# Estimating Parameters of Structural Models Using Neural Networks

Max Wei and Zhenling Jiang*

This draft: June 3, 2021

**Abstract**

Machine learning tools such as neural networks are increasingly applied in marketing and economics to learn complex relations in data. The learned relations allow machines to perform various tasks, such as recognizing objects from images or recognizing emotions from speech. This paper explores using a neural net to learn the relation between data (moments) and the parameter values of a structural economic model, so that it can "recognize," or estimate, these parameter values from the data (moments). We train the neural net with the datasets generated by the structural model under different parameter values. The neural net can be trained to give not only the point estimates of parameters but also their statistical accuracy. We show this Neural Net Estimator (NNE) converges to meaningful and well-known limits when the number of training datasets is sufficiently large. NNE does not require computing integrals over the unobservables in the structural model. Thus, it is suitable for models where such integrals are costly in MLE/GMM. We benchmark NNE in two Monte Carlo studies. NNE is able to achieve high estimation accuracies under very light estimation costs.

**Keywords**: structural estimation, neural networks, machine learning, computational costs, market entry, choice models.

*Marshall School of Business, Unversity of Southern California and Wharton School, University of Pennsylvania. Email: yanhaowe@usc.edu; zhenling@wharton.upenn.edu. We thank Eric Bradlow, Greg Lewis, Laura Liu, Aviv Nevo, Holger Sieg, Shunyuan Zhang, and the seminar participants at the 42nd Marketing Science Conference, QME 2020 Conference, UCLA Anderson, and Northwestern Kellogg for their very constructive comments. Max thanks Sam Boysel for research assistance.

# 1 Introduction

Machine learning tools are increasingly applied in the empirical research of marketing and economics. They have been used to classify traditionally intractable data, such as images and audios, or build flexible models to explain and predict economic outcomes, such as consumer choices.[1] These applications rely on the strong capability of machine learning methods to learn complex relationships in the data. Among them, neural networks (together with the algorithms to train them) have been a powerful method to learn functions. For example, neural networks are frequently used to recognize objects in images, effectively learning a mapping from the high-dimensional pixel data of an image to the label for the object in the image (e.g., a cat, elephant, or giraffe).

This paper explores an alternative application of neural networks. We use neural nets to "recognize," or estimate, the parameter values of a given structural econometric model, by learning a mapping from data to parameter values. To train the neural net, we use the structural model to simulate a number of datasets, each of which generated under a different set of parameters. These datasets, together with the known parameter values underlying each of them, constitute the training set from which the neural net learns the mapping from data to parameters. The neural net, once trained, can be applied to the real data to give the estimates for parameters. Alongside point estimates, the neural net can also be trained to output the statistical accuracy of the point estimates for inferences.

We study both the theoretical and practical properties of this proposed estimation approach, which we refer to as the Neural Net Estimator (hereafter NNE). As theoretical properties, we establish that NNE tends to meaningful and well-known limits when the number of training datasets is sufficiently large. NNE does not require computing integrals over the unobservables in the structural model (it only requires using the model to generate training datasets). Thus, as to applications, the approach should be suitable for models where these integrals impose a major cost in the evaluations of likelihood (for MLE) or moment functions (for GMM). We benchmark NNE in two Monte Carlo studies. It is able to achieve good estimation accuracy with very light estimation costs. In addition, we demonstrate that NNE can be used in a way to help pinpoint the source of identification for a parameter.

In the next section, we provide a general formulation of NNE. We start with a conceptual framework of structural estimation. An economic model specifies some outcome $\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\theta})$

[1] See, e.g., Kleinberg et al. (2017), Zhang et al. (2019), Chiong and Shum (2019), and Liu et al. (2019). Athey (2018) offers a comprehensive review on the impact of machine learning on economics.

that depends on a set of observables $\boldsymbol{x}_i$, unobservables $\boldsymbol{\varepsilon}_i$, and structural parameters $\boldsymbol{\theta}$. Depending on the model, $\boldsymbol{g}(\cdot)$ can represent a utility maximization problem, a game between firms, etc. Let $\boldsymbol{y}_i$ denote the observed outcome. An estimator provides a mapping from datasets to parameter values:

$$\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n \mapsto \boldsymbol{\theta}. \tag{1}$$

Extremum estimators (e.g., MLE and GMM) start with this mapping's right side, searching through parameter guesses to maximize an objective function. For MLE, this objective function is the likelihood. Let $\mathbb{I}(\cdot)$ be the indicator function; the likelihood for observation $i$ equals $P(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \int \mathbb{I}\{\boldsymbol{y}_i = \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\theta})\}\, dP(\boldsymbol{\varepsilon}_i)$. In many models, this integral over the unobservables $\boldsymbol{\varepsilon}_i$ is evaluated via simulations, which entails solving $\boldsymbol{g}(\cdot)$ at many draws of $\boldsymbol{\varepsilon}_i$. On top of this, an optimization routine needs to evaluate the likelihood at many different parameter guesses. A similar computational burden applies to GMM.

We attempt to use neural networks to directly learn the mapping in (1). The goal is to have a neural net that outputs the correct parameters when given a dataset generated by the structural model under those parameters. We train this neural net using a "training set," which is constructed as follows. Let $\ell = 1, ..., L$ index the members of the training set. For each $\ell$, we draw parameter values $\boldsymbol{\theta}^{(\ell)}$ from a parameter space $\Theta$. Then, for each $i \in \{1, ..., n\}$ we draw $\boldsymbol{\varepsilon}_i^{(\ell)}$ and compute $\boldsymbol{y}_i^{(\ell)} = \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i^{(\ell)}; \boldsymbol{\theta}^{(\ell)})$. The collection $\{\boldsymbol{\theta}^{(\ell)}, \{\boldsymbol{y}_i^{(\ell)}, \boldsymbol{x}_i\}_{i=1}^n\}$ constitutes a member of the training set. We train a neural net to predict $\boldsymbol{\theta}^{(\ell)}$ from $\{\boldsymbol{y}_i^{(\ell)}, \boldsymbol{x}_i\}_{i=1}^n$ across all $\ell$. After training, the neural net can be applied to the real dataset $\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n$ to output the parameter estimates for the real data.

In some applications, using an entire dataset as the input for NNE requires a large neural net, which can be costly to train. To this end, we replace each dataset with a set of summary statistics, such as the first, second, and cross moments of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i^{(\ell)}$. The neural net then learns a mapping from data *moments* to parameter values. There are two consequences of using data moments as the input of the neural net. First, the cost to train the neural net (after generating the training datasets) need not increase with data size $n$. Second, the parameters are identified from the data moments (which falls into the same line with the method of moments). In this regard, neural nets offer some practical flexibility in that they self-select relevant inputs via training. Specifically, we may include possibly redundant data moments — if certain moments do not contribute to parameter identification, they will not help explain the variation in parameters across $\ell$ in the training set, and thus the neural net

will learn not to use these moments for estimating parameters.[2]

In addition to point estimates, NNE can also provide the statistical accuracy for these point estimates. We let the neural net output a set of standard deviations (SDs) alongside the point estimates for parameters. Intuitively, these SDs are trained to measure the discrepancies between NNE's point estimates and true parameter values in the training set. The neural net can also be configured to output an entire covariance matrix instead of individual SDs for inferences.

We establish several theoretical results. We show as the training set's size $L \to \infty$, the point estimates by an appropriately trained NNE converge to the mean of the limited-information Bayesian parameter posterior (with a flat prior). Here, "limited-information" indicates that the posterior is conditional on a set of data moments instead of the entire data. The SDs given by NNE converge to the standard deviations of the posterior. The covariance matrix given by NNE converges to the covariance matrix of the posterior. These asymptotics are in $L$ and hold for any dataset size $n$. We derive these results using the econometric literature on the learning properties of neural networks (e.g., White, 1990; Chen, 2007). These results provide us with a theoretical justification for the parameter estimates outputted by NNE in general.

Because NNE attempts a direct mapping from data to parameters, it does not require computing integrals over unobserved variables in the model. This property offers a useful practical benefit, because one major source of burden in structural estimation is the need to integrate over unobservables (usually via simulations) to evaluate the likelihood for MLE or moment conditions for GMM. As a result, NNE is suitable for models where the main estimation burden lies in the heavy model simulations to evaluate these integrals. Notable examples are models with rich unobserved firm or consumer heterogeneity. However, NNE may not be suitable for problems where these integrals have closed forms, or the main burden lies elsewhere. For example, the biggest cost in models of dynamic decisions or games is often from solving the dynamic program rather than simulations (the same policy function can be used across the simulations under a given set of parameter values).

Along the above line, we conduct two Monte Carlo studies: a market entry game with unobserved profit shocks (Section 3) and a continuous choice model with random coefficients (Section 4). NNE is able to achieve smaller estimation errors compared to MLE and GMM

[2]More precisely, NNE learns the expectation of parameters conditional on the included data moments as $L \to \infty$ for any fixed $n$ (see Section 2). In this sense, introducing redundant moments does not add finite-sample bias in NNE.

under similar and light estimation costs. The intuition behind this result lies in the different trade-offs between accuracy and costs that simulations introduce in the different estimators. For MLE or GMM, using simulations to evaluate the likelihood or moment functions introduces a source of inaccuracy to the estimates. Increasing the number of simulations reduces the inaccuracy but adds the cost of estimation. NNE is not affected by this inaccuracy. But it receives the inaccuracy caused by training the neural net based on a finite number of simulated training datasets $L$. Increasing $L$ (accompanied by a more flexible neural net) reduces the inaccuracy but adds the cost. In our studies of the two standard structural models, NNE can achieve good estimation accuracy while incurring a very light cost. This result is consistent with the exceptional learning capability of neural nets in general.

A useful property of NNE is that once trained, it can be applied to a newly simulated dataset with almost no extra cost. This property allows us to use NNE to conveniently explore parameter identification. Specifically, we may apply NNE to many simulated datasets, then compare the datasets for which a parameter can be precisely recovered vs. the datasets for which the recovery is difficult. This comparison often offers clues for the source of identification for that parameter. We illustrate this idea in an exercise that attempts to identify the information structure of an entry game.

