $D_{KL}(p\|q) \ge 0$ for all $0 \le p, q \le 1$ and $D_{KL}(p\|q) = 0$ if and only if $p = q$. Finally, let $y_{t,\min} \overset{\text{def}}{=} \min\{y_{jt} \mid j \in S_t \text{ s.t. } y_{jt} > 0\}$ and $y_{\min} \overset{\text{def}}{=} \min\{y_{t,\min} \mid t \in [T]\}$. Note that by assumption $y_{t,\min} > 0$ for all $t \in [T]$, and consequently, $y_{\min} > 0$. Then, we can establish the following convergence guarantee:

Theorem 1 (Sublinear Convergence). *Let $\mathbf{g}^*$ denote the optimal solution to the* Convex Mixture *problem and $\mathbf{g}^{(k)}$ denote the kth iterate generated by Algorithm 1. Then, for the loss functions defined in Section 3, it follows*

$$\mathsf{SQ}\left(\mathbf{g}^{(k)}\right) - \mathsf{SQ}(\mathbf{g}^*) \le \frac{4}{k+2} \qquad \textit{for all } k \ge 1,$$

$$\mathsf{NLL}\left(\mathbf{g}^{(k)}\right) - \mathsf{NLL}(\mathbf{g}^*) \le \frac{4}{\xi_{\min}^2 \cdot (k+\kappa)} \qquad \textit{for all } k \ge \bar{K} \textit{ for some constant } \kappa \textit{ and index } \bar{K},$$

where $\xi_{\min}$ is the smallest ξ such that $0 < \xi \le y_{\min}$ and

$$\min_{1\le t\le T} N_t \cdot D_{KL}\left(y_{t,\min}\|\xi\right) \le N \cdot \mathsf{NLL}\left(\mathbf{g}^{(0)}\right) - \sum_{t'=1}^{T} N_{t'} \cdot H\left(\boldsymbol{y}_{t'}\right).$$

Such a $\xi_{\min}$ always exists.

The result above establishes an $O(1/k)$ convergence guarantee for our estimator for both loss functions, assuming that the support-finding step in Algorithm 1 can be solved optimally. The detailed proof is given in Online Appendix A.2; here, we provide a proof sketch. Our proof builds on existing techniques developed for establishing convergence rates of the CG algorithm. This is an active area of research, with different rates having been derived for different variants of the CG algorithm, under different assumptions for the structures of the objective function and the constraint set (Jaggi 2013, Garber and Hazan 2015, Lacoste-Julien and Jaggi 2015). The convergence guarantee for the SQ loss function, in fact, follows directly from the existing result in Jaggi (2013), which shows that the CG algorithm converges at an $O(1/k)$ rate if the so-called *curvature constant* is bounded from above. If the domain is bounded and the Hessian of the objective function is bounded from above, then the curvature constant is known to be bounded from above. In our case, the domain $\text{conv}(\overline{\mathcal{P}})$ is bounded (because any vector $\boldsymbol{f} \in \text{conv}(\overline{\mathcal{P}})$ has entries between 0 and 1). The Hessian of the SQ loss is a diagonal matrix, where each entry is bounded above by 1. Therefore, it follows that the curvature constant is bounded from above, thus allowing us to establish the $O(1/k)$ guarantee by directly invoking existing results.

The Hessian of the NLL loss function, on the other hand, is not bounded from above. For simplicity, suppose that $N_{jt} > 0$ for all $t \in [T]$ and any $j \in S_t$; our result also holds when some of the sales counts are 0; see the discussion at the beginning of Online Appendix A.2.2. Then, it is easy to see that the Hessian is a diagonal matrix with the entry corresponding to (product, offer-set) pair (j, S_t) equal to $N_{jt}/(N \cdot \mathrm{g}_{jt}^2)$. Because g_{jt} can be arbitrarily close to 0 in the domain $\text{conv}(\overline{\mathcal{P}})$, the diagonal entries are not bounded from above, and, thus, existing results don't directly apply. To address this issue, suppose that we can establish a nontrivial lower bound, say, $\xi^* > 0$, for the optimal solution $\mathbf{g}^*$ so that $\mathrm{g}_{jt}^* \ge \xi^* > 0$ for all $t \in [T]$ and all $j \in S_t$. It then follows that the Hessian of the NLL loss is bounded from above when the domain is restricted to $\tilde{\mathcal{D}} \overset{\text{def}}{=} \{\mathbf{g} \in \text{conv}(\overline{\mathcal{P}}) : \mathrm{g}_{jt} \ge \xi^* \; \forall t \in [T] \; \forall j \in S_t\}$. And, if we solve Convex Mixture over the restricted domain $\tilde{\mathcal{D}}$,[9] we immediately obtain the $O(1/k)$ convergence rate.

Although solving Convex Mixture over the restricted domain $\tilde{\mathcal{D}}$ is feasible in principle, it is difficult to implement in practice because computing a good lower bound ξ^* may not be straightforward. Instead, we show that running the fully corrective variant of the CG algorithm (the variant implemented in Algorithm 1), although being agnostic to a lower bound, still converges at $O(1/k)$ rate. For that, we first show that each iterate $\mathbf{g}^{(k)}$ generated by Algorithm 1 is bounded from below by $\xi_{\min}$, where $\xi_{\min}$ is as defined in Theorem 1. Then, we exploit this property to establish the $O(1/k)$ convergence rate with the constant scaling in $1/\xi_{\min}^2$.

To get the best convergence rate, we need to use the tightest lower bound $\xi_{\min}$. Our bound is derived for general cases, and in this generality, the bound is tight. To see that, consider the setting when the observations consist of only market shares, so that $T = 1$, $S_1 = [n]$, and the sales fractions $\boldsymbol{y}_1$ comprise the observed market shares. In this case, it can be shown that the optimal solution $\mathbf{g}^* = \boldsymbol{y}_1$.[10] When Algorithm 1 is initialized at $\mathbf{g}^{(0)} = \boldsymbol{y}_1$, it follows from the definition that $\xi_{\min} = y_{\min} = y_{1,\min}$, which is the tightest bound possible.

We can also derive a simple-to-compute (lower) bound for $\xi_{\min}$, as stated in the following proposition:

Proposition 1. *Let $N_{\min} = \min\{N_{jt} \mid t \in [T]; j \in S_t \text{ s.t.} N_{jt} > 0\}$. Then, it follows that*

$$y_{\min} \ge \xi_{\min} \ge y_{\min} \cdot \exp\left(-1 - \frac{N \cdot \mathsf{NLL}\left(\mathbf{g}^{(0)}\right) - \sum_{t=1}^{T} N_t \cdot H\left(\boldsymbol{y}_t\right)}{N_{\min}}\right).$$

When $T = 1$, $S_1 = [n]$, and $\mathbf{g}^{(0)} = \boldsymbol{y}_1$, it follows from the above proposition that $y_{\min} \ge \xi_{\min} \ge y_{\min}/e$. Therefore, the simple-to-compute (lower) bound loses a factor of e in this case.

Remark. Theorem 1 assumes that the support-finding step in Algorithm 1 can be solved optimally.[11] In cases where the optimal solution cannot be found, a weaker convergence guarantee can be established, as long as the iterates are (sufficiently) improving, that is, $\mathsf{loss}(\mathbf{g}^{(k)}) < \mathsf{loss}(\mathbf{g}^{(k-1)})$, for each iteration k. In this case, it follows from existing results (see, e.g., Zangwill 1969) that the sequence of iterates converges to a stationary point, which in the case of a convex program is an optimal solution.

5.2. Characterization of the Recovered Mixture Types

We now focus on the support-finding step and characterize the structure of the optimal solution. These solutions comprise the support of the resulting mixing distribution. In each iteration k, the support-finding step is equivalent to solving the following problem (by dropping constant terms):

$$\min_{\boldsymbol{f} \in \overline{\mathcal{P}}} \sum_{t=1}^{T} \sum_{j \in S_t} c_{jt}^{(k)} f_{jt},$$

where $c_{jt}^{(k)} = \left(\nabla \mathsf{loss}(\mathbf{g}^{(k-1)})\right)_{jt}$. The optimal solution $\boldsymbol{f}^{(k)}$ to the above problem lies either in $\mathcal{P}$ or $\overline{\mathcal{P}} \setminus \mathcal{P}$. If it lies in $\mathcal{P}$, then (by definition) there exists a parameter vector $\boldsymbol{\omega}_k \in \mathbb{R}^D$ such that $\boldsymbol{f}(\boldsymbol{\omega}_k) = \boldsymbol{f}^{(k)}$, so that any such $\boldsymbol{\omega}_k$ may be used to describe the customer type and make the choice probability prediction $e^{\boldsymbol{\omega}_k^\top \mathbf{z}_{jS}} / \left(\sum_{\ell \in S} e^{\boldsymbol{\omega}_k^\top \mathbf{z}_{\ell S}}\right)$ for the probability of choosing product j from some offer-set S. However, if the optimal solution $\boldsymbol{f}^{(k)}$ lies in the boundary—that is, $\overline{\mathcal{P}} \setminus \mathcal{P}$—then there is no straightforward way to characterize the customer type or make out-of-sample predictions. To deal with this challenge, we provide a compact characterization of what we call the *boundary types*, defined as follows:

Definition 1 (Boundary and Nonboundary Types). A customer type $\boldsymbol{f}$ is called a boundary type if $\boldsymbol{f} \in \overline{\mathcal{P}} \setminus \mathcal{P}$ and a nonboundary type otherwise.

We show below that each boundary type is characterized by two parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$:

Theorem 2 (Characterization of Boundary Types). *Given a boundary type $\boldsymbol{f}$ in $\overline{\mathcal{P}} \setminus \mathcal{P}$, there exist parameters $\boldsymbol{\omega}_0, \boldsymbol{\theta} \in \mathbb{R}^D$ such that, for each $1 \le t \le T$ and $j \in S_t$, we have*

$$f_{jt} = \lim_{r \to \infty} \frac{\exp\left((\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top \mathbf{z}_{jt}\right)}{\sum_{\ell \in S_t} \exp\left((\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top \mathbf{z}_{\ell t}\right)}.$$

Furthermore, $f_{jt} = 0$ for at least one (product, offer-set) pair (j, S_t).