This paper adds to a fast-growing literature at the intersection of machine learning and marketing/economics. Machine learning tools, such as random forests, neural networks, and reinforcement learning, have been applied to various marketing questions (e.g., Timoshenko and Hauser, 2019; Liu et al., 2019; Zhu et al., 2020; Yoganarasimhan, 2020). Our paper joins a stream of works that leverage machine learning tools in estimation problems. For example, Kaji et al. (2020) apply adversarial machine learning to estimation, where they train classifier algorithms to measure the distance between real and simulated data. Lewis and Syrgkanis (2018) use the adversarial approach to select moment conditions. Wager and Athey (2018) apply random forests to estimate heterogeneous treatment effects. Chernozhukov et al. (2018) study using machine learning to capture high-dimensional nuisance parameters. Our paper is the first to use neural nets to learn a direct mapping from data to parameter estimates and statistical accuracy of these estimates. Further, we establish theoretical properties for NNE by drawing from the econometric literature on the general statistical properties of neural nets (e.g., Chen, 2007; Farrell et al., 2019).

This paper is broadly related to the literature that develops computationally light estimators (e.g., Bajari et al., 2007; Pakes et al., 2007; Su and Judd, 2012). Each of these estimators

mitigates the burden of solving a certain class of economic models. Our estimator, however, focuses on mitigating the burden of evaluating integrals over unobservables. In this aspect, our estimator shares the same focus with indirect inference (Gourieroux et al., 1993; Collard-Wexler, 2013; Bao and Ni, 2017). Indirect inference uses an auxiliary reduced-form model to summarize data, and chooses the parameters of the structural model to minimizes the distance between the summary statistics of the real and simulated data. However, note that indirect inference still falls under extremum estimators. NNE differs in that it attempts a direct mapping from data to parameters and leverages machine learning methods.

## 2 Learning parameters from data

In this section, we start by describing a conceptual framework of structural estimation. Under this framework, we describe how NNE computes the estimates for structural parameters and the statistical accuracy of these estimates. We also establish asymptotic results for NNE.

### 2.1 Structural model estimation

A parametric model specifies an outcome $\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\theta})$ that depends on: (i) a vector of observed characteristics $\boldsymbol{x}_i$, (ii) a vector of unobservables $\boldsymbol{\varepsilon}_i$, and (iii) a vector of model parameters $\boldsymbol{\theta}$. (Throughout this paper, the word "model" exclusively refers to the economic model to be structurally estimated, in order to avoid confusion with machine learning "models.") For example, in a classic entry game (Berry, 1992), each $i$ indexes a market, the output of $\boldsymbol{g}$ collects the entry decision of each firm in market $i$, variable $\boldsymbol{x}_i$ collects the characteristics of the firms in market $i$, and $\boldsymbol{\varepsilon}_i$ collects the unobserved profit shocks of the firms in market $i$.

Let $\boldsymbol{y}_i$ denote the outcome for observation $i$ in the data. The method of maximum likelihood estimation (MLE) estimates the parameters as follows

$$\widehat{\boldsymbol{\theta}}_{MLE} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}} \frac{1}{n}\sum_{i=1}^{n} \log P(\boldsymbol{y}_i|\boldsymbol{x}_i;\boldsymbol{\theta}),$$

where

$$P(\boldsymbol{y}_i|\boldsymbol{x}_i;\boldsymbol{\theta}) = \int \mathbb{I}\left\{\boldsymbol{y}_i = \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\theta})\right\} dP(\boldsymbol{\varepsilon}_i). \tag{2}$$

In the above, $P(\boldsymbol{\varepsilon}_i)$ is the distribution of $\boldsymbol{\varepsilon}_i$ specified by the structural model. Note that to evaluate the likelihood, the unobserved shocks $\boldsymbol{\varepsilon}_i$ need to be integrated out as in equation

(2). In an entry game, these shocks are the unobserved parts of firms' profits. In a choice model with random coefficients, these shocks are the deviations of individuals' coefficients from their population means.

For many models, there is no explicit expression for the integral in equation (2), and one must evaluate it via Monte Carlo simulations. To do so, one makes $R$ random draws of $\boldsymbol{\varepsilon}_i$ from $P(\boldsymbol{\varepsilon}_i)$ and then solves the structural model for each draw of $\boldsymbol{\varepsilon}_i$ to compute $\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\theta})$.[3] On top of this, an optimization routine needs to evaluate the likelihood at hundreds or thousands of different $\boldsymbol{\theta}$'s. As researchers move to increasingly complex models to better capture data, the large number of simulations required to carry out the estimation becomes increasingly an obstacle in implementing these models.

An alternative to MLE is the generalized method of moments (GMM).[4] To construct the moment conditions for GMM, one often makes use of the model-predicted outcome, $\widetilde{\boldsymbol{g}}_i = \int \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\theta}) dP(\boldsymbol{\varepsilon}_i)$. Common moment conditions include the orthogonality between $\boldsymbol{x}_i$ and the prediction residuals $\boldsymbol{y}_i - \widetilde{\boldsymbol{g}}_i$. For many models, the integral to calculate $\widetilde{\boldsymbol{g}}_i$ needs to be evaluated through simulations, which can create a large estimation cost for the same reason above.

More generally, we can think of any estimator as a mapping from the space of datasets to the space of parameter sets:

$$\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n \mapsto \boldsymbol{\theta}. \tag{3}$$

Extremum estimators, including MLE and GMM, start with the right hand side of this mapping. They make parameter guesses and check whether a particular guess produces good fit with data as measured by the likelihood or moment functions.

In recent years, there has been an upward trend in harnessing machine learning tools for empirical modeling in marketing and economics. The focus has primarily centered around two aspects: (i) adding traditionally intractable features in $\boldsymbol{x}_i$, such as the sentiments expressed in reviews or photos, or (ii) devising more flexible functional forms for $\boldsymbol{g}$, such as via random forests. This paper focuses on a different aspect: estimating $\boldsymbol{\theta}$ using machine learning tools,

[3] In some dynamic decision models, given a set of parameter values, the dynamic problem only needs to be solved once for different draws of $\boldsymbol{\varepsilon}_i$. The main estimation cost is from solving for the policy function of the dynamic problem. In this case, NNE is unlikely to offer large cost advantages.

[4] GMM is often applied when MLE is difficult to implement. One example is when $\boldsymbol{y}_i$ is continuous, so that the observed outcomes will always be evaluated as probability zero events with finite draws of $\boldsymbol{\varepsilon}_i$. See Section 4. However, this difficulty does not apply if the model permits a closed-form likelihood function (e.g., Borkovsky et al., 2017).

which we now describe.

### 2.2 Learning parameter values

Our goal is to recover $\boldsymbol{\theta}$ from a dataset $\mathcal{D} \equiv \{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n$, by using neural nets to directly learn the mapping in (3). To motivate, let us consider a simple model where a closed form for mapping (3) is available. In linear regressions, we know that the OLS projection formula, $(\sum_{i=1}^n \boldsymbol{x}_i \boldsymbol{x}_i')^{-1}(\sum_{i=1}^n \boldsymbol{x}_i y_i)$, provides a mapping from data to the regression coefficients. Pretending for a moment that we do not know this formula, we may nevertheless ask machine learning tools, which are designed to learn complex functions (Goodfellow et al. 2016), to learn and approximate the formula. While this idea is redundant for linear regressions, it becomes useful for more complex models, for which closed-form expressions of mapping (3) are generally not available.

By learning the mapping (3), we essentially try to "predict" the structural parameter values from a given dataset. The literature on machine learning tells us that the key to a good predictor is a good training set. In most machine learning tasks (e.g., image classification), good training sets require laborious human labeling. Fortunately, in the context of NNE, a training set can be created using the structural model without much human effort. Specifically, let $\ell = 1, 2, ..., L$. For each $\ell$, draw $\boldsymbol{\theta}^{(\ell)}$ uniformly from a parameter space $\Theta$.[5] Let $\mathcal{D}^{(\ell)} \equiv \{\boldsymbol{y}_i^{(\ell)}, \boldsymbol{x}_i\}_{i=1}^n$ denote a dataset that is generated using the model under $\boldsymbol{\theta}^{(\ell)}$ and conditional on the observed $\{\boldsymbol{x}_i\}_{i=1}^n$.

Now, we can train a machine learning algorithm, such as neural networks, using the training set:

$$\{\boldsymbol{\theta}^{(\ell)}, \mathcal{D}^{(\ell)}\}_{\ell=1,2...,L} \tag{4}$$

The training adjusts the machine learning algorithm so that it can predict $\boldsymbol{\theta}^{(\ell)}$ from $\mathcal{D}^{(\ell)}$ as accurately as possible. Once trained, the algorithm can be used to estimate the parameter values for the real data.

One practical problem here is that the dimension of $\mathcal{D}$ can be large. For example, if $\boldsymbol{y}_i \in \mathbb{R}^2$, $\boldsymbol{x}_i \in \mathbb{R}^{10}$, and $n = 1000$, then the representation of $\mathcal{D}$ requires a vector of size

[5] In practice, one can choose $\Theta$ to cover all parameter values that are considered economically reasonable for the application at hand. More formally, one can verify whether a choice of $\Theta$ is reasonable by checking whether the NNE estimates are sensitive to $\Theta$. We provide more details in the appendix.

In addition, one can draw $\boldsymbol{\theta}^{(\ell)}$ from a non-uniform distribution, particularly if it is a meaningful prior over $\Theta$. Our results on the convergence to Bayesian posterior will still hold (with the understanding that the posterior is computed under this non-uniform prior instead of the flat prior).

$(2+10)n = 1.2 \times 10^4$. Neural nets accepting this size of inputs are not uncommon nowadays, but take a fair amount of time to train. More importantly, it is not necessary to use the entirety of $\mathcal{D}$: the method of moments has taught us that a set of data *moments* is often sufficient for recovering parameters.

We replace $\mathcal{D}$ with a vector of data moments, denoted as $\boldsymbol{m}$. For example, assuming $\boldsymbol{x}_i$ includes a constant, we may specify $\boldsymbol{m}$ to collect the elements in

$$\frac{\sum_{i=1}^n \boldsymbol{y}_i \boldsymbol{x}_i'}{n}, \quad \frac{\sum_{i=1}^n \boldsymbol{y}_i^2 \boldsymbol{x}_i'}{n}, \quad \text{and } \frac{\sum_{i=1}^n \boldsymbol{y}_i (\boldsymbol{x}_i^2)'}{n}.$$

In the above, $\boldsymbol{y}_i^2$ and $\boldsymbol{x}_i^2$ both denote elementwise squares. For $\boldsymbol{y}_i \in \mathbb{R}^2$, $\boldsymbol{x}_i \in \mathbb{R}^{10}$, the dimension of $\boldsymbol{m}$ in this example is $20 + 20 + 20 = 60$, a much more manageable input size for neural nets. Depending on the application, we may also include moments of *demeaned* $\boldsymbol{y}_i$ and $\boldsymbol{x}_i$ (e.g., the cross-covariance) or moments of higher orders.

Replacing $\mathcal{D}$ with $\boldsymbol{m}$ leads to the same identification requirement as in the method of moments.[6] That is, the parameters are identified from the moments selected into $\boldsymbol{m}$. We may not obtain meaningful estimates if a relevant moment is missing from $\boldsymbol{m}$. On the other hand, NNE offers some flexibility in that we do not have to precisely select the relevant moments — we can include potentially redundant moments. If a moment in $\boldsymbol{m}$ is redundant for identifying parameters, it will not help explain the variation in parameters across the training datasets, which means that the neural net will learn (via training) not to use this moment for estimating parameters. Therefore, with a sufficiently large $L$, introducing redundant moments in NNE should not add finite-sample bias in parameter estimates.[7] This intuition is consistent with our later Proposition 1-3 (which hold for any finite $n$).

Using the data moments $\boldsymbol{m}^{(\ell)}$ to represent the data $\mathcal{D}^{(\ell)}$, our training set becomes:

$$\{\boldsymbol{\theta}^{(\ell)}, \boldsymbol{m}^{(\ell)}\}_{\ell=1,2...,L}. \tag{5}$$

Use $\boldsymbol{f} : \boldsymbol{m} \mapsto \boldsymbol{\theta}$ to generically denote any member of the family of mappings from the space of moments to the space of parameter values. We seek to find a specific mapping $\widehat{\boldsymbol{f}}$ in this family such that $\boldsymbol{\theta}^{(\ell)} \simeq \widehat{\boldsymbol{f}}(\boldsymbol{m}^{(\ell)}), \forall \ell$. If we denote the moments from the real data as $\boldsymbol{m}^{(0)}$,

[6] Lack of identification can arise in structural models for multiple reasons (notably with multiple equilibria; see Tamer, 2010). This paper does not attempt to tackle this issue. In this paper, we focus on the problem of estimation *given* identification.

[7] For finite-sample bias problem with redundant moments in GMM, see Chen and Liao (2015) and the references therein.

then the parameter estimates for the real data are simply the plug-in estimates $\widehat{\boldsymbol{f}}(\boldsymbol{m}^{(0)})$.

We construct $\widehat{\boldsymbol{f}}$ using neural nets. Here, we do not delve into the details of neural nets. For our exposition, it is sufficient to consider neural nets as a flexible functional form for $\boldsymbol{f}$ parameterized by a set of "edge weights" (see Appendix A.1 for more details). As in most machine learning algorithms, an appropriate neural net is trained by minimizing a loss function. Let $k$ index the dimensions of $\boldsymbol{\theta}$. We first consider the loss function that computes the mean squared error (MSE) of a candidate $\boldsymbol{f}$ in the training set:

$$C_1(\boldsymbol{f}) = L^{-1} \sum_{\ell=1}^{L} \sum_{k} \left[ f_k(\boldsymbol{m}^{(\ell)}) - \theta_k^{(\ell)} \right]^2. \tag{6}$$

We choose $\widehat{\boldsymbol{f}}$ as the neural net $\boldsymbol{f}$ that minimizes $C_1$. There are alternative loss functions that we will consider later. In terms of computation, the minimization of any loss function can be quickly performed via a routine known as back-propagation, which is implemented in popular computational softwares such as Matlab and R. Note that we do not have a regularization term in the loss function. This choice puts us in the standard econometric framework for neural networks where the neural net complexity is taken to be small relative to the training set. Adding a regularization term would trade off bias for variance, and may require debiasing.[8]

An important question is whether the trained neural net is able to give meaningful parameter estimates. In practice, we can assess this question by examining the test error of $\widehat{\boldsymbol{f}}$. Specifically, we simulate additional datasets $\ell = L+1, ..., L^*$ to form a test set, and then calculate the test error as the mean squared error of $\widehat{\boldsymbol{f}}$ in the test set. Importantly, note that no parts of the test set have been used in the training of $\widehat{\boldsymbol{f}}$. In other words, the test error reflects how well NNE recovers parameter values from the datasets that it has not seen during training.

There is also theoretical basis for NNE to give meaningful parameter estimates. Particularly, one can show that $\widehat{\boldsymbol{f}}$ converges to something meaningful as $L \to \infty$. We first describe the intuition of this result. Note that conditional on $\{\boldsymbol{x}_i\}_{i=1}^n$, there is a joint distribution $P(\boldsymbol{\theta}, \boldsymbol{m})$ implied by the uniform distribution over $\Theta$ and the structural model (which generates $\boldsymbol{m}$ given $\boldsymbol{\theta}$). The pairs $\{\boldsymbol{\theta}^{(\ell)}, \boldsymbol{m}^{(\ell)}\}_{\ell=1}^L$ are i.i.d. samples from the distribution $P(\boldsymbol{\theta}, \boldsymbol{m})$. As a result, the loss function $C_1$ in (6) is a sample version of the following expected loss,

[8] See Chernozhukov et al. (2018).

where the expectation is taken over $P(\boldsymbol{\theta}, \boldsymbol{m})$.

$$\overline{C}_1(\boldsymbol{f}) = \sum_k \mathbb{E}\left[f_k(\boldsymbol{m}) - \theta_k\right]^2.$$

Intuitively, with $L \to \infty$, minimizing $C_1$ approaches minimizing $\overline{C}_1$. From probability theory, we know that an expected square error such as $\mathbb{E}\left[f_k(\boldsymbol{m}) - \theta_k\right]^2$ is minimized by choosing $f_k(\boldsymbol{m}) = \mathbb{E}(\theta_k|\boldsymbol{m})$ for all $\boldsymbol{m}$. Therefore, we would expect $\widehat{\boldsymbol{f}}$ to approach $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ as a function of $\boldsymbol{m}$. This expectation $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ can be seen as a best estimate for parameters given the data moments. If we use the language of Bayesian statistics, $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ is known as the mean of the limited-information Bayesian posterior (under the flat prior over $\Theta$). Here, "limited-information" indicates that the posterior is conditional on a set of data moments rather than the complete dataset (Kim, 2002). We should obtain the standard Bayesian posterior mean if using the training set (4) instead.

We formalize the above intuition with the help of the nonparametric literature on the general properties of neural nets. To derive asymptotic results, we introduce a sequence of function spaces $\{\mathcal{F}_L\}_{L=1}^{\infty}$. Here, $\mathcal{F}_L$ generically denotes a set of functions representable by a class of neural nets. Because the exact specification of $\mathcal{F}_L$ concerns only technical aspects of deriving asymptotic results (as White 1989 also points out), we describe it in the appendix. For our exposition below, the key property is that the complexity of $\mathcal{F}_L$ grows with $L$ (in the form of more hidden units and larger bounds for edge weights). The intuition follows the general theory of nonparametric fitting — the fitting model needs to adapt to sample size.

The proposition below is shown using Chen (2007) and White (1990). As above, let $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ be the function of $\boldsymbol{m}$ that gives the conditional mean of parameters. Let $\widehat{\boldsymbol{f}}_L$ denote the neural net in $\mathcal{F}_L$ that minimizes the loss function $C_1$. We use the 2-norm $\|\cdot\|$ to measure functions: $\|\boldsymbol{f}\| \equiv \left[\int \sum_k f_k(\boldsymbol{m})^2 dP(\boldsymbol{m})\right]^{1/2}$. A norm of zero means $\boldsymbol{f}$ is zero a.s.

**Proposition 1.** *Suppose: (i) $\Theta$ is compact; (ii) the moments $\boldsymbol{m}$ has a compact support; (iii) $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ is continuous in $\boldsymbol{m}$. Under the loss function (6), we have $\|\widehat{\boldsymbol{f}}_L - \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})\| \to 0$ in probability as $L \to \infty$.* □

All proofs are given in Appendix A.4. In words, the proposition says, by increasing the number of simulated datasets in the training set, we can obtain an approximation to $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ to an arbitrary precision. Note that the result is in $L \to \infty$ and thus holds for any data size $n$. The conditions are relatively mild: the compactness of $\Theta$ is also assumed in

standard treatment of MLE and GMM. The support-compactness of $\boldsymbol{m}$ can be restrictive technically but is satisfied in many applications (see Farrell et al. 2019 for some discussion). The continuity condition for $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ is satisfied in most applications, and in particular does not preclude $\boldsymbol{\theta}$ from taking discrete values.

The main purpose of the theoretical result here is to provide a justification for the parameter estimates given by our proposed approach. Though the justification is provided through a link to Bayesian posteriors, it is important to note that at least in this paper, we do not attempt to establish NNE as a Bayesian method. In marketing, Bayesian methods are typically used in hierarchical forms to capture and estimate the individual parameters of consumers. In this paper, the focus for our applications of NNE is to reduce the costs of estimating the main parameters in some structural models.