The proof in Online Appendix A.3 shows how to compute the parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ given any boundary type $\boldsymbol{f} \in \overline{\mathcal{P}} \setminus \mathcal{P}$. Here, we focus on understanding the implications of the above characterization.

First, because for any boundary type $\boldsymbol{f}$, $f_{jt} = 0$ for at least one (product, offer-set) pair (j, S_t), there exists *no* logit parameter vector $\boldsymbol{\omega}$ such that $f_{jt} = e^{\boldsymbol{\omega}^\top \mathbf{z}_{jt}} / \left(\sum_{\ell \in S_t} e^{\boldsymbol{\omega}^\top \mathbf{z}_{\ell t}}\right)$ for all j, S_t. Second, boundary types arise as a result of limiting logit models, obtained as the parameter vector $\boldsymbol{\omega}$ is pushed to infinity. In particular, Theorem 2 states that for any boundary type $\boldsymbol{f}$, there exists parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ such that the choice probabilities for observed (product, offer-set) pairs under $\boldsymbol{f}$ are equal to those under the limiting type $\lim_{r \to \infty} \boldsymbol{f}(\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})$, where recall that $\boldsymbol{f}(\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})$ corresponds to the customer type with logit parameter $\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta}$. Below, we discuss this characterization in more detail.

The key aspect of our characterization is a preference ordering over the products defined by the parameter vector $\boldsymbol{\theta}$. This preference order determines the choice of the products from a given offer-set. For ease of exposition, we describe the preference ordering for the case when product features don't change with the offer-set, so we write $\mathbf{z}_j$ instead of $\mathbf{z}_{jt}$ for the feature vector of product j. The discussion below extends immediately to the more general case by associating a separate product to each feature vector of interest. To describe the preference order, define product utility $u_j \overset{\text{def}}{=} \boldsymbol{\theta}^\top \mathbf{z}_j$ for product j. These utility values can be visualized as projections of the product feature vectors $\mathbf{z}_j$'s onto the vector $\boldsymbol{\theta}$. They define a preference order $\succeq$ among the products such that $j \succeq j'$, read as "product j is weakly preferred over product j'," if and only if $u_j \ge u_{j'}$. The relation $\succeq$ is in general a weak ordering and *not* a strict ordering because product utilities may be equal. In order to explicitly capture indifferences, we write $j \succ j'$ if $u_j > u_{j'}$ and $j \sim j'$ if $u_j = u_{j'}$.

Now, when offered a set S, customers of this type purchase only the most preferred products as determined according to the preference order $\succeq$. To see that, let $C(S)$ denote the set of most preferred products in S, so that for all $j \in C(S)$, we have $j \sim \ell$ if $\ell \in C(S)$ and $j \succ \ell$ if $\ell \in S \setminus C(S)$. Let $u^* \overset{\text{def}}{=} \max\{u_j : j \in S\}$ denote the maximum utility among the products in S. We have that $u^* = u_j$ for all $j \in C(S)$ and $u^* > u_j$ for all $j \in S \setminus C(S)$. Given this and multiplying the numerator and denominator of the choice probabilities defined in Theorem 2 by $e^{-r \cdot u^*}$, we can write for any $j \in S$,

$$\frac{\exp\left((\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top \mathbf{z}_j\right)}{\sum_{\ell \in S} \exp\left((\boldsymbol{\omega}_0 + r \cdot \boldsymbol{\theta})^\top \mathbf{z}_\ell\right)} = \frac{e^{-r \cdot (u^* - u_j)} \cdot \exp\left(\boldsymbol{\omega}_0^\top \mathbf{z}_j\right)}{\sum_{\ell \in C(S)} \exp\left(\boldsymbol{\omega}_0^\top \mathbf{z}_\ell\right) + \sum_{\ell \in S \setminus C(S)} e^{-r \cdot (u^* - u_\ell)} \cdot \exp\left(\boldsymbol{\omega}_0^\top \mathbf{z}_\ell\right)}. \tag{9}$$

When we take the limit as $r \to \infty$, each of the terms $e^{-r \cdot (u_\ell - u^*)}$, $\ell \in S \setminus C(S)$, goes to zero, so the denominator converges to $\sum_{\ell \in C(S)} \exp(\boldsymbol{\omega}_0^\top \mathbf{z}_\ell)$. The numerator converges to $\exp(\boldsymbol{\omega}_0^\top \mathbf{z}_j)$ if $j \in C(S)$ and 0 if $j \in S \setminus C(S)$. Therefore, we obtain the following choice probability prediction $f_{j,S}(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ for any product j and offer-set S from Theorem 2:

$$f_{j,S}(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = \begin{cases} \exp(\boldsymbol{\omega}_0^\top \mathbf{z}_j)/\left(\sum_{\ell \in C(S)} \exp(\boldsymbol{\omega}_0^\top \mathbf{z}_\ell)\right), & \text{if } j \in C(S) \text{ and} \\ 0, & \text{if } j \in S \setminus C(S). \end{cases}$$

From the discussion above, we note the contrasting roles of the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\omega}_0$. The parameter vector $\boldsymbol{\theta}$ (through the preference ordering $\geq$ it induces) determines the *consideration set* $C(S)$, whereas the parameter vector $\boldsymbol{\omega}_0$ determines the logit choice probabilities from within the consideration set. The parameter vector $\boldsymbol{\theta}$ dictates how a product's features impact its inclusion into the consideration set. For instance, suppose u^* is the maximum utility in offer-set S and product j with utility $u_j < u^*$ is not in consideration now. Further, suppose one of the features is price and the corresponding coefficient is $\theta_p < 0$. Then, product j will enter into consideration only if its price is sufficiently dropped to make its utility greater than or equal to u^*. In other words, the price should be dropped by at least $(u^* - u_j)/|\theta_p|$ to ensure consideration of product j.

The choice behavior we identify for the boundary types is consistent with existing literature, which establishes that customers often consider a subset of the products on offer before making the choice (Jagabathula and Rusmevichientong 2016). In fact, consideration sets of the kind we identify are a special case of the linear compensatory decision rule that has been used as a heuristic for forming consideration sets in existing literature (Hauser 2014). The rule computes the utility for each product as a weighted sum of the feature values and chooses all products that have a utility greater than a cutoff to be part of the consideration set. Finally, multiple distinct tuples of parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ can result in the same limiting choice probabilities f for the observed data. Because the data do not provide any further guidance, we arbitrarily select one of them. Studying the impact of different selection rules on the prediction accuracy is a promising avenue for future work.

We conclude this subsection with the following systematic procedure that summarizes our discussion for making out-of-sample choice predictions for a boundary type:

Algorithm 2 (Predicting Choice Probabilities for Boundary Type $f(\boldsymbol{\omega}_0, \boldsymbol{\theta})$)

```
1: Input: Offer-set $S$ with product features $\mathbf{z}_{jS} \in \mathbb{R}^D$ for each $j \in S$
2: Compute utilities $u_j = \boldsymbol{\theta}^\top \mathbf{z}_{jS}$ for each $j \in S$.
3: Form consideration set $C(S) = \{j \in S \mid u_j = \max_{\ell \in S} u_\ell\}$
4: For any $j \notin C(S)$, set $f_{j,S}(\boldsymbol{\omega}_0, \boldsymbol{\theta}) \leftarrow 0$
5: For any $j \in C(S)$, set
       $f_{j,S}(\boldsymbol{\omega}_0, \boldsymbol{\theta}) \leftarrow \frac{\exp(\boldsymbol{\omega}_0^\top \mathbf{z}_{jS})}{\sum_{\ell \in C(S)} \exp(\boldsymbol{\omega}_0^\top \mathbf{z}_{\ell S})}$
6: Output: Choice probabilities $\{f_{j,S}(\boldsymbol{\omega}_0, \boldsymbol{\theta}) : j \in S\}$.
```

5.3. Analysis of Recovered Distribution for Two Special Cases

We now analyze scenarios under which the optimal solution to the support-finding step is indeed a boundary type. This helps in providing further insights into the structure of the recovered mixing distribution. Solving the support-finding step in the general case is a hard problem, and, therefore, to keep the analysis tractable, we focus on the setting in which the data consist of sales counts in a single time period when all products are offered (such as market-shares data). For this case, the notation can be simplified.

Because there is only a single offer-set $S_1 = [n]$, we represent the features as $\mathbf{z}_i$ for each product $i \in [n]$. Further, the sales counts can be represented by using a single vector $\boldsymbol{y} := (y_1, y_2, \ldots, y_n) \in [0,1]^n$ such that $\sum_{i=1}^n y_i = 1$, where $y_i \geq 0$ is the fraction of sales for product i. The choice probabilities $\boldsymbol{f} \in \overline{\mathcal{P}}$ are of the form $\boldsymbol{f} = (f_1, f_2, \ldots, f_n)$, also satisfying $\sum_{i=1}^n f_i = 1$. Similarly, the estimates produced by Algorithm 1 at any iteration k are of the form $\mathbf{g}^{(k)} = (\mathrm{g}_1^{(k)}, \mathrm{g}_2^{(k)}, \ldots, \mathrm{g}_n^{(k)})$, where again $\sum_{i=1}^n \mathrm{g}_i^{(k)} = 1$. With this notation, the loss functions defined in Section 3 can be written as

$$\mathsf{NLL}(\mathbf{g}) = -\sum_{i=1}^n y_i \log(\mathrm{g}_i); \quad \mathsf{SQ}(\mathbf{g}) = \frac{1}{2}\sum_{i=1}^n (y_i - \mathrm{g}_i)^2, \tag{10}$$

while the support-finding step is of the form, with $c_i \overset{\text{def}}{=} -\left(\nabla \mathsf{loss}(\mathbf{g}^{(k-1)})\right)_i$ for each $i \in [n]$ (we switch to maximization to aid the analysis below):

$$\max_{f \in \overline{\mathcal{P}}} \sum_{i=1}^n c_i \cdot f_i, \tag{11}$$

where we drop the explicit dependence of the coefficient c_i on the iteration number k for simplicity of notation. We analyze the optimal solution to the above subproblem under two cases: (1) All product features are continuous; and (2) some product features are binary.[12]

5.3.1. All Product Features Are Continuous.

When all features are continuous, the optimal solution to subproblem (11) depends on the geometric structure of the observed product features. Specifically, we consider the (convex) polytope formed by the convex hull of the product features $\mathbf{z}_1, \ldots, \mathbf{z}_n$, denoted as $\mathcal{Z}_n \overset{\text{def}}{=} \mathrm{conv}(\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\})$. For this polytope, we define an extreme point as follows.

Definition 2 (Extreme Points). $\mathbf{z}_j$ is called an *extreme point* of the convex polytope $\mathscr{Z}_n$ if $\mathbf{z}_j \notin \text{conv}(\{\mathbf{z}_i \, : \, i \neq j, 1 \leq i \leq n\})$. Equivalently, extreme points correspond to vertices of $\mathscr{Z}_n$.

With this definition, we can establish conditions under which a boundary type is an optimal solution to the support-finding step (11). In particular, we have the following result:

Theorem 3 (Recovery of Boundary Types). *Suppose we observe sales data for only the offer-set* $[n]$*. Let* $j_{\max} = \arg\max_{j\in[n]} c_j$*. If* $\mathbf{z}_{j_{\max}}$ *is an extreme point of the polytope* $\mathscr{Z}_n$*, then the boundary type* $\boldsymbol{f}(\mathbf{0}, \boldsymbol{\theta}_{j_{\max}})$ *is an optimal solution to support-finding step* (11)*, where* $\boldsymbol{\theta}_{j_{\max}}$ *is such that* $\boldsymbol{\theta}_{j_{\max}}^\top \mathbf{z}_{j_{\max}} > \boldsymbol{\theta}_{j_{\max}}^\top \mathbf{z}_j$ *for all* $j \neq j_{\max}$*. In particular,* $\boldsymbol{f}(\mathbf{0}, \boldsymbol{\theta}_{j_{\max}})$ *is of the form*

$$f_j(\mathbf{0}, \boldsymbol{\theta}_{j_{\max}}) = \begin{cases} 1 & \text{if } j = j_{\max} \\ 0 & \textit{otherwise.} \end{cases}$$

The proof in Online Appendix A.4 shows the existence of such a $\boldsymbol{\theta}_{j_{\max}}$. The above result shows that our estimation method recovers boundary types that consider only a single product amongst the offered products. The result also leads to the following corollary:

Corollary 1 (All Extreme Points $\Longrightarrow$ Boundary Types Always Optimal). *Suppose we observe sales data for only the offer-set* $[n]$*. If all feature vectors* $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$ *are extreme points of the polytope* $\mathscr{Z}_n$*, then boundary types are always optimal solutions for the support-finding step* (11).

The above result implies that when all product features are extreme points of the polytope $\mathscr{Z}_n$, the support-finding step (11) can be solved to optimality in each iteration, where the optimal solution corresponds to a boundary type that chooses a single product with probability 1 from amongst all offered products. Consequently, our estimation method decomposes the population into such boundary types to explain the observed-choice data. In fact, in this scenario, we can also establish the following convergence guarantee for the iterates generated by Algorithm 1:

Theorem 4 (Convergence in Finite Number of Iterations). *Suppose we observe sales data for only the offer-set* $[n]$*. Further, suppose that* $\mathbf{z}_j$ *is an extreme point of the polytope* $\mathscr{Z}_n$ *for all* $j \in [n]$*. For both the* NLL *and* SQ *loss functions defined in* (10)*, the estimates* $\mathbf{g}^{(k)}$ *produced by Algorithm 1 converge to the optimal solution* $\mathbf{g}^*$ *in at most n iterations. In particular, the optimal solution* $\mathbf{g}^* = \boldsymbol{y}$ *and, consequently, the CG algorithm is able to perfectly match the observed sales fractions.*

Because of the complexity of the resulting optimization problems, there are few convergence guarantees for the estimation of logit models that exist in the literature. For instance, Hunter (2004) presents necessary and sufficient conditions for an iterative minorization–maximization (MM) algorithm to converge to the maximum-likelihood estimate for a *single-class* MNL model. Recently, James (2017) proposed an MM algorithm for estimation of mixed logit models with a multivariate normal mixing distribution, but did not provide any conditions for convergence. To the best of our knowledge, our result is one of the first to provide a convergence guarantee for general mixtures of logit models.

5.3.2. Some (or All) Product Features Are Binary. Next, we consider the case when some of the features are binary. To state the result, we need to introduce additional notation. For each product $\ell \in [n]$, let $\mathbf{z}_\ell \in \mathbb{R}^{D_1}$ and $\mathbf{b}_\ell \in \{0,1\}^{D_2}$ represent a set of continuous and binary features, respectively, where $D_1 + D_2 = D$ and $D_2 > 0$. Define the binary relation $\sim$ on $[n]$ as: $i \sim j \Longleftrightarrow \mathbf{b}_i = \mathbf{b}_j$. It is easy to see that $\sim$ is an equivalence relation on $[n]$, and therefore let $\mathscr{E}$ represent the equivalence classes—that is, $[n] = \bigcup_{e\in\mathscr{E}} S_e$ and $S_{e_1} \cap S_{e_2} = \emptyset$ for all $e_1, e_2 \in \mathscr{E}$ such that $e_1 \neq e_2$.

With the above notation, we can show that the optimal solution to the support-finding step always corresponds to a boundary type, having the following structure:

Theorem 5 (Binary Feature $\Longrightarrow$ Boundary Types Are Always Optimal). *Suppose we observe sales data for only the offer-set* $[n]$*. Then, there exists* $e^* \in \mathscr{E}$ *such that the optimal solution to support-finding step* (11) *is a boundary type* $\boldsymbol{f}(\boldsymbol{\omega}_0, \boldsymbol{\theta})$ *with* $\boldsymbol{\theta} \in \mathbb{R}^D$ *satisfying*

$$\boldsymbol{\theta}^\top(\mathbf{z}_j \circ \mathbf{b}_j) > \boldsymbol{\theta}^\top(\mathbf{z}_i \circ \mathbf{b}_i) \;\; \forall\, j \in S_{e^*};\; \forall\, i \in [n] \setminus S_{e^*},$$

where $\circ$ *denotes vector concatenation. In particular,* $f_i(\boldsymbol{\omega}_0, \boldsymbol{\theta}) = 0$ *for all* $i \in [n] \setminus S_{e^*}$ *so that the boundary type only considers products within subset* $S_{e^*} \subset [n]$.

Theorem 5 establishes that if products have certain binary features, then the support-finding step (11) always has a boundary type as the optimal solution. The consideration sets of the resulting types follow a conjunctive decision rule (Hauser 2014), where customers screen products with a set of "must have" or "must not have" aspects—corresponding to each binary attribute—reflecting (strong) noncompensatory preferences. We can interpret the above result in the context of our sushi case study (see Section 7.1), where the products represent two different kinds of sushi varieties—*maki* and *nonmaki*. The above result says that we recover boundary types in each iteration, each of which only consider one kind of sushi variety: either maki or nonmaki. Note that the mixing distribution can contain more than one boundary type with the same consideration set, as the types will be differentiated in their choice behavior according to the parameters $(\boldsymbol{\omega}_0, \boldsymbol{\theta})$. In particular, based on the value of the parameter $\boldsymbol{\theta}$, even some products within subset S_{e^*} may *not* be considered by the boundary type. We analyze the structure of the recovered mixing distribution in more detail in the case study.

5.4. Heuristic Approaches Based on Above Theoretical Results for Solving the Support-Finding Step in the General Case

Characterizing the optimal solution to the support-finding step in the general case is hard because the structure of the optimal solution is governed by the particular values of the coefficients $\{c_{jt}^{(k)}\}_{j,t}$ at any iteration k—which themselves are dependent on the initial solution $\mathbf{g}^{(0)}$—and the product feature variations within each offer-set. Nevertheless, our theoretical results for the characterization of boundary types (Section 5.2) as well as the single offer-set case (Section 5.3) inform the design of some heuristics for solving the support-finding step in more general scenarios:

- *Single offer-set with all continuous features.* If the condition in Theorem 3 is satisfied, we know that the optimal solution to the support-finding step is a boundary type. Otherwise, the optimal solution can be a boundary or nonboundary type. If we define $\mathcal{X}_{n,\text{ext}} \overset{\text{def}}{=} \{j \in [n] \mid \mathbf{z}_j \text{ is an extreme point of } \mathcal{X}_n\}$, then it follows that $\boldsymbol{f}(\mathbf{0}, \boldsymbol{\theta}_j) \in \overline{\mathcal{P}}$ for all $j \in \mathcal{X}_{n,\text{ext}}$, where $\boldsymbol{\theta}_j$ is as defined in Theorem 3. Consequently, the set of such boundary types, $\mathcal{B} = \{\boldsymbol{f}(\mathbf{0}, \boldsymbol{\theta}_j) \mid j \in \mathcal{X}_{n,\text{ext}}\}$, provides a feasible search set for subproblem (11). We can also determine a nonboundary type, say, $\boldsymbol{f}(\boldsymbol{\omega}^{(k)})$, as an approximate solution, based on the discussion in Section 4.1.1. Then, we can output the type that achieves the best objective as an approximate solution to (11)—that is, we output: $\arg\max_{f \in \mathcal{B} \,\cup\, \{f(\boldsymbol{\omega}^{(k)})\}} \sum_{i=1}^{n} c_i \cdot f_i$. We employ a heuristic based on this approach to solve the support-finding step in our sushi case study in Section 7.1.
- *Multiple offer-sets.* Here, again, the optimal solution can either be a boundary type or a nonboundary type. Similar to the above scenario, we can determine a nonboundary type as an approximate solution to the support-finding step. In our numerical experiments, we observed, in some cases, that the (nonboundary) type output by the BFGS method was assigning very "small" probabilities to some (product, offer-set) pairs. This could be an indication that the optimal solution is actually a boundary type. Motivated by this, we can design a procedure that performs a post hoc analysis on the type output by the BFGS method, to determine a boundary type as a candidate solution to the support-finding step. We then output the customer type that achieves the better objective amongst the two as an approximate solution to the support-finding step; refer to Online Appendix B.3 for an illustration of this procedure for our case study on the IRI Academic Data Set (Section 7.2).