### 2.3 Learning accuracy

The development so far has focused on the point estimates of parameters. Next, we focus on computing the statistical accuracy of these point estimates to allow inferences. Continuing with the link to $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ established above, we seek to estimate $\mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})$. In the terminology of Bayesian statistics, we seek to estimate the variance of the limited-information posterior distribution.[9]

First, we configure the neural net so that its output has two parts. Using $\boldsymbol{f}$ to denote a neural net, we write $\boldsymbol{f} = (\boldsymbol{\mu}, \boldsymbol{V})$, where $\boldsymbol{\mu}$ gives a mean vector and $\boldsymbol{V}$ gives a covariance matrix. With this notation, it is understood that the outputs of $\boldsymbol{f}$ are re-arranged and duplicated, if necessary, into a mean vector and a covariance matrix. For example, $\boldsymbol{f}$ needs to give only the lower triangular entries of the covariance matrix, and if we restrict $\boldsymbol{V}$ to be diagonal, $\boldsymbol{f}$ only needs to give the diagonal entries. In applications, one may also want to re-parameterize the covariance matrix using the Cholesky decomposition.[10]

In the training set, let $(\boldsymbol{\mu}^{(\ell)}, \boldsymbol{V}^{(\ell)}) \equiv \boldsymbol{f}(\boldsymbol{m}^{(\ell)})$ for each $\ell$. In other words, $\boldsymbol{\mu}^{(\ell)}$ and $\boldsymbol{V}^{(\ell)}$ are the two parts of the output given by the neural net for the training dataset $\ell$. We use the following loss function to train the neural net. The loss function uses the log density

[9] An alternative approach to assess the accuracy of the estimates by NNE is bootstrapping. In principle, one can apply bootstrapping to NNE in the same way that it applies to MLE or GMM. However, bootstrapping is computationally heavy in general.

[10] An alternative way to compute $\mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})$ is to rely on Proposition 1 to train separate neural nets for the first and second moments of $P(\boldsymbol{\theta}|\boldsymbol{m})$. Then, the variance can be computed as the difference between the second moment and the first moment squared. We give more details, as well as the asymptotic result, of this approach at the end of Appendix A.4.

function of a normal distribution (however, it does not require normality on the underlying distribution of $\boldsymbol{\theta}$ given $\boldsymbol{m}$, as we will show).

$$C_2(\boldsymbol{f}) = L^{-1} \sum_{\ell=1}^{L} \log\left(|\boldsymbol{V}^{(\ell)}|\right) + \left(\boldsymbol{\theta}^{(\ell)} - \boldsymbol{\mu}^{(\ell)}\right)' \left(\boldsymbol{V}^{(\ell)}\right)^{-1} \left(\boldsymbol{\theta}^{(\ell)} - \boldsymbol{\mu}^{(\ell)}\right). \tag{7}$$

We train a neural net $\widehat{\boldsymbol{f}}$ that minimizes this loss function $C_2$. Very importantly, $C_2$ does *not* require $P(\boldsymbol{\theta}|\boldsymbol{m})$ to be normal. Specifically, for any general distribution for $P(\boldsymbol{\theta}|\boldsymbol{m})$, if we restrict $\boldsymbol{V}$ to be diagonal, then $\widehat{\boldsymbol{f}}$ will consistently estimate the mean and variances of this distribution. If we allow a full covariance matrix $\boldsymbol{V}$, then $\widehat{\boldsymbol{f}}$ will additionally consistently estimate the covariance terms of this distribution. We formalize these results below. For this proposition, we use $\widehat{\boldsymbol{f}}_L$ to denote a minimizer of the loss function (7) in $\mathcal{F}_L$.

**Proposition 2.** *Suppose condition (i) - (iii) in Proposition 1 hold, and in addition: (iv)* $\mathbb{V}\mathrm{ar}(\boldsymbol{\theta}|\boldsymbol{m})$ *is continuous in* $\boldsymbol{m}$ *and* $\mathbb{V}\mathrm{ar}(\boldsymbol{\theta}|\boldsymbol{m}) \geq \delta$ *for some* $\delta > 0$*. Under loss (7) with diagonal covariance matrix, we have* $\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| \to 0$ *in probability with* $\boldsymbol{f}^* = [\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m}), \mathbb{V}\mathrm{ar}(\boldsymbol{\theta}|\boldsymbol{m})]$.

*In addition to condition (i)-(iv) above, suppose: (v)* $\mathbb{C}\mathrm{ov}(\boldsymbol{\theta}|\boldsymbol{m})$ *is continuous in* $\boldsymbol{m}$ *and its smallest eigenvalue is bounded below by some positive number. Under loss (7) with full covariance matrix, we have* $\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| \to 0$ *in probability with* $\boldsymbol{f}^* = [\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m}), \mathbb{C}\mathrm{ov}(\boldsymbol{\theta}|\boldsymbol{m})]$. □

Note the proposition does not make any distributional assumption on $P(\boldsymbol{\theta}|\boldsymbol{m})$. Nevertheless, it is useful to point out that $P(\boldsymbol{\theta}|\boldsymbol{m})$ generally tends to a normal distribution as $n \to \infty$. This result comes from Bayesian statistics, where it is known that under regularity conditions, parameter posteriors tend to normality as the data size $n$ becomes large (Gelman et al., 2004). The same result also holds for limited-information Bayesian posterior (Kim, 2002). In applications, if one is willing to assume an (approximately) normal $P(\boldsymbol{\theta}|\boldsymbol{m})$, then the mean and covariance matrix given by $\widehat{\boldsymbol{f}}(\boldsymbol{m}^{(0)})$ are adequate to characterize the full distribution of $\boldsymbol{\theta}$ given the data moments $\boldsymbol{m}^{(0)}$.

The intuition behind Proposition 2 can be obtained by, again, looking for the population version of the loss function. In fact, we can gain more intuition from a more general version of the loss function as follows. Let $\phi(\cdot;\boldsymbol{\gamma})$ be a family of positive density functions for $\boldsymbol{\theta}$ parameterized by vector $\boldsymbol{\gamma} \in \Gamma$ for some space $\Gamma$. Let the neural net $\boldsymbol{f}$ output the components of $\boldsymbol{\gamma}$. Let

$$C_3(\boldsymbol{f}) = L^{-1} \sum_{\ell=1}^{L} - \log \phi \left[ \boldsymbol{\theta}^{(\ell)};\ \boldsymbol{f}(\boldsymbol{m}^{(\ell)}) \right]. \tag{8}$$

Note that $C_3$ reduces to $C_2$ if $\phi$ takes the probability density of normal distributions; it further reduces to $C_1$ if we assume an identity covariance matrix for the normal density. More generally, loss functions based on a log probability density are known as the cross-entropy loss functions in the machine learning literature.

Similar to how we have seen $C_1$ as a sample version of $\overline{C}_1$, the loss function in (8) can be seen as a sample version of

$$\overline{C}_3(\boldsymbol{f}) = -\mathbb{E} \left[ \log \phi \left( \boldsymbol{\theta};\ \boldsymbol{f}(\boldsymbol{m}) \right) \right].$$

As we know from statistical theory, minimizing the above expectation for any $\boldsymbol{m}$ amounts to a minimization of the Kullback–Leibler divergence from the density $\phi$ to the distribution $P(\boldsymbol{\theta}|\boldsymbol{m})$. As a result, it is reasonable to expect the trained neural net to parameterize $\phi$ such that the Kullback–Leibler divergence from $\phi$ to $P(\boldsymbol{\theta}|\boldsymbol{m})$ is as small as possible, even if $\phi$ is not correctly specified to include the latter's true distribution. As it turns out, the Kullback–Leibler divergence from a normal family $\phi$ to any general distribution is minimized when $\phi$ takes the mean and covariance matrix of that distribution (White, 1982), thus we have Proposition 2.

We formalize the above intuition regarding $C_3$ in the following proposition. For this proposition, let $\widehat{\boldsymbol{f}}_L$ denote the neural net in $\mathcal{F}_L$ that minimizes the loss function $C_3$. We use $\mathcal{KL}(\cdot\|\cdot)$ to denote the Kullback–Leibler divergence. Recall that $\Gamma$ denotes the space for $\boldsymbol{\gamma}$ that parameterizes the density family $\phi(\cdot;\boldsymbol{\gamma})$.

**Proposition 3.** *Let the loss function be (8). Suppose: (i) $\Gamma$ is compact with a non-empty interior, (ii) $\boldsymbol{m}$ has a compact support, (iii) $\phi$ is continuously differentiable on $\Theta \times \Gamma$, (iv) $\mathbb{E}\left[\log \phi(\boldsymbol{\theta};\boldsymbol{\gamma})|\boldsymbol{m}\right]$ is continuous in $\boldsymbol{m}$ and $\boldsymbol{\gamma}$, (v) for each $\boldsymbol{m}$, $\operatorname{argmin}_{\boldsymbol{\gamma}\in\Gamma} \mathcal{KL}\left[P(\boldsymbol{\theta}|\boldsymbol{m}) \parallel \phi(\boldsymbol{\theta};\boldsymbol{\gamma})\right]$ is a single point in the interior of $\Gamma$. Let $\boldsymbol{f}^*(\boldsymbol{m})$ denote the minimizing point in (v). Then $\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| \to 0$ in probability.* □

In words, NNE asymptotically minimizes the distance between $\phi$ and $P(\boldsymbol{\theta}|\boldsymbol{m})$. The key conditions are a sufficient level of continuity and the uniqueness of the distance-minimizing density in the $\phi$ family. Compared to the two propositions above, these conditions in Propo-

sition 3 are more high-level. However, they allow us to show that the link between NNE and Bayesian statistics holds under the much more general loss function.

In this section, we have provided a treatment of NNE as an alternative approach for parameter estimation and inference. By construction, an important property of NNE is that it does not require computing integrals over the unobserved variables in the structural model (All that it requires is the ability to simulate the model to generate the training datasets). These integrals are generally needed to evaluate likelihood or moment functions. Thus, NNE likely holds an advantage in the problems where the computational burden to evaluate these integrals is particularly high. We examine this intuition with Monte Carlo studies in the next sections.

Finally, note that the application of NNE does not preclude one from using the inference theory of extremum estimators (which is admittedly far more fully established in literature). One can use NNE to produce a set of parameter estimates, and then feed them as starting values for extremum estimators. Good starting values significantly reduce the cost of implementing extremum estimators.

## 3 Monte Carlo: Market Entry Game

In the previous section, we establish asymptotic results for NNE when the number of training datasets is sufficiently large. In this and next session, we use Monte Carlo studies to examine NNE's performance in applications. This section focuses on estimating entry games, where multiple firms simultaneously decide whether to enter a market. The strategic interactions between firms introduce a potentially complex mapping from the parameters to the observable outcomes (i.e., the entry decisions), which makes the estimation a non-trivial task (see Ellickson and Misra 2011 for a summary of the challenges in estimating entry games).

Entry games have the feature that makes NNE suitable: the probabilities for firms' entry decisions need to integrate over unobserved firm heterogeneity, and the simulations required to evaluate these integrals constitute a major part of the estimation cost. The simulations also introduce inaccuracy in the likelihood or moment functions, which leads to additional estimation errors. NNE avoids integrating over the unobservables and thus may require a lower cost (for the same estimation errors) or allow smaller estimation errors (at the same cost).