6. Robustness to Different Ground-Truth Mixing Distributions

In this section, we use a simulation study to showcase the ability of our nonparametric estimator to obtain good approximations to various underlying mixing distributions. Our method uses only the transaction data and has no prior knowledge of the structure of the ground-truth distribution. We compare the mixing distribution estimated by our method to the one estimated by a standard random parameters logit (RPL) benchmark, which makes the static assumption that the underlying mixing distribution is multivariate normal. Our results demonstrate the cost of model misspecification—the parametric RPL benchmark yields significantly poor approximations to the ground-truth mixing distributions. On the other hand, our nonparametric method is able to automatically learn from the transaction data to construct a good approximation. This specific property of our estimator makes it very appealing in practice, where one has little knowledge of the ground-truth mixing distribution.

6.1. Setup

So that we can readily compare our method to existing methods, we borrow the experimental setup from Fox et al. (2011) for our simulation study. Fox et al. (2011) propose a nonparametric linear regression-based estimator for recovering the mixing distribution. The key distinction from our method is that they require knowledge of the support of the mixing distribution, but our method does not. We discuss the implications of this difference toward the end of this section.

The universe consists of $n = 11$ products, one of which is the no-purchase or the outside option. The firm offers all the products in the universe to the customers, but customizes the product features, offering product j with feature vector $\mathbf{z}_{jt} = (z_{jt1}, z_{jt2})$ in period t. We assume that the outside option is represented by the all-zeros feature vector $(0, 0)$. Customers make choices according to a mixture of logit model with ground-truth mixing distribution Q. In each time period t, a customer makes a single choice by first sampling an MNL parameter vector $\boldsymbol{\omega}^{(t)} = (\omega_1^{(t)}, \omega_2^{(t)})$ and then choosing product j with probability

$$f_{jt}\left(\boldsymbol{\omega}^{(t)}\right) = \frac{\exp\left(\omega_1^{(t)} \cdot z_{jt1} + \omega_2^{(t)} \cdot z_{jt2}\right)}{\sum_{\ell \in [n]} \exp\left(\omega_1^{(t)} \cdot z_{\ell t1} + \omega_2^{(t)} \cdot z_{\ell t2}\right)}.$$

We consider three underlying ground-truth distributions Q:

1. Mixture of two bivariate Gaussians: $Q^{(2)} = 0.4 \cdot \mathcal{N}([3,-1], \mathbf{\Sigma}_1) + 0.6 \cdot \mathcal{N}([-1,1], \mathbf{\Sigma}_2)$;
2. Mixture of four bivariate Gaussians:

$$Q^{(4)} = 0.2 \cdot \mathcal{N}([3,0], \mathbf{\Sigma}_1) + 0.4 \cdot \mathcal{N}([0,3], \mathbf{\Sigma}_1) + 0.3 \cdot \mathcal{N}([1,-1], \mathbf{\Sigma}_2) + 0.1 \cdot \mathcal{N}([-1,1], \mathbf{\Sigma}_2);$$

3. Mixture of six bivariate Gaussians:

$$Q^{(6)} = 0.1 \cdot \mathcal{N}([3,0], \mathbf{\Sigma}_1) + 0.2 \cdot \mathcal{N}([0,3], \mathbf{\Sigma}_1) + 0.2 \cdot \mathcal{N}([1,-1], \mathbf{\Sigma}_1) + 0.1 \cdot \mathcal{N}([-1,1], \mathbf{\Sigma}_2) + 0.3 \cdot \mathcal{N}([2,1], \mathbf{\Sigma}_2) + 0.1 \cdot \mathcal{N}([1,2], \mathbf{\Sigma}_2),$$

where $\mathbf{\Sigma}_1 = \begin{bmatrix} 0.2 & -0.1 \\ -0.1 & 0.4 \end{bmatrix}$ and $\mathbf{\Sigma}_2 = \begin{bmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \end{bmatrix}$ denote the variance–covariance matrices of the component Gaussian distributions.

We generated nine instances by varying the ground-truth mixing distribution Q over the set $\{Q^{(2)}, Q^{(4)}, Q^{(6)}\}$ and the number of time periods T over the set {2,000; 5,000; 10,000}. For each combination of Q and T and time period $t \in [T]$, we generate choice data as follows: (a) We sample product features z_{jtd} according to the distribution $\mathcal{N}(0, 1.5^2)$ independently for all products $j \in [n]$, except the no-purchase option, and for all features $d \in \{1,2\}$; (b) we sample a logit parameter vector $\boldsymbol{\omega}^{(t)}$ from the ground-truth mixing distribution Q; and, then, (c) we generate a single choice $j \in [n]$ with probability $f_{jt}(\boldsymbol{\omega}^{(t)})$. Note that there is a single choice observation $N_t = 1$ in each time period $t \in [T]$. We replicate the above process $R = 50$ times. For each replication $r \in [R]$, we obtain mixture cumulative distribution functions (CDFs) $\hat{F}_r^{RPL}$ and $\hat{F}_r^{CG}$ by fitting the standard RPL model with a bivariate normal (having nonzero correlation) mixing distribution[13] and optimizing the NLL loss using our CG algorithm, respectively. To assess the goodness of fit, we use the following two metrics proposed by Fox et al. (2011): the root mean integrated squared error (RMISE) and the mean integrated absolute error (MIAE), defined as

$$\text{RMISE} = \sqrt{\frac{1}{R}\sum_{r=1}^{R}\left[\frac{1}{V}\sum_{v=1}^{V}\left(\hat{F}_r(\boldsymbol{\beta}_v) - F_0(\boldsymbol{\beta}_v)\right)^2\right]} \quad \text{and}$$

$$\text{MIAE} = \frac{1}{V \cdot R}\sum_{r=1}^{R}\sum_{v=1}^{V}\left|\hat{F}_r(\boldsymbol{\beta}_v) - F_0(\boldsymbol{\beta}_v)\right|,$$

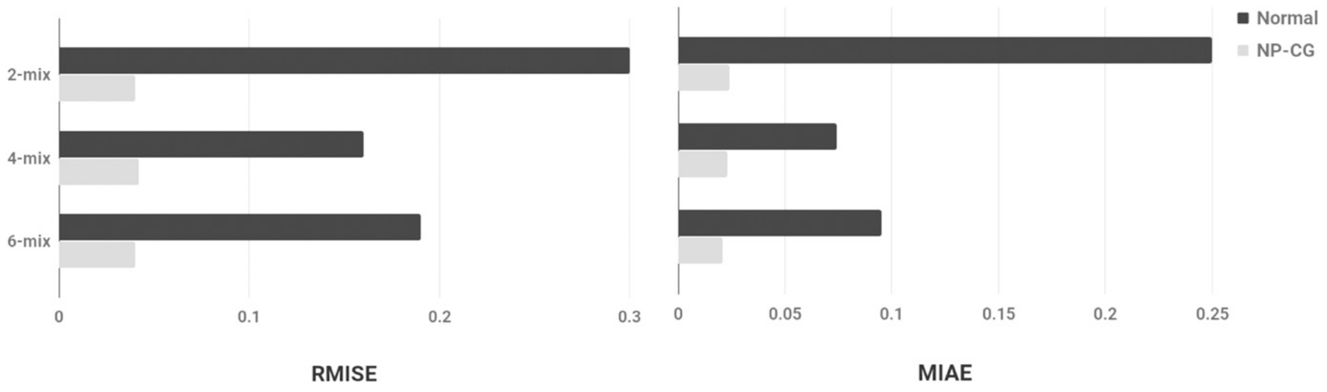
where $\hat{F}_r \in \{\hat{F}_r^{RPL}, \hat{F}_r^{CG}\}$, $\boldsymbol{\beta}_v$'s represent $V = 10^4$ uniformly spaced points in the rectangle $[-6,6] \times [-6,6]$ where the CDF is evaluated[14] and F_0 is the CDF of the ground-truth mixing distribution Q.

6.2. Results and Discussion

Figure 1 and Table 1 summarize the results we obtained when we ran our estimator for $K_{\max} = 81$ iterations.[15] Figure 1 shows a bar graph comparing our method to the RPL model on the RMISE and MIAE metrics. These metrics are compared for the three ground-truth mixing distributions, for the case with $T = 10,000$ periods. Table 1 shows a more complete comparison, including for the cases with $T = 2,000$ and $T = 5,000$ periods. We make the following observations:

1. Our nonparametric method is able to automatically construct a good approximation of the ground-truth mixing distribution Q from the transaction data, without any prior knowledge of the structure of Q. The benchmark RPL model, on the other hand, performs significantly worse because of model misspecification.
2. Table 1 shows that our estimator becomes better as the number of periods (and, correspondingly, the samples) T increases. This improvement, which is characteristic of nonparametric estimators, shows that our method is able to extract more information

Figure 1. Error Metrics for the Different Ground-Truth Mixing Distributions (T = 10,000 Periods)



Notes. "Normal" and "NP-CG" refer to the RPL model with a bivariate normal mixing distribution and our nonparametric CG-based estimator, respectively. The labels 2-mix, 4-mix, and 6-mix refer, respectively, to the ground-truth mixing distributions $Q^{(2)}$, $Q^{(4)}$, and $Q^{(6)}$, described in the main text. Lower values for the error metrics are preferred.

Table 1. Error Metrics for the Different Ground-Truth Mixing Distributions as a Function of the Number of Periods T

	RMISE						MIAE					
	2-mix		4-mix		6-mix		2-mix		4-mix		6-mix	
T	Normal*	NP-CG	Normal	NP-CG	Normal	NP-CG	Normal	NP-CG	Normal	NP-CG	Normal	NP-CG
2,000	0.29	0.067	0.15	0.067	0.18	0.066	0.24	0.039	0.082	0.037	0.094	0.035
5,000	0.3	0.053	0.15	0.051	0.18	0.053	0.25	0.03	0.078	0.0289	0.094	0.028
10,000	0.3	0.04	0.16	0.042	0.19	0.04	0.25	0.024	0.074	0.023	0.095	0.021

Notes. The labels 2-mix, 4-mix, and 6-mix refer, respectively, to the ground-truth mixing distributions $Q^{(2)}$, $Q^{(4)}$, and $Q^{(6)}$, described in the main text. Lower values for the error metrics are preferred. Normal, RPL model with a bivariate normal mixing distribution; NP-CG, our nonparametric CG-based estimator.