Below, we first briefly describe the setup. Then, we implement NNE and compare it with

extremum estimators. Finally, we explore using NNE to find the source of identification for the information structure of the game.

### 3.1 Model and data setup

We describe the model, the standard approaches to estimate the model, and how we simulate datasets for the Monte Carlo study. We roughly follow Berry (1992). Index markets by $k \in \{1, ..., K\}$. There are $J$ potential firms in each market. Each firm decides whether to enter the market. We use $s_{j,k} \in \{0, 1\}$ to indicate whether firm $j \in \{1, ..., J\}$ in market $k$ enters the market. The profit for an entering firm is specified as

$$\pi_{j,k} = \boldsymbol{x}'_{j,k}\boldsymbol{\beta} - \delta N_k + \varepsilon_{j,k},$$

where $\boldsymbol{x}_{j,k}$ collects some observed characteristics for the firm, $N_k \equiv \sum_{j=1}^{J} s_{j,k}$ is the number of entrants in market $k$, and $\varepsilon_{j,k}$ is a profit shock unobserved to econometrician but known to all the firms. We assume $\varepsilon_{j,k}$ follows i.i.d. $\mathcal{N}(0, 1)$.

A firm enters if and only if it makes positive profit:

$$s_{j,k} = \begin{cases} 1, & \text{if } \pi_{j,k} > 0; \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

There can be multiple Nash equilibria. For example, consider $J = 2$ and a market that can support only one firm to be profitable; either firm entering is an equilibrium. Since our focus here is not the multiplicity issue, we use a selection rule assumed by Berry (1992), which asks firms to enter by the order of profitability. Specifically, rank firms by $\boldsymbol{x}'_{j,k}\boldsymbol{\beta} + \varepsilon_{j,k}$. We ask the highest ranked firm to enter first, then ask the second highest firm to enter, ..., until we reach a firm that finds it not profitable to enter.[11]

Given a selection rule, we can write down the likelihood function in the MLE as follows.

$$\sum_{k=1}^{K} \log \Pr\left(\boldsymbol{s}_k | \boldsymbol{x}_k; \boldsymbol{\beta}, \delta\right),$$

where $\boldsymbol{s}_k$ is the vector collecting $s_{1,k}, ..., s_{J,k}$, and $\boldsymbol{x}_k$ is the matrix collecting $\boldsymbol{x}_{1,k}, ..., \boldsymbol{x}_{J,k}$.

[11] Alternative selection rules (see Ellickson and Misra, 2011) can greatly increase the computational cost to simulate a game. Generally, for models that are costly to simulate, an estimator that requires a smaller number of simulations, such as NNE, provides an even larger saving of computational time.

The likelihood term for market $k$, $\Pr(\boldsymbol{s}_k|\boldsymbol{x}_k;\boldsymbol{\beta},\delta)$, is the probability of observing the entry outcome $\boldsymbol{s}_k$ in the data given the values of $\boldsymbol{x}_k$ and the parameters. Note that it is a *joint* probability over all the firms in market $k$, which accounts for the interdependence among the firms' entry decisions.

The entry probability $\Pr(\boldsymbol{s}_k|\boldsymbol{x}_k;\boldsymbol{\beta},\delta)$ is evaluated by simulations. Specifically, let $r = 1,...,R$ index the simulations. Each simulation $r$ draws a vector $\boldsymbol{\varepsilon}_k^{(r)}$ and then computes the associated Nash equilibrium $\boldsymbol{s}_k^{(r)}$ under $\boldsymbol{x}_k$, $\boldsymbol{\beta}$, and $\delta$. We approximate the entry probability by $R^{-1}\sum_{r=1}^{R}\mathbb{I}\{\boldsymbol{s}_k^{(r)} = \boldsymbol{s}_k\}$, that is, the portion of simulation draws that result in the same entry outcome as observed in the data. However, there are two caveats. First, because entry decisions are discrete, the simulated likelihood is a non-smooth function of parameters. This non-smoothness causes difficulties for optimization routines. The second problem is related but more fundamental. The number of possible entry outcomes for a market is $2^J$, which is large even with a modest $J$ such as $J = 5$ or 10. Thus, unless $R$ is very large, we are likely to run into the case where an observed outcome $\boldsymbol{s}_k$ does not match any of the simulated outcomes. In this case, $\boldsymbol{s}_k$ is evaluated as a probability zero event and the likelihood function is undefined. A common solution to these problems is to smooth the likelihood function. We provide details in Appendix A.2.

Aside from MLE, a common estimation strategy for entry games is GMM. One defines a firm-specific residual $\nu_{j,k}$ as the difference between the observed and predicted entry decisions:

$$\nu_{j,k} = s_{j,k} - \mathbb{E}\left(s_{j,k}|\boldsymbol{x}_k;\boldsymbol{\beta},\delta\right),\ j = 1,...,J.$$

The expectation on the right hand side above is evaluated by simulations as $R^{-1}\sum_{r=1}^{R} s_{j,k}^{(r)}$. To construct moment conditions, we interact $\nu_{j,k}$ with $\boldsymbol{x}_k$ separately for every $j$. GMM does not have the problem of the objective function being undefined as described above for MLE. But the objective function is still non-smooth with respect to the parameters. Thus, one may use a non-gradient-based optimization routine. We use pattern search.

One may think of the Bayesian approach as an alternative estimation strategy (particularly considering our theoretical results that link NNE to Bayesian posteriors). However, the Bayesian approach typically has not been used to estimate entry games, probably because it holds no clear advantages over MLE in this case. In marketing, the Bayesian approach is more often used in hierarchical forms to facilitate the estimation of individual-level parameters featured in some models.

Our Monte Carlo study first uses the entry model to generate datasets, then applies various

estimators (MLE, GMM, and NNE) on these generated datasets to recover parameters. For exposition, we will refer to any such generated dataset as a "real" dataset, to distinguish it from the also-model-generated datasets used to train NNE.

Specifically, we generate a "real" dataset as follows. We set the number of markets $K = 1000$ and the number of firms $J = 5$. For each market and firm, $\boldsymbol{x}_{j,k}$ is drawn as a 11-by-1 vector. The first element $x_{1,j,k}$ is a market-specific characteristic i.i.d. $\mathcal{N}(0,1)$ across markets. So $x_{1,j,k} = x_{1,1,k}$ for $j = 2, ..., J$. (Examples of market-specific characteristics are the consumer population or the median household income in a market.) Elements $x_{2,j,k}$ to $x_{6,j,k}$ are the firm dummies interacted with a second market-specific characteristic that is again i.i.d. $\mathcal{N}(0,1)$ across markets. Elements $x_{7,j,k}$ to $x_{11,j,k}$ are simply the firm dummies. Finally, we draw a vector of $\boldsymbol{\varepsilon}_k$ for each market $k$ and solve for the associated Nash equilibrium to produce $\boldsymbol{s}_k$.

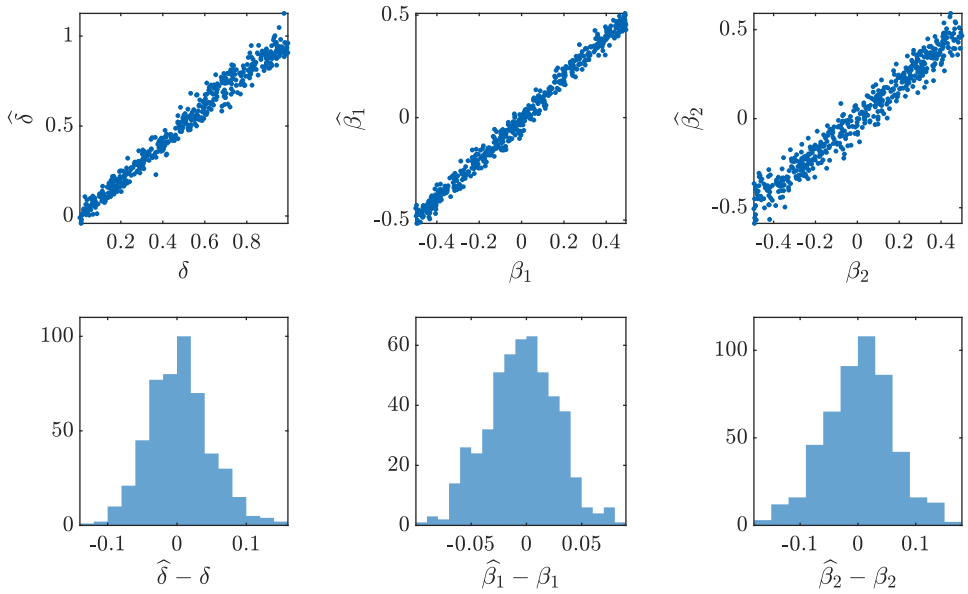
### 3.2 Implementing NNE

Now we describe the application of NNE in the entry game. The implementation of NNE has two steps: generating the training set and applying a neural net to the training set. For the first step, we use the entry model to generate $L^* = 5000$ datasets to construct 5000 parameter-moment pairs, $\{\boldsymbol{\theta}^{(\ell)}, \boldsymbol{m}^{(\ell)}\}_{\ell=1}^{L^*}$. Among these pairs, we use 90% for training and the rest 10% for testing.

Specifically, we generate the parameter-moment pairs as follows. Let $\ell = 1, ..., L^*$. For each $\ell$, we first draw $\delta^{(\ell)}$ uniformly from the unit interval $[0, 1]$ and each element of $\boldsymbol{\beta}^{(\ell)}$ uniformly from $[-\frac{1}{2}, \frac{1}{2}]$. This range for the parameter space $\Theta$ is chosen to allow a variety of distributions for the number of entrants across markets. As discussed in Section 2.2, in a real application, one can choose $\Theta$ based on what parameter values are considered economically reasonable. In addition, one can conduct a sensitivity analysis of the NNE estimates to help decide $\Theta$. We describe this analysis in more detail in Appendix A.3.