*The metrics for the RPL model with a bivariate normal mixing distribution (referred to as "Normal") are taken from table 3 in Fox et al. (2011); we obtained similar numbers in our implementations.

as more data are made available. The RPL model, by contrast, does not exhibit any such consistent pattern.

3. Although not shown in Table 1, we note that the errors metrics reported by Fox et al. (2011, tables 1 and 2) for their method are comparable (or slightly worse) to those obtained under our method. Their method, however, needs the support of the mixing distribution as input. For their experiments under the simulation setup above, they use a uniform discrete grid as the support of the mixing distribution. This approach, however, does not scale to high-dimensional settings with larger D values. Our estimator does not suffer from this limitation—we show that it scales to the feature dimensions in real-world case studies, with $D = 5$ (Section 7.1) and $D = 11$ (Section 7.2).

7. Predictive Performance of the Estimator

We perform two numerical studies on real-world data to showcase the predictive accuracy of our method. The first case study uses market-share data, whereas the second study applies our estimation technique on sales-transaction data from multiple stores with varying offer-sets and product prices.

7.1. Case Study 1: SUSHI Preference Data Set

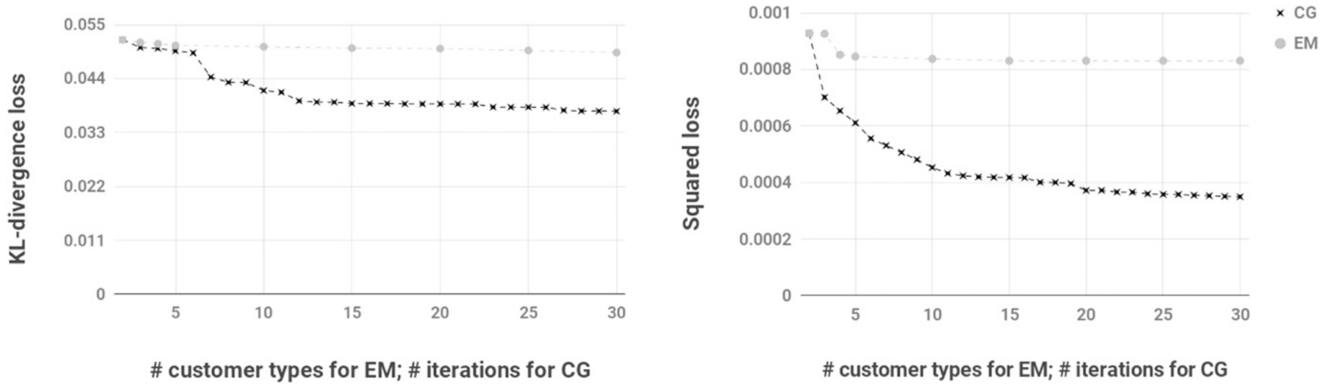
In this study, we compare our CG method with the expectation-maximization benchmark (EM) on in-sample fit and predictive and decision accuracies. We use the SUSHI Preference Data Set (Kamishima et al. 2005) for our study. This data set has been used extensively in prior work on learning customer preferences. It consists of the preferences of 5,000 customers over 100 varieties of sushi. Each customer provides a rank ordering of the top 10 of her most preferred sushi varieties from among all the 100 varieties. Each sushi variety is described by a set of features like price, oiliness in taste, frequency with which the variety is sold in the shop, and so forth. Table EC.5 in Online Appendix B.2 describes the subset of $D = 5$ features that we used in our experiments. One of the features, style, is binary-valued, and the rest are continuous-valued.

7.1.1. Setup. We processed the data to obtain aggregate market-share information as follows. We assume that customers can choose from any of the 100 varieties of sushi, and they choose their most preferred variety. Therefore, the market share y_j of sushi variety j is equal to the fraction of customers who ranked sushi variety j at the top. Only 93 sushi varieties had nonzero market shares, so we restrict our analysis to these varieties; therefore, $n = 93$. We represent the data as the *empirical market-shares* vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$. We then fit a mixture of logit models to this market-share data using our CG estimator and the EM benchmark. For the EM method, we vary the number K of latent classes over the set $\{2, 3, 4, 5, 10, 15, 20, 25, 30\}$ to estimate K-class LC-MNL models. We initialized the CG estimator with the output of a two-class LC-MNL model fit using the EM algorithm. To solve the optimization problem in the support-finding step, we use a heuristic algorithm (see discussion in Section 5.4 and Online Appendix B.2) that is based on our theoretical development and obtains an approximate solution by exploiting the fact that the optimal solution to the support-finding step must be a boundary type because one of the features is binary-valued (see Theorem 5). We run the CG algorithm for $K_{\max} = 30$ iterations so that the maximum number of types found is at most 30.

7.1.2. In-Sample Fit and Structure of Recovered Mixing Distribution

We first discuss the in-sample performance achieved by both methods. For the NLL loss, we measure the performance in terms of the KL-divergence loss, defined as $D_{KL}^{\text{algo}} \overset{\text{def}}{=} \mathsf{NLL}^{\text{algo}} - H(\boldsymbol{y})$, where $H(\boldsymbol{y}) = -\sum_{j=1}^{n} y_j \cdot \log y_j$ is the entropy of the empirical market-shares vector and represents the lowest achievable in-sample NLL loss (by any method), and $\mathsf{NLL}^{\text{algo}}$ denotes the NLL loss achieved by $\mathsf{algo} \in \{\text{EM}, \text{CG}\}$. Figure 2 plots the in-sample KL-divergence loss and squared loss as a function of the number of customer types for the EM benchmark and the number of iterations for our CG estimator. Note that the number of iterations of the CG method is an upper bound on the

Figure 2. In-Sample Performance on the SUSHI Data Set



Notes. The horizontal axis represents both the number of customer types estimated by the EM benchmark, as well as the number of iterations in the CG algorithm; because the number of iterations is an upper bound on the number of types the CG estimator recovers, the plots represent a fair comparison between the methods. Lower values of the loss are preferred.

number of customer types it recovers. Therefore, in the comparison, the CG method is allowed to use the same number of customer types as—or even fewer than—the EM benchmark.

We make the following observations. First, the CG method consistently achieves a better in-sample fit than the EM benchmark, even when using far fewer customer types. In particular, at the end of 30 iterations, CG achieves $D_{KL}^{\text{CG}} = 0.0372$ with $K = 29$ types as opposed to $D_{KL}^{\text{EM}} = 0.049$ with $K = 30$ types—a 24% reduction. For the squared loss, CG found $K = 23$ types with an in-sample loss of 3.49×10^{-4} as opposed to 8.31×10^{-4} achieved by EM with $K = 30$ types—a 58% reduction. The CG algorithm iteratively adds customer types that explain the observed-choice data to the mixing distribution, which results in a much better fit as compared with the EM algorithm that updates all customer types together in each iteration. In particular, the EM algorithm is directly solving a (nonconvex) optimization problem over $K - 1 + K \cdot D$ parameters, for a K-class LC-MNL model, which makes it challenging to locate the optimal solution. Our method, on the other hand, iteratively searches for the next customer type by solving the support-finding step, which, although a nonconvex problem, can be solved optimally in certain scenarios and possesses a lot more structure. Second, the improvement in SQ loss is significantly higher than the improvement in NLL loss. The reason is that the M-step in the EM benchmark is nonconvex when optimizing the SQ loss; consequently, it can only be solved approximately, resulting in slow convergence and worse performance for the EM benchmark. The CG algorithm, on the other hand, required very little customization,[16] showing its plug-and-play nature when dealing with different loss functions.

We next analyze the structures of the customer types recovered by our method. For this analysis, we focus on the NLL loss. At the end of 30 iterations, the CG method recovered 29 customer types. Except for the 2 customer types that were part of the initial solution, each of the remaining 27 types found by the CG method is a boundary type. These boundary types fall into two classes: those that consider only the maki (a.k.a., rolled sushi) variety and those that consider only the nonmaki variety. It follows from Theorem 5 that these are the only two possible boundary types because there is only one binary-valued feature, representing whether the sushi is maki or nonmaki. Of the 93 varieties of sushi, 13 varieties are maki, and the remaining 80 are nonmaki. We find that five customer types—comprising 2.5% of the probability mass—only consider the 13 maki varieties, so if one of the maki varieties is stocked out, they substitute to one of the remaining maki varieties. The remaining 22 customer types, comprising 46.1% of the probability mass, only consider the 80 nonmaki varieties. The types recovered by our method exhibit strong preferences over the sushi varieties. In fact of the 10 customer types with the largest proportions, 6 types consider only a single sushi variety. By contrast, the EM algorithm recovers customer types who consider all the sushi varieties and is therefore unable to fully capture the underlying heterogeneity in the population with the same number of customer types. See Figure EC.1 in Online Appendix B.2 for a visual representation of the distinction. Our theoretical characterization of the choice behavior of boundary types in Section 5.2 further allows managers to determine changes in sushi characteristics (such as the price) to induce maki customer types to consider nonmaki varieties and vice versa. Finally, the presence of customer types that only consider a single sushi variety is consistent with prior work where customers were observed to (consider and) purchase only a single brand of cars (Lapersonne et al. 1995).

7.1.3. Predictive Accuracy on New Assortments

To test the predictive performance of the recovered mixture on previously unseen assortments, we consider the following tasks:

1. Predict market shares when one/two existing sushi varieties are *dropped* from the assortment.
2. Predict market shares when one new sushi variety is *added* to the assortment.
3. Predict market shares when one existing sushi variety is *replaced* by a new variety.

The above prediction tasks are motivated by real-world situations in which products may be discontinued because of low demand and/or be unavailable due to stock-outs, or new products are introduced into the market. Being able to predict how the population reacts to such changes can be very useful for a firm. We measured predictive accuracies in terms of two popular metrics, mean absolute percentage error and root-mean-square error, which are defined as follows: For each algo $\in \{\text{EM}, \text{CG}\}$ and given any test offer-set S_{test}, we compute

$$\text{MAPE}^{\text{algo}} = 100 \times \left(\frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} \frac{\left| \hat{y}_i - \hat{y}_i^{\text{algo}} \right|}{\hat{y}_i} \right) \quad \text{and}$$

$$\text{RMSE}^{\text{algo}} = \sqrt{\frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} (\hat{y}_i - \hat{y}_i^{\text{algo}})^2},$$

where $\hat{y}_i^{\text{algo}}$ is the *predicted* market share for sushi variety $i \in S_{\text{test}}$ under the mixture of logit models[17] estimated using algo and $\hat{y}_i$ is the *true* market share computed from the test data. We report the *average error* across all possible test assortments. For the first scenario, when one sushi variety is dropped, there are 93 test assortments resulting from dropping each sushi variety in turn. Similarly for the case when two sushi varieties are dropped, resulting in $\binom{93}{2}$ test assortments. When one new variety is added, the training data consist of the market shares when 92 sushi varieties are offered to the population—we consider all 93 training assortments—and, in each case, the test data consist of only a single assortment, containing all 93 varieties. We report the average error on this test assortment across each of the training assortments. Similarly, when one existing variety is replaced by a new variety, the training data consist of market shares when 92 sushi varieties are offered to the population, and for each training assortment, there are 92 test assortments—obtained by replacing each existing sushi variety in turn with a new variety. We first compute the average test error for each training assortment and, finally, report the mean of these average test errors across the training assortments.[18]

Table 2 reports the errors for each prediction task. For the EM algorithm, we choose the best-performing model amongst all estimated K-class LC-MNL models. It is evident that our mixture-estimation method significantly outperforms the EM benchmark across both metrics and all prediction tasks. In particular, we notice an average of 28% reduction in RMSE and 16% reduction in MAPE. Finally, we also observe from Table 2 that the CG method is almost 16× faster than EM-based estimation, showing that it can scale better to data sets containing large number of choice observations.

7.1.4. Decision Accuracy

We now focus on the decision accuracies of the methods. We consider the *assortment-optimization* decision, which involves determining the subset of products to offer to the population to maximize expected revenue.

Setup. In order to compute the optimal assortment and ground-truth revenues, we preprocessed the data as follows: We assume that the 93 sushi varieties with nonzero market shares in the data set comprise the entire sushi market. We focus on maximizing the revenue from the sale of the top 49 sushi varieties by market share. The remaining 44 varieties form the outside option. Treating the outside option as one "product," we obtain a total of $n = 50$ products. Without loss of generality, we suppose that the outside option is indexed by $j = 50$. For each sushi variety $j \in [n]$, we let y_j denote its market share; for the outside option, we obtain its market share by summing the market

Table 2. Mixture-Estimation Times and Error in Market-Share Predictions on Test Assortments

		Drop 1		Drop 2		Add 1		Replace 1	
Estimator	Estimation time (s)	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
EM	914	4.6×10^{-3}	83.67	4.6×10^{-3}	83.17	4.8×10^{-3}	90.23	4.8×10^{-3}	89.71
CG	59	3.3×10^{-3}	69.75	3.4×10^{-3}	69.52	3.4×10^{-3}	75.06	3.5×10^{-3}	75.07
Improvement (%)	93.5	28.3	16.6	26.1	16.4	29.2	16.8	27.1	16.3

Notes. "Drop 1" and "Drop 2" refer, respectively, to the cases when the test assortment is formed by dropping one and two existing sushi varieties from the assortment. "Add 1" and "Replace 1" refer, respectively, to the cases when the test assortment is formed by adding a new sushi variety to the assortment and replacing an existing sushi variety with a new variety. We obtain an average of 28% improvement in the RMSE and 16% in the MAPE metrics. The experiments were conducted on a computer with a 2.1-GHz AMD Opteron(TM) 6272 processor, 32 GB RAM, and Ubuntu 14.04 OS—our approach is almost 16× faster than EM.

shares of all the 44 sushi varieties it comprises. We let r_j denote the price (present as the normalized price feature in the data set) of product j. We set the price of the outside option to 0. We suppose that the outside option is always offered. Then, our goal is to find the subset of the remaining products to maximize the expected revenue; that is, our goal is to solve

$$\max_{S\in[n-1]} \sum_{j\in S} r_j \cdot \big(\text{Probability that } j \text{ is chosen from } S \cup \{n\}\big).$$

We fit mixtures of logit models by optimizing the NLL loss using the CG and EM methods and then solve the above optimization problem under both the models. To solve the optimization problem, we used the mixed-integer linear program (MILP) described in Méndez-Díaz et al. (2014). This MILP takes as input the proportions of each mixture component, product utilities under each mixture component, and the product prices and outputs the optimal assortment. We solved the MILPs using Gurobi Optimizer version 6.5.1. The MILPs ran to optimality, so the recovered assortments were optimal for the given models.

Results and Discussion. We fit a K-class LC-MNL model using the EM method and run the CG algorithm for K iterations to estimate a mixture of logit models, where $K \in \{20, 25, 30\}$. Table 3 reports the optimal assortment sizes and the ground-truth revenues extracted from the population. We compute the ground-truth revenue by assuming that each of the 5,000 customers in the data set purchases the most preferred of the offered products, as determined from her top-10 ranking; if none of the offered products appears in the customer's top-10 ranking, then we assume that the customer chose the outside option.

We note that the EM method offers only five sushi varieties as part of its optimal assortment. The reason is that the customer types recovered by the EM method are not sufficiently diverse (refer to the discussion in Section 7.1.1), because of which the MILP concludes that a small offering suffices to extract the most revenue from the population. In fact, the MILP ends up offering the five sushi varieties with the highest prices. By contrast, our method finds customer types with strong preferences who have sufficiently different tastes, so that the MILP concludes that a larger variety (around 20), consisting of both high-priced and low-priced sushi varieties, is needed in the optimal offering. The consequence is that we are able to extract up to 23% more revenues from the population.

7.2. Case Study 2: IRI Academic Data Set

We now illustrate how our method applies to a typical operations setting in which both the offer-sets and product prices vary over time. Offer-sets vary because of stock-out events (in retail settings) and deliberate scarcity (in revenue-management settings). Prices vary because of promotion activity or dynamic pricing policies. We use real-world sales-transaction data from the IRI Academic Data Set (Bronnenberg et al. 2008), which contains purchase transactions of consumer packaged goods for chains of grocery and drug stores. The data set consists of weekly sales transactions aggregated over all customers. Each transaction contains information such as the week and store of purchase, the universal product code (UPC) of the purchased item, the price of the item, and so forth. For our analysis, we consider transactions for five product categories in the first 2 weeks of the year 2011: shampoo, yogurt, toothbrush, household cleaner, and coffee. Table EC.6 in Online Appendix B.3 describes the summary statistics of the data set.

7.2.1. Setup.

We consider a setup similar to that of Jagabathula and Rusmevichientong (2016), who used the IRI data set to test the predictive power of their pricing method. We preprocess the raw transactions (separately for each product category) as follows. We aggregate the purchased items by vendors[19] to deal with the sparsity of the data. Then, we further aggregate the vendors into $n = 10$ "products"—one product each for the top nine vendors with the largest market shares and a single product for all remaining vendors. This aggregation ensures that there is sufficient coverage of products in the training and test offer-sets.

Table 3. Optimal Assortment Sizes and Ground-Truth Revenue Generated

Estimator	Number of customer types recovered	Optimal assortment size	Revenue
EM	20	5	9.9×10^3
	25	5	9.9×10^3
	30	5	9.9×10^3
CG	20	17	11.9×10^3
	24	20	12.1×10^3
	28	22	12.2×10^3

Notes. Our estimation method is able to extract around 23% more revenue than that generated by the EM benchmark. Here, revenue is measured in units of the normalized price feature (see Table EC.5 in the online appendix) of each sushi variety.

Next, each combination of store and week corresponds to a discrete time period t. The offer-set S_t is chosen as the union of all products purchased during the particular store–week combination. Then, for each product and offer-set pair (j, S_t), the number of sales N_{jt} is computed by using the observed sales for product j in the store–week combination, corresponding to S_t. The price p_{jt} of product j in offer-set S_t is set as the sales-weighted average of the prices of the different UPCs that comprise the product. The number of offer-sets obtained for each product category after this preprocessing step are also listed in Table EC.6 in the online appendix.

We assume that when offered subset S_t and prices $(p_{jt} : j \in S_t)$, each arriving customer samples the MNL parameter vector $(\boldsymbol{\mu}, \beta)$ according to some mixing distribution Q and chooses product $j \in S_t$ with probability:

$$f_{jt}(\boldsymbol{\mu}, \beta) = \frac{\exp\left(\mu_j - \beta \cdot p_{jt}\right)}{\sum_{\ell \in S_t} \exp\left(\mu_\ell - \beta \cdot p_{\ell t}\right)}.$$

Here, $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n) \in \mathbb{R}^n$ are the alternative specific coefficients, and $\beta \in \mathbb{R}$ is the price coefficient. The taste vector $\boldsymbol{\omega} = (\boldsymbol{\mu}, \beta) \in \mathbb{R}^D$ with $D = n + 1 = 11$ in this context.

We fit a mixture of logit models to the processed transaction data using both the CG and EM methods. As for the sushi case study, we initialize the CG algorithm with the output of a two-class LC-MNL model fit using the EM algorithm and solve the optimization problem in the support-finding step using the heuristic method described in Online Appendix B.3 (also refer to the discussion in Section 5.4). We run the CG method for $K_{\max} = 10$ iterations, which results in a mixture with at most 10 customer types. We also fit a 10-class LC-MNL model using the EM algorithm.