Given $\boldsymbol{\theta}^{(\ell)} = (\delta^{(\ell)}, \boldsymbol{\beta}^{(\ell)})'$, we draw $\boldsymbol{\varepsilon}_k^{(\ell)}$ for each market $k$ and solve the associated entry equilibrium to realize the entry decision $s_{j,k}^{(\ell)} \in \{0, 1\}$ for all $j$. The entry decisions imply the number of entrants $N_k^{(\ell)} = \sum_{j=1}^{J} s_{j,k}^{(\ell)}$. Let $\boldsymbol{y}_{j,k}^{(\ell)} = (s_{j,k}^{(\ell)}, N_k^{(\ell)})'$. For the data-summarizing moments $\boldsymbol{m}^{(\ell)}$, we include: (i) the mean and covariance matrix of $\boldsymbol{y}_{j,k}^{(\ell)}$, (ii) the cross-covariance matrix between $\boldsymbol{y}_{j,k}^{(\ell)}$ and $\boldsymbol{x}_{j,k}$, and (iii) the mean and variance of $\boldsymbol{x}_{j,k}$. This configuration amounts to $5+22+22 = 49$ moments to be fed to NNE. In principle, one need not include the moments involving $\boldsymbol{x}_{j,k}$ only, because such moments are constant across $\ell$. However, we find

Figure 1: Estimates by NNE in Test Set for Market Entry




Notes: The top row plots the NNE estimates against true parameter values in the test set (500 datasets). The bottom row plots the distribution of NNE's estimation errors in the test set. The NNE shown here is trained using loss $C_1$.

that depending on the application, including these moments may slightly reduce the burden in training the neural net.

We apply a simple neural network using one hidden layer with the rectifier activation function.[12] For $L^* = 5000$, we use 128 units in the hidden layer (It is a common practice to use the powers of 2 as the number of hidden units). We find that the results in our Monte Carlo studies are, to a large extent, not sensitive to the choice of neural net configuration. The training of this neural net takes about 10 seconds with Matlab's deep learning package on a regular desktop. We give more details on how we configure and train neural nets in Appendix A.1.

To evaluate estimation accuracy, we apply the trained neural net to each dataset in the 10% test set. The top row of Figure 1 plots the NNE estimates against the true parameter values, for $\delta$, $\beta_1$, and $\beta_2$. The clear concentration around the 45 degree lines shows that

[12] All of our results hold using sigmoid activation functions. The rectifier is a more popular choice for the activation function in practice because it tends to reduce the burden of training.

Table 1: NNE's Test-Set Performance for Market Entry

Squared-error loss $C_1$:

| | $\delta$ | $\beta_1$ | $\beta_2$ | $\beta_3$ - $\beta_{11}$ |
|---|---|---|---|---|
| Mean Bias | 0.004 (.002) | -0.001 (.001) | -0.002 (.002) | ... |
| RMSE | 0.045 (.001) | 0.031 (.001) | 0.056 (.002) | ... |
| Mean SD | - | - | - | - |

Entropy loss $C_2$:

| | $\delta$ | $\beta_1$ | $\beta_2$ | $\beta_3$ - $\beta_{11}$ |
|---|---|---|---|---|
| Mean Bias | -0.001 (.002) | -0.002 (.001) | 0.004 (.003) | ... |
| RMSE | 0.043 (.002) | 0.027 (.001) | 0.055 (.002) | ... |
| Mean SD | 0.039 (.001) | 0.026 (.001) | 0.051 (.001) | ... |

Notes: The statistics shown are based on NNE's outputs in the test set (500 datasets). RMSE stands for root mean squared error. SD refers to the standard deviation estimated by NNE. The numbers in parentheses are standard errors for the reported statistics.

the NNE recovers parameter values with reasonable accuracy. The bottom row of Figure 1 plots the distributions of the estimation error, defined as the difference between the NNE estimate for a parameter and true value for that parameter. We see that the distributions in all plots are centered around zero. We omit plotting the estimates for $\beta_3$ to $\beta_{11}$, because these parameters are either firm fixed effects or associated with firm 2-5 and all firms are symmetric. Including these parameters in the figure will not change our conclusions qualitatively. The NNE presented in Figure 1 is trained using the squared-error loss $C_1$. The plots for the NNE trained using the entropy loss $C_2$ are very similar, thus we omit showing them here.

Table 1 provides summary statistics for the estimation errors in the test set. The upper panel reports the NNE trained using $C_1$, whereas the lower panel reports the NNE trained using $C_2$ with a diagonal $\boldsymbol{V}$. We see the mean biases in both panels are all close to zero. Note these biases are with respect to the true parameters drawn from the uniform prior on $\Theta$. The table also reports the root mean squared errors (RMSEs). To put the numbers in perspective, consider an estimator that always outputs the center of each parameter's range in $\Theta$. Given that we set each parameter's range to be an interval of unit length, the RMSE for such an estimator would be $\int_0^1 (t - 1/2)^2 dt \simeq 0.289$. The much smaller RMSEs shown in Table 1 indicate that the NNE recovers parameter values with good accuracy, consistent with what we have seen in Figure 1.

The lower panel of Table 1 also reports the mean standard deviations (SDs) as outputted

by the NNE (as $\sqrt{\text{diag}(\boldsymbol{V})}$). Note that the SD is larger for the parameter for which the RMSE is larger, suggesting that the SDs correctly measure the accuracy of the point estimates by NNE. The overall smaller magnitude of the mean SDs compared to the RMSEs can be attributed to the heteroskedasticity in estimation errors — when a set of random errors have different standard deviations, the RMSE tends to over-estimate the mean of these standard deviations.

### 3.3 Comparison to other estimators

We compare NNE with extremum estimators. The comparison is made by drawing multiple "real" datasets (in the way described at the end of Section 3.1). For each "real" dataset, we obtain three sets of estimates for $\boldsymbol{\theta}$ using NNE, MLE, and GMM, respectively. The implementation of NNE follows our discussion in Section 3.2. The implementations of MLE and GMM follow Section 3.1.

Table 2 reports the RMSEs (root mean squared errors) of the parameter estimates by the three estimators, averaged across 250 "real" datasets. For NNE, $L^*$ denotes the combined size of the training and test sets ($L = 0.9L^*$). For MLE and GMM, $R$ denotes the number of simulations used to evaluate the entry probabilities in each market. All estimators use the same parameter bounds $\Theta$. Again, we focus on parameter $\delta$, $\beta_1$, and $\beta_2$. Including the other parameters in the presentation does not change our conclusions qualitatively.

We discuss Table 2 from two aspects: computational burden and estimation errors. In terms of computational burden, we focus on the simulation cost, that is, the total number of game simulations required to carry out the estimation. For NNE, this cost equals $L^* \times K$, where $K$ is the number of markets. For MLE and GMM, this cost equals $R$ times $K$ times the number of objective function evaluations needed by the optimization routines. Note, compared to MLE, the optimization routine in our implementation of GMM requires a larger number of objective function evaluations because: (i) it is non-gradient-based, and (ii) the optimization routine is run twice, with the results from the first time used to calculate the optimal weighting matrix for the moment conditions.

To be complete, Table 2 also reports the compute time. Note the compute time can vary depending on the code implementation (such as the degree of vectorization) as well as hardware. For NNE, the compute time includes the training of the neural net, which, as we discussed, is a fixed cost that is tied with neither the data size (i.e., $K$) nor the cost of model simulation (e.g., an alternative equilibrium selection rule that makes it harder to compute the

Table 2: RMSEs of Different Estimators in Market Entry

| | $\delta$ | $\beta_1$ | $\beta_2$ | $\beta_3$ - $\beta_{11}$ | Compute time* | Simulation Costs |
|---|---|---|---|---|---|---|
| NNE: | | | | | | |
| $L^* = 1000$ | 0.054 | 0.033 | 0.071 | ... | 11 | $1\times10^6$ |
| | (.003) | (.002) | (.003) | ... | | |
| $L^* = 5000$ | 0.040 | 0.026 | 0.056 | ... | 14 | $5\times10^6$ |
| | (.002) | (.001) | (.003) | ... | | |
| MLE: | | | | | | |
| $R = 10$ | 0.072 | 0.032 | 0.064 | ... | 11 | $7.5\times10^6$ |
| | (.003) | (.001) | (.003) | ... | | |
| $R = 100$ | 0.040 | 0.027 | 0.057 | ... | 33 | $73.9\times10^6$ |
| | (.002) | (.001) | (.003) | ... | | |
| $R = 1000$ | 0.036 | 0.023 | 0.052 | ... | 179 | $704.2\times10^6$ |
| | (.002) | (.001) | (.002) | ... | | |
| GMM: | | | | | | |
| $R = 2$ | 0.104 | 0.037 | 0.060 | ... | 16 | $6.1\times10^6$ |
| | (.005) | (.002) | (.003) | ... | | |
| $R = 20$ | 0.096 | 0.032 | 0.051 | ... | 52 | $82.7\times10^6$ |
| | (.004) | (.002) | (.003) | ... | | |
| $R = 200$ | 0.093 | 0.031 | 0.051 | ... | 261 | $998.4\times10^6$ |
| | (.004) | (.002) | (.003) | ... | | |

Notes: $L^*$ is the combined size of the training and test sets. $R$ is the number of entry games simulated to evaluate the entry probabilities in a market. Simulation cost displays the total number of entry games simulated to carry out the estimation. For NNE, this cost simply equals $L^*$ times $K$. For MLE and GMM, the cost equals $R$ times $K$ times the number of objective function evaluations by optimization routines. The optimization routines bound parameters within $\Theta$ and use the center of $\Theta$ as the starting point. The numbers shown in the table are averaged across 250 "real" datasets. For each "real" dataset, the true parameter values are drawn uniformly from $\Theta$.

*Compute time is in seconds and based on Matlab codes; it can vary depending on the code implementation as well as hardware.

Nash equilibrium). In this sense, the simulation cost should be a more informative measure of computational burden if one wants to extrapolate our results to heavier structural estimation scenarios.

In the third row of Table 2, we configure $R = 10$ for MLE so its simulation cost is on par with that of NNE at $L^* = 5000$. We see the NNE has the smaller RMSEs. In fact, the NNE has comparable or smaller RMSEs than MLE at $R = 100$, though the latter configuration incurs a much higher simulation cost. It is only when $R$ is increased to 1000 that the RMSEs of MLE become smaller than those of NNE.