7.2.2. Results. Similar to Jagabathula and Rusmevichientong (2016), we conduct a twofold cross-validation. We randomly partition the offer-sets into two parts of roughly equal sizes, fit a mixture of logit model to one part (the training set), and then evaluate its predictions on the other part (the test set). We repeat this process with the train and test sets interchanged. We report performance on both the train and test data sets—all quantities referred to below are computed by taking an average across the two folds. For the NLL loss, we measure the performance using the metric $\Delta^{\mathsf{dataset}}_{\mathsf{algo}} = \mathsf{NLL}^{\mathsf{dataset}}_{\mathsf{algo}} - H^{\mathsf{dataset}}$, where H^{dataset} is the sales-weighted entropy of the observed sales, defined as $H^{\mathsf{dataset}} = -\frac{1}{N}\sum_{t=1}^{T}\sum_{j \in S_t} N_{jt} \log y_{jt}$, for $\mathsf{dataset} \in \{\text{train}, \text{test}\}$, $\mathsf{algo} \in \{\text{EM}, \text{CG}\}$, and $\mathsf{NLL}^{\mathsf{dataset}}_{\text{EM}}$, $\mathsf{NLL}^{\mathsf{dataset}}_{\text{CG}}$ denote the NLL loss achieved by the EM and CG methods, respectively. In Table 4, we report the percentage improvement $100 \times (\Delta^{\mathsf{dataset}}_{\text{EM}} - \Delta^{\mathsf{dataset}}_{\text{CG}})/\Delta^{\mathsf{dataset}}_{\text{EM}}$. Similarly, for the SQ loss, we report $100 \times (\mathsf{SQ}^{\mathsf{dataset}}_{\text{EM}} - \mathsf{SQ}^{\mathsf{dataset}}_{\text{CG}})/\mathsf{SQ}^{\mathsf{dataset}}_{\text{EM}}$, where $\mathsf{SQ}^{\mathsf{dataset}}_{\mathsf{algo}}$ denotes the SQ loss achieved by $\mathsf{algo} \in \{\text{EM}, \text{CG}\}$ on $\mathsf{dataset} \in \{\text{train}, \text{test}\}$.

Table 4. Percentage Improvements in Average Train/Test Loss over EM Benchmark

	SQ loss		NLL loss	
Product category	Train	Test	Train	Test
Shampoo	6.4	5.1	3.4	2.3
Toothbrush	5.3	4.3	2.4	1.3
Household cleaner	5.3	4.1	1.9	1.2
Yogurt	7.0	5.1	8.3	7.1
Coffee	4.3	2.6	3.7	2.4
Average	5.7	4.2	3.9	2.9

Our estimator achieves better in-sample loss across all product categories for both loss functions—an average of 5.7% reduction for SQ loss and 3.9% for the NLL loss. The in-sample improvement is largest for the yogurt category—we obtain 7.0% reduction for SQ loss and 8.3% for the NLL loss. The superior in-sample fit translates to better test performance as well, with 5.1% reduction in SQ loss for the yogurt and shampoo categories and 7.1% reduction in NLL loss for the yogurt category.

We also analyze the structure of the recovered mixing distribution, including the presence of boundary types—see Online Appendix B.3 for a detailed discussion.

8. Extension: Accounting for Endogeneity in Product Features

In many applications of discrete choice modeling, a product feature may be correlated with features not included in the model. The omitted features tend to be those that are unobserved. If such correlations are ignored during estimation, then the coefficient estimated for the included feature could be biased. This phenomenon is referred to broadly as *endogeneity*. The classical example is that product prices are often correlated with unobservables, such as product quality, and ignoring such unobservables may lead one to conclude that higher prices lead to higher demands, when, in fact, the higher demand was caused by higher quality. Petrin and Train (2010) offer other examples of endogeneity.

Several techniques have been proposed in existing literature to deal with the issue of endogeneity in discrete choice models. In this section, we show how one such technique can be incorporated into our method. We use the *control function* method proposed by Petrin and Train (2010), which generalizes the demand shocks approach proposed in Berry et al. (1995). We illustrate its use in our method using the following modification of the simulation setup from Section 6.

8.1. Utility Model

We follow the setup of Section 6. We fix a choice of the ground-truth mixing distribution Q and number of time periods T. We then generate the choice data as follows. In each period $t \in [T]$, a customer arrives and is offered all the $n = 11$ products, including the no-purchase option. Instead of sampling a two-dimensional parameter vector as before, the customer now samples a three-dimensional parameter vector $(\omega_1^{(t)}, \omega_2^{(t)}, \omega_3^{(t)})$ according to Q and assigns the following utility to product j: $U_{jt} = \omega_1^{(t)} \cdot x_{jt} + \omega_2^{(t)} \cdot z_{jt} + \omega_3^{(t)} \cdot \mu_{jt} + \varepsilon_{jt}$, where $(x_{jt}, z_{jt}, \mu_{jt})$ is the feature vector of product j in period t and $(\varepsilon_{jt} : j \in [n])$ are independent and identically distributed standard Gumbel random variables. The feature vector of the no-purchase option is set to (0,0,0). Customers choose the product with the highest utility, resulting in the standard MNL choice probability. The key difference in the utility model from the setup above is that while x_{jt} and z_{jt} are observed, μ_{jt} is unobserved and is correlated with x_{jt}. As is standard in the literature, we assume that the endogenous feature is impacted by a set of instruments and the exogenous feature: $x_{jt} = \gamma_1 \cdot \mathrm{w}_{jt,1} + \gamma_2 \cdot \mathrm{w}_{jt,2} + \gamma_3 \cdot z_{jt} + \mu_{jt}$.

8.2. Control-Function Correction

To deal with endogeneity, the control function (CF) approach obtains a proxy for the term μ_{jt} by regressing the endogenous feature x_{jt} on the instruments ($\mathrm{w}_{jt,1}$, $\mathrm{w}_{jt,2}$) and the exogenous feature z_{jt} and then plugs in the residual $\hat{\mu}_{jt} = x_{jt} - \hat{\boldsymbol{\gamma}}^\top(\mathrm{w}_{jt,1}, \mathrm{w}_{jt,2}, z_{jt})$, where $\hat{\boldsymbol{\gamma}}$ represents the estimated regression parameters. In other words, the method estimates the coefficients using the following utility model: $\hat{U}_{jt} = \omega_1^{(t)} \cdot x_{jt} + \omega_2^{(t)} \cdot z_{jt} + \omega_3^{(t)} \cdot \hat{\mu}_{jt} + \varepsilon_{jt}$.

Once we plug in the residual, the estimators are run as before. They now estimate a mixing distribution over $D = 3$ parameters, where the additional random parameter is for the unobservable μ_{jt}.

8.3. Setup

For our experiments, we sample $(\omega_1^{(t)}, \omega_2^{(t)})$ according to the distribution $Q^{(2)}$, which is a mixture of two bivariate Gaussians, as defined in Section 6. We sample $\omega_3^{(t)}$ according to $\mathcal{N}(-1, 0.3^2)$, independently of $\omega_1^{(t)}$ and $\omega_2^{(t)}$. For each time period t and product j (except the no-purchase option), we sample the exogenous feature z_{jt} according to $\mathcal{N}(0, 1.5^2)$, the instruments $\mathrm{w}_{jt1}, \mathrm{w}_{jt2}$ according to $\mathcal{N}(0,1)$, and the unobservable μ_{jt} according to $\mathcal{N}(0,1)$, all independently of each other. We choose $\boldsymbol{\gamma} = (0.54, 0.54, 0.54)$ to ensure that the marginal distribution of x_{jt} matches the marginal distribution of the features for the case without endogeneity in Section 6. Then, we generate choices for $T = 15{,}000$ periods.

8.4. Results

Table 5 compares our CG method to the standard RPL model with a diagonal variance-covariance matrix on the same RMISE and MIAE metrics, both when endogeneity is ignored and when endogeneity is corrected by using the CF approach. We compute the error metrics only for the distribution of (ω_1, ω_2), and not ω_3. We make the following observations:

1. Ignoring endogeneity can worsen the recovery of the underlying mixing distribution, as is evident in the noticeably larger RMISE value for the benchmark RPL model.
2. Misspecification in the mixing distribution can impact recovery more adversely than ignoring endogeneity. Our method without the CF correction has lower error metrics than the benchmark *with* the CF correction. This shows that having the freedom of choosing the mixing distribution can help mitigate the effects of endogeneity bias.
3. Our estimator is compatible with the CF approach, allowing one to correct for endogeneity and obtain a better approximation to the underlying mixing distribution.

9. Conclusions

This paper proposes a novel nonparametric method for estimating the mixing distribution of a mixture of logit models, given sales transactions and product availability data. Unlike traditional methods that impose a parametric assumption on the mixing distribution, our approach finds the best-fitting distribution to the data, where the fit to the data is measured through a loss function, such as the standard log-likelihood loss, from the class of all possible mixing distributions. We formulate the estimation problem as a constrained convex program by using the insight that, instead of optimizing over the mixing distribution, the estimation problem can be solved by directly optimizing over the predicted choice probabilities for the observed choices in the data—subject to the constraint that they

Table 5. Recovery Metrics with Endogenous Product Features

	RMISE		MIAE	
Estimator	Without CF	With CF	Without CF	With CF
Normal	0.121	0.095	0.057	0.046
NP-CG	0.074	0.059	0.039	0.038

Notes. All differences are statistically significant at the 1% level according to a paired-samples *t*-test. "Without CF" refers to the case when endogeneity is ignored, and "With CF" refers to the case when control function (CF) correction is applied.

are consistent with some underlying mixing distribution. We then apply the conditional gradient algorithm to solve this convex program, which simultaneously performs both tasks of optimizing over the predicted choice probabilities and recovering the underlying mixing distribution consistent with those probabilities. Our theoretical results establish the sublinear convergence rate of our estimator and characterize the structure of the mixing distribution recovered by our method. Specifically, in addition to standard logit types, we show that our method naturally recovers customer types with consideration sets, and our theoretical analysis studies the consideration set structure of such types. Through a numerical study on synthetic data, we show that our estimator can obtain good approximations to various complex ground-truth mixing distributions, despite having no knowledge of their underlying structure. We also show that our approach outperforms the standard EM benchmark in terms of in-sample fit, predictive, and decision accuracies, while being an order of magnitude faster, in two case studies on real data.