The intuition behind these results is consistent with our analysis in Section 2. In general, we know MLE is asymptotically efficient. However, this efficiency assumes that the likelihood function is accurately evaluated. But in this case, evaluating the entry probability in the likelihood requires simulations (to integrate over the unobserved profit shocks). To get a sense of the inaccuracy caused by simulations, note that with $J = 5$ firms, there are $2^J = 32$ possible entry outcomes in a market. With $R = 10$, each entry outcome is covered by less than $1/3$ of a simulation, on average. The simulation errors, which require $R \to \infty$ to diminish, introduce inaccuracy in the evaluations of entry probability and thus the likelihood function. Because NNE does not require the evaluation of entry probability, it is not affected by this particular inaccuracy. In addition, the efficiency of MLE is an asymptotic result. With a finite dataset, MLE can be less efficient than some estimators.[13]

The above intuition regarding the simulation errors carries to the comparison between NNE and GMM. We see that NNE with $L^* = 5000$ has smaller RMSEs than GMM at a similar simulation cost ($R = 2$), and holds some advantages even as GMM incurs higher simulation costs ($R = 20$ and $R = 200$). Compared to MLE, an additional issue with GMM is that the estimation accuracy is bounded by the amount of information in the data captured by the moment conditions. This issue remains even as $R \to \infty$ (which is also suggested by the numbers reported in the table). Here, we have used a set of moment conditions that are standard for entry games (see Section 3.1). We explore this issue further in the next Monte Carlo study (Section 4).

NNE has its own source of inaccuracy that stems from using a neural net to learn the estimator mapping. This inaccuracy is not shared by extremum estimators and is a function of the size of the training set (e.g., compare $L^* = 1000$ with $L^* = 5000$ in Table 2). However,

[13] Examples include shrinkage estimators and Bayesian estimator. From Section 2, we know there is a close connection between NNE and Bayesian estimator.

we see that the powerful ability of neural nets to learn functions, commonly demonstrated elsewhere (e.g., classifying images, forecasting data variables), carries over to the parameter estimation problem: In this Monte Carlo study, a simple single-hidden-layer neural net is able to learn an estimator with good performance from a modest-sized training set ($L^* = 5000$).

Finally, we discuss a result on the issue of model mis-specification. Compared to other estimators, NNE might seem more reliant on the structural model being correct because it explicitly uses the model-generated datasets for training. Therefore, a question is how the estimator would behave with a mis-specified model. Our development in Section 2, which establishes an asymptotic equivalence between NNE and Bayesian statistics, suggests that NNE should not be more vulnerable to model mis-specification than traditional methods in general. We examine this theoretical implication with a Monte Carlo exercise under the market entry model. We change the true effect of entries on profits to be in log scale ($-\delta \log N_k$), but still assume the linear model ($-\delta N_k$) in estimation. The estimates for $\delta$ (unreported here) show that the biases in NNE, MLE, and GMM tend to be in the same direction and have very similar magnitudes. This result is consistent with what our theory suggests.

### 3.4 Exploring sources of identification

In this subsection, we apply NNE to model selection. Specifically, we ask NNE to separately identify two different information structures for the entry game, complete information and incomplete information. We also use NNE to explore which data feature enables this separate identification. An important aspect of this exercise is that it shows how machine learning tools, which are usually regarded as black-box predictors, can be used in a way to discover the source of identification for structural models.

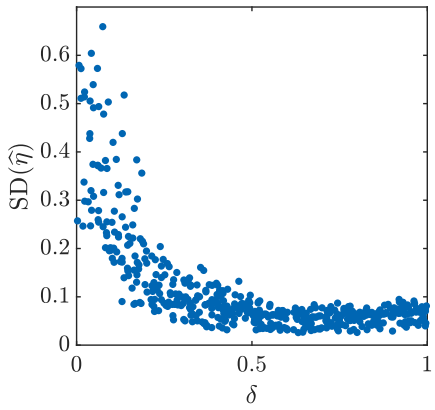
The entry model that we have described so far assumes complete information. Below, we describe the model with incomplete information (Seim, 2006), where each firm $j$'s knowledge of its own shock $\varepsilon_{j,k}$ is private. A firm does not see the profit shocks of competing firms. This game permits a Bayesian Nash equilibrium, where each firm makes the entry decision based on the *probabilities* that other firms will enter. Let $p_{j,k}$ be the probability that firm $j$ will enter market $k$. We have in equilibrium, for any $j$,

$$s_{j,k} = 1 \;\text{ iff }\; \boldsymbol{x}'_{j,k}\boldsymbol{\beta} - \delta \left( 1 + \sum_{i \neq j} p_{i,k} \right) + \varepsilon_{j,k} > 0.$$

Table 3: Summary of NNE in Market Entry with Model Selection

| | $\delta$ | $\beta_1$ | $\beta_2$ | $\beta_3$ - $\beta_{11}$ | $\eta$ |
|---|---|---|---|---|---|
| Mean Bias | 0.001 (.003) | -0.001 (.001) | -0.002 (.003) | ... | -0.015 (.009) |
| RMSE | 0.062 (.002) | 0.034 (.001) | 0.057 (.002) | ... | 0.180 (.010) |
| Mean SD | 0.057 (.001) | 0.032 (.001) | 0.056 (.002) | ... | 0.132 (.005) |

Notes: Same as Table 1.




Notes: The figure plots SD($\hat{\eta}$) against the true parameter value of $\delta$ in the test set. SD($\hat{\eta}$) is the standard deviation of $P(\eta|\boldsymbol{m})$ as estimated by NNE.

Figure 2: When Can the Two Models of Entry be Separated?

Note that the plus 1 term in the parentheses accounts for firm $j$'s own entry. Thus,

$$p_{j,k} = \Phi\left(\boldsymbol{x}'_{j,k}\boldsymbol{\beta} - \delta - \delta\sum_{i\neq j} p_{i,k}\right),$$

where $\Phi$ is the c.d.f. of the standard normal distribution. The equilibrium entry probabilities are given by the vector $\boldsymbol{p}_k$ that solves the above equation for all $j = 1, ..., J$.

Suppose we want to know which model, the complete-information game or the incomplete-information game, provides a better description of a dataset. Two subsequent questions arise. First, *whether* any variation in the data allows us to separate the two models? Second, *which* variation in the data allows us to separate the two models?

We first ask NNE to help answer the first question. To do so, we draw an additional binary parameter $\eta^{(\ell)}$ uniformly from $\{0, 1\}$ when generating the training and test sets. If

$\eta^{(\ell)} = 0$, we simulate $\boldsymbol{s}_k^{(\ell)}$ with the complete-information game. If $\eta^{(\ell)} = 1$, we simulate $\boldsymbol{s}_k^{(\ell)}$ with the incomplete-information game. We construct $\boldsymbol{m}^{(\ell)}$ in the same way as before. We train the neural net to produce an estimate $\widehat{\eta}$ for this binary parameter that selects the information structure in addition to the estimates for other parameters. By our results in Section 2, $\widehat{\eta}$ should reflect $\Pr(\eta = 1|\boldsymbol{m})$, the Bayesian belief that the data moments come from the incomplete-information game.

Table 3 summarizes NNE's performance in the test set. From the last column, we see the RMSE of $\widehat{\eta}$ is substantially smaller than an estimator that always predicts $\eta = 0.5$ (which would have a RMSE of 0.5), implying that NNE is mostly able to tell which of the two models underlies a dataset.

Now we turn to the second question, which data variation separates the two models? We note that NNE has a unique advantage in answering this question: it readily provides the levels of statistical accuracy of $\widehat{\eta}$ across many different datasets (in the test set). We may compare the datasets for which $\text{SD}(\widehat{\eta})$ is small vs. the datasets for which $\text{SD}(\widehat{\eta})$ is large. Because a large $\text{SD}(\widehat{\eta})$ indicates difficulties to identify $\eta$, the comparison may offer us clues for the source of identification for $\eta$.

Figure 2 plots $\text{SD}(\widehat{\eta})$ against the true value of $\delta$ in the test set. Recall $\delta$ is the negative effect of an entry on firm profits. We see it is easier to separate the two models when $\delta$ is large. This pattern points to an important difference between the two models under larger $\delta$. With a relatively large $\delta$, it is much more unlikely to observe many entrants (e.g., $N_k = J$) in the complete-information game than in the incomplete-information game. The reason is that the incomplete-information game permits outcomes with regrets (i.e., a firm finds itself ending up with negative profits), but the complete information game does not. NNE capitalizes on this difference to identify which game is at play.

Note that in order to perform a similar exercise as the one in Figure 2 with MLE or GMM, one would have to run the complete estimation procedure separately for *each* simulated dataset. The total cost will be very high if not prohibitive in heavy estimation problems. For NNE, however, the neural net only needs to incur training once, after which it can be applied to new simulated datasets with virtually zero additional costs.

# 4 Monte Carlo: Continuous Choice

Many applications in marketing and economics focus on consumer decisions. In this section, we describe a Monte Carlo study of a continuous choice model with random coefficients. In a way, it complements the Monte Carlo study in the previous section that focuses on discrete choices under strategic interactions. This continuous choice model features highly nonlinear parameters, including the variances of the random coefficients and the degrees of decreasing marginal effects, which are known to be difficult to estimate accurately.

Random coefficients, or more generally any form of rich unobserved consumer heterogeneity, make NNE a potentially suitable estimator. This is because NNE avoids the integrals over the unobserved heterogeneity that are required to evaluate the likelihood or moment functions.