There are numerous avenues for future work. We specifically focused on the MNL model in this paper because of its widespread use, but our approach is general and directly applicable for other choice model families (with the caveat that the subproblems can be solved reasonably efficiently). Applying our mixture-estimation technique in the context of choice models like the nested logit or Mallows model is an interesting direction for future work. Moreover, it can be shown that our framework can be used to learn distributions over preference orderings, resulting in a fully nonparametric approach. The subproblem in each iteration in that context corresponds to finding a single preference ordering (or ranking) that has connections with the learning to rank (Liu 2009) and rank-aggregation (Dwork et al. 2001) literatures. In fact, Jagabathula and Rusmevichientong (2019) recently applied some of these ideas to estimate the best-fitting distribution over preference orderings, but they did not take into account any product features. Extending their approach to also account for product features is a promising future direction. Finally, our current estimation method cannot account for fixed parameters (across customer types) in the utility specification. Incorporating fixed effects into the estimation framework will also be an important next step.

Acknowledgments
The authors thank the department editor, the associate editor, and the anonymous referees whose comments and feedback helped improve the manuscript greatly; and the conference participants at Manufacturing & Service Operations Management and INFORMS, the participants at the University of Minnesota Institute for Mathematics and Its Applications Data-Driven Supply Chain Management workshop; and the seminar participants at New York University Stern School of Business, University of Texas Dallas Naveen Jindal School of Management, and Northwestern Kellogg School of Management for their constructive comments that have helped improve the paper.

Endnotes
[1] Indeed, their numerical experiments consider only bivariate mixing distributions.

[2] We use the notation $[m] \overset{\text{def}}{=} \{1, 2, \ldots, m\}$ for any positive integer m in the rest of the paper.

[3] Our method requires some modifications in order to be applied to individual-level panel data. We discuss these modifications in Online Appendix E.

[4] The negative log-likelihood loss is the typical choice in existing techniques.

[5] We note that searching over the space of all possible mixing distributions can lead to potential overfit issues, which we discuss in Section 4.1.

[6] For technical reasons, we need to consider the closure of the set $\mathcal{P}$, which also contains all limit points of convergent sequences in the set $\mathcal{P}$.

[7] Following the *early stopping* rule in machine-learning literature; see, for instance, Yao et al. (2007) and Prechelt (2012).

[8] We use the standard convention that $0 \cdot \log 0 = 0$ when computing the entropy.

[9] It can be verified that the constraint set $\tilde{\mathcal{D}}$ is still compact and convex.

[10] Provided the product features satisfy certain structural conditions; see Theorem 4.

[11] Actually, Jaggi (2013) showed that solving it approximately with some fixed additive error is also sufficient to ensure the $O(1/k)$ convergence rate.

[12] This subsumes the setting of categorical features because a categorical feature is usually transformed into a set of binary features using an encoding scheme like dummy coding or one-hot coding.

[13] We solved problem (3) using Python SciPy library's minimize interface with the "L-BFGS-B" method—https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html (accessed May 28, 2019).

[14] The true mixing distribution's support lies in this region with probability close to 1.

[15] Fox et al. (2011) consider support sizes of k^2 with $k = 1, 2, \ldots, 9$ in their experiments. We refer the reader to Online Appendix B.1 for the complete table of results for our estimator.

[16] In fact, we only had to modify the objective and gradient computations.

[17] We use the mixing distribution estimated by optimizing the NLL loss.

[18] The improvements were similar when considering the minimum and maximum of the average test errors.

[19] Each purchased item in the data set is identified by its collapsed universal product code—a 13-digit-long code with digits 4–8 denoting the vendor.

References
Bach F (2013) Learning with submodular functions: A convex optimization perspective. *Foundations Trends Machine Learn.* 6(2–3):145–373.

Berry S, Levinsohn J, Pakes A (1995) Automobile prices in market equilibrium. *Econometrica* 63(4):841–890.

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

Bohning D, Schlattmann P, Lindsay B (1992) Computer-assisted analysis of mixtures (C.A.MAN): Statistical algorithms. *Biometrics* 48(1): 283–303.

Bronnenberg BJ, Kruger MW, Mela CF (2008) Database paper—the IRI marketing data set. *Marketing Sci.* 27(4):745–748.

Clarkson KL (2010) Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algorithms* 6(4):63.

Dwork C, Kumar R, Naor M, Sivakumar D (2001) Rank aggregation methods for the web. *Proc. 10th Internat. Conf. World Wide Web* (ACM, New York), 613–622.

Feng L, Dicker LH (2018) Approximate nonparametric maximum likelihood for mixture models: A convex optimization approach to fitting arbitrary multivariate mixing distributions. *Comput. Statist. Data Anal.* 122:80–91.

Fox JT, il Kim K, Ryan SP, Bajari P (2011) A simple estimator for the distribution of random coefficients. *Quant. Econom.* 2(3):381–418.

Frank M, Wolfe P (1956) An algorithm for quadratic programming. *Naval Res. Logist. Quart.* 3(1–2):95–110.

Garber D, Hazan E (2015) Faster rates for the Frank-Wolfe method over strongly-convex sets. *Proc. 32nd Internat. Conf. Machine Learn. (ICML-15)* (ACM, New York), 541–549.

Guélat J, Marcotte P (1986) Some comments on Wolfe's 'away step'. *Math. Programming* 35(1):110–119.

Harchaoui Z, Juditsky A, Nemirovski A (2015) Conditional gradient algorithms for norm-regularized smooth convex optimization. *Math. Programming* 152(1–2):75–112.

Hauser JR (2014) Consideration-set heuristics. *J. Bus. Res.* 67(8): 1688–1699.

Hunter DR (2004) MM algorithms for generalized Bradley-Terry models. *Ann. Statist.* 32(1):384–406.

Jagabathula S, Rusmevichientong P (2016) A nonparametric joint assortment and price choice model. *Management Sci.* 63(9):3128–3145.

Jagabathula S, Rusmevichientong P (2019) The limit of rationality in choice modeling: Formulation, computation, and implications. *Management Sci.* 65(5):2196–2215.

Jaggi M (2011) Sparse convex optimization methods for machine learning. Unpublished PhD thesis, ETH Zürich, Zurich.

Jaggi M (2013) Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *Proc. 30th Internat. Conf. Machine Learn. (ICML-13)* (ACM, New York), 427–435.

Jaggi M, Sulovsk M (2010) A simple algorithm for nuclear norm regularized problems. *Proc. 27th Internat. Conf. Machine Learn. (ICML-10)* (ACM, New York), 471–478.

James J (2017) MM algorithm for general mixed multinomial logit models. *J. Appl. Econometrics* 32(4):841–857.

Jiang W, Zhang CH (2009) General maximum likelihood empirical Bayes estimation of normal means. *Ann. Statist.* 37(4):1647–1684.

Joulin A, Tang K, Fei-Fei L (2014) Efficient image and video co-localization with Frank-Wolfe algorithm. Fleet D, Pajdla T, Schiele B, Tuytelaars T, eds. *Computer Vision–ECCV 2014*, Lecture Notes in Computer Science, vol. 8694 (Springer, Cham, Switzerland), 253–268.

Kamishima T, Kazawa H, Akaho S (2005) Supervised ordering—an empirical survey. *5th IEEE Internat. Conf. Data Mining* (IEEE, Piscataway, NJ), 673–676.

Kiefer J, Wolfowitz J (1956) Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters. *Ann. Math. Statist.* 27(4):887–906.

Krishnan RG, Lacoste-Julien S, Sontag D (2015) Barrier Frank-Wolfe for marginal inference. *Adv. Neural Inform. Processing Systems* 28: 532–540.

Lacoste-Julien S, Jaggi M (2015) On the global linear convergence of Frank-Wolfe optimization variants. *Adv. Neural Inform. Processing Systems* 28:496–504.

Laird N (1978) Nonparametric maximum likelihood estimation of a mixing distribution. *J. Amer. Statist. Assoc.* 73(364):805–811.

Lapersonne E, Laurent G, Le Goff JJ (1995) Consideration sets of size one: An empirical investigation of automobile purchases. *Internat. J. Res. Marketing* 12(1):55–66.

Lindsay BG (1983) The geometry of mixture likelihoods: A general theory. *Ann. Statist.* 11(1):86–94.

Lindsay BG (1995) *Mixture Models: Theory, Geometry and Applications*, NSF-CBMS Regional Conference Series in Probability and Statistics, vol. 5 (Institute of Mathematical Statistics, Hayward, CA).

Liu TY (2009) Learning to rank for information retrieval. *Foundations Trends Inform. Retrieval* 3(3):225–331.

McFadden D, Train K (2000) Mixed MNL models for discrete response. *J. Appl. Econometrics* 15(5):447–470.

McLachlan G, Peel D (2000) *Finite Mixture Models* (John Wiley & Sons, New York).

Méndez-Díaz I, Miranda-Bront JJ, Vulcano G, Zabala P (2014) A branch-and-cut algorithm for the latent-class logit assortment problem. *Discrete Appl. Math.* 164(1):246–263.

Nocedal J, Wright SJ (2006) *Numerical Optimization*, 2nd ed. (Springer, New York).

Petrin A, Train K (2010) A control function approach to endogeneity in consumer choice models. *J. Marketing Res.* 47(1):3–13.

Prechelt L (2012) Early stopping—but when? Montavon G, Orr GB, Müller KR, eds. *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science, vol. 7700 (Springer, Berlin), 53–67.

Robbins H (1950) A generalization of the method of maximum likelihood-estimating a mixing distribution. *Ann. Math. Statist.* 21(2):314–315.

Shalev-Shwartz S, Srebro N, Zhang T (2010) Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM J. Optim.* 20(6):2807–2832.

Train KE (2008) EM algorithms for nonparametric estimation of mixing distributions. *J. Choice Model.* 1(1):40–69.

Train KE (2009) *Discrete Choice Methods with Simulation*, 2nd ed. (Cambridge University Press, Cambridge, UK).

Wang YX, Sadhanala V, Dai W, Neiswanger W, Sra S, Xing E (2016) Parallel and distributed block-coordinate Frank-Wolfe algorithms. *Proc. 33rd Internat. Conf. Machine Learn. (ICML-16)* (ACM, New York), 1548–1557.

Yao Y, Rosasco L, Caponnetto A (2007) On early stopping in gradient descent learning. *Constructive Approximation* 26(2):289–315.

Zangwill WI (1969) *Nonlinear Programming: A Unified Approach* (Prentice-Hall, Englewood Cliffs, NJ).

Zhang T (2003) Sequential greedy approximation for certain convex optimization problems. *IEEE Trans. Inform. Theory* 49(3): 682–691.