## 4.1 Model and data setup

Models allowing for continuous choices have been applied in marketing to study purchase quantities (e.g., Chintagunta, 1993; Dubé, 2004; Chan, 2006). We describe a simple continuous-choice model that can be seen as a component of the more complex models used in these applications. Consumer $i$ chooses the consumption quantity of a product, denoted as $y_i$, and the quantity of an outside option, denoted as $y_i^o$. (The model can be extended to allow more than one product aside from the outside option.) Consumer $i$'s utility is specified as

$$U_i(y_i, y_i^o) = \psi_i \log(y_i) + \log(y_i^o),$$

where $\psi_i > 0$ denotes the preference for the product in question. Preference for the outside good is normalized to 1. The log-linear form captures diminishing marginal utility. Let $\boldsymbol{x}_i$ collect the observed characteristics of the consumer. Let $x_{k,i}$, $k = 0, ..., K$ denote the dimensions of $\boldsymbol{x}_i$, among which $x_{0,i} = 1$. Preference for the product $\psi_i$ is parameterized as

$$\psi_i = \exp\left(\widetilde{\beta}_{0,i} + \sum\nolimits_{k=1}^{K} \widetilde{\beta}_{k,i} \cdot x_{k,i}^{\alpha_k}\right).$$

In the above, $\widetilde{\beta}_{k,i} \sim \mathcal{N}(\beta_k, \sigma_k)$ for each $k \in \{0, ..., K\}$. These are random coefficients to capture unobserved heterogeneity across consumers. Parameter $\alpha_k \in [0, 1]$ allows for decreasing marginal effects of $x_{k,i}$. The exponential form ensures that $\psi_i$ is non-negative.

Consumer $i$ maximizes utility $U_i$ under the budget constraint:

$$y_i^o + p_i y_i \leq B_i.$$

For our purposes, it suffices to consider the simplified case where $p_i = 1$ and $B_i = 1$. In this case, a consumer's choices always satisfy $y_i^o = 1 - y_i$, so we only need to track $y_i$ in the estimation.

In the data, one observes the purchased quantity $y_i$ and consumer characteristics $\boldsymbol{x}_i$ for each consumer $i$. Use vector $\boldsymbol{\theta} \equiv (\boldsymbol{\beta}', \boldsymbol{\sigma}', \boldsymbol{\alpha}')'$ to collect all the parameters to be estimated. Continuous choice models are typically estimated using GMM, thanks to the difficulty in evaluating the likelihood (in MLE or Bayesian estimation) — the probability for simulated choices to match an observed continuous outcome $y_i$ is always zero. In general, simulating the likelihood is difficult when outcome variables are continuous. There is an additional reason for us to focus on GMM: it facilitates a more direct comparison with NNE which is also based on moments. To construct the moment conditions for GMM, define residuals $\varepsilon_{1,i}$ and $\varepsilon_{2,i}$ as the differences between the observed and model-predicted choice quantities:

$$\varepsilon_{1,i} = y_i - \mathbb{E}\left(y_i | \boldsymbol{x}_i; \boldsymbol{\theta}\right);$$

$$\varepsilon_{2,i} = y_i^2 - \mathbb{E}\left(y_i^2 | \boldsymbol{x}_i; \boldsymbol{\theta}\right).$$

We include the second-order residual $\varepsilon_{2,i}$ to help estimate the variance parameters, $\boldsymbol{\sigma}$. The moment conditions are constructed by interacting $\boldsymbol{\varepsilon}_i \equiv (\varepsilon_{1,i}, \varepsilon_{2,i})'$ with $(\boldsymbol{x}_i', \sqrt{\boldsymbol{x}_i'}, \log \boldsymbol{x}_i')'$. Here, the nonlinear transformations of $\boldsymbol{x}_i$ are added to help estimate the parameters for the decreasing marginal effects, $\boldsymbol{\alpha}$.[14]

The two conditional expectations in the definitions of $\varepsilon_{1,i}$ and $\varepsilon_{2,i}$ are evaluated with simulations, as $\frac{1}{R}\sum_{r=1}^{R} y_i^{(r)}$ and $\frac{1}{R}\sum_{r=1}^{R} (y_i^{(r)})^2$, where $y_i^{(r)}$, $r = 1, ..., R$ denote the simulated choices of consumer $i$. To simulate a choice for consumer $i$, we draw a set of random coefficients, $\widetilde{\beta}_{0,i}, ..., \widetilde{\beta}_{K,i}$, and solve the associated utility maximization problem.

Similar to our Monte Carlo exercises in Section 3, we use the model to generate a "real" dataset, upon which we apply different estimators to recover the parameter values. To generate a "real" dataset, we draw $\boldsymbol{x}_i$ as a 4 by 1 vector, where $x_{0,i} = 1$ and $x_{1,i}$, $x_{2,i}$, and $x_{3,i}$ are each drawn i.i.d. from the exponential distribution with a mean of 1. The exponential

[14] Alternatively, we have also used $\boldsymbol{x}_i^2$ and $\boldsymbol{x}_i^3$ as the nonlinear transformations to construct moments. They result in larger RMSEs for both NNE and GMM. All of our results hold qualitatively.

Table 4: RMSEs of Different Estimators in Continuous Choice

| | $\beta_1$ | $\sigma_1$ | $\alpha_1$ | Other parameters | Compute time* | Simulation costs |
|---|---|---|---|---|---|---|
| NNE: | | | | | | |
| $L^* = 1000$ | 0.102 | 0.147 | 0.184 | ... | 9.2 | $5\times10^6$ |
| | (.006) | (.007) | (.009) | ... | | |
| $L^* = 5000$ | 0.092 | 0.120 | 0.155 | ... | 22 | $25\times10^6$ |
| | (.006) | (.007) | (.009) | ... | | |
| GMM: | | | | | | |
| $R = 1$ | 0.135 | 0.163 | 0.220 | ... | 14 | $6.8\times10^6$ |
| | (.009) | (.012) | (.018) | ... | | |
| $R = 5$ | 0.125 | 0.151 | 0.183 | ... | 26 | $35.8\times10^6$ |
| | (.008) | (.014) | (.018) | ... | | |
| $R = 50$ | 0.112 | 0.142 | 0.175 | ... | 140 | $377.5\times10^6$ |
| | (.008) | (.013) | (.017) | ... | | |

Notes: For GMM, $R$ is the number of choices simulated per consumer to evaluate her expected choice. Simulation cost displays the total number of choices simulated in order to carry out the estimation. For NNE, this cost equals $L^*$ times $n$. For GMM, this cost equals $R$ times $n$ times the number of objective function evaluations by the optimization routine. The GMM optimization routine bounds parameters within $\Theta$ and uses the center of $\Theta$ as the starting point. The numbers shown are averaged across 250 "real" datasets. For each "real" dataset, the true parameter values are drawn uniformly from $\Theta$.
*Compute time is in seconds and based on Matlab codes; it can vary depending on the code implementation as well as hardware.

distribution ensures that $x_{k,i}$ is positive so that $x_{k,i}^{\alpha_k}$ is well defined. We set the number of consumers $n = 5000$. For the parameter space $\Theta$, we specify $\beta_k \in [-\frac{1}{2}, \frac{1}{2}]$, $\sigma_k \in [0, 1]$, and $\alpha_k \in [0, 1]$ for all $k$. This $\Theta$ allows a wide variety of distributions for $y_i$ over the $[0, 1]$ support (e.g., bell-shaped, heavily skewed towards 0, heavily skewed towards 1).

### 4.2 Implementing NNE and comparing to GMM

For NNE, we construct the training and test sets $\{\boldsymbol{\theta}^{(\ell)}, \boldsymbol{m}^{(\ell)}\}_{\ell=1}^{L^*}$ as follows. First, we draw $\boldsymbol{\theta}^{(\ell)}$ uniformly from the parameter space $\Theta$. Given $\boldsymbol{\theta}^{(\ell)}$, we simulate a purchase quantity $y_i^{(\ell)}$ for each consumer $i$ (by drawing a set of random coefficients and solving the associated utility maximization problem).

We construct $\boldsymbol{m}^{(\ell)}$ in parallel to the moment conditions used in GMM. Specifically, let $\bar{y}_i^{(\ell)}$ denote $y_i^{(\ell)}$ demeaned across $i$. We include in $\boldsymbol{m}^{(\ell)}$ the cross moments between $(\bar{y}_i^{(\ell)}, (\bar{y}_i^{(\ell)})^2)'$ and $(\boldsymbol{x}_i', \sqrt{\boldsymbol{x}_i'}, \log \boldsymbol{x}_i')'$. Like in GMM, the purpose of the square term of $\bar{y}_i^{(\ell)}$ is to help estimate

$\boldsymbol{\sigma}$ and the purpose of the nonlinear transformations of $\boldsymbol{x}_i$ is to help estimate $\boldsymbol{\alpha}$. We also add the mean of $y_i^{(\ell)}$ and $\boldsymbol{x}_i$ to $\boldsymbol{m}^{(\ell)}$. Note there are duplicate elements in $\boldsymbol{m}^{(\ell)}$, but we do not need to remove them because generally neural nets can self-select the useful inputs. We use the same configuration of neural networks as in the market entry game.

We benchmark the performance of NNE against that of GMM. As with the Monte Carlo study for the market entry game, we generate 250 "real" datasets. For each "real" dataset, we obtain two sets of estimates for $\boldsymbol{\theta}$ using NNE and GMM, respectively. Both NNE and GMM use the same parameter space $\Theta$.

Table 4 reports the results. As in Section 3, we shall focus on the simulation cost as a measure for computational burden. This measure counts the total number of model simulations required to carry out an estimation procedure. The minimization routine for GMM requires about a thousand objective function evaluations. As a result, we see that GMM at $R = 1$ incurs a similar simulation cost as NNE at $L^* = 1000$. However, the NNE has smaller RMSEs. As $R$ increases, the RMSEs of GMM become smaller, but the simulation cost increases. Compared to NNE at $L^* = 5000$, GMM at $R = 50$ incurs about ten times the simulation cost but still has larger RMSEs. Overall, the results are consistent with those that we have seen in the market entry game (Table 2).

To be complete, Table 4 also reports the compute time. For NNE, the compute time includes the training time of the neural net. We see NNE can achieve smaller estimation errors with less compute time. However, do note the compute time can vary depending on the code implementation (e.g., the degree of vectorization) as well as hardware.

# 5 Conclusion

In this paper, we propose a novel approach to estimate the parameters of structural econometric models. It takes advantage of the fact that structural models can be used to generate datasets. These generated datasets, together with the parameter values under which they are generated, can be used to train a machine learning tool to "recognize" parameter values from datasets. Different from extremum estimators, the Neural Net Estimator does not require an optimization routine to search over parameter guesses. Instead, it tries to learn a direct mapping from the space of data moments to the space of parameter values. For suitable structural estimation problems, NNE is able to achieve comparable or smaller estimation errors than simulated MLE and GMM, while incurring a significantly lower computational

cost.

The paper leaves several possibilities unexplored, which can open interesting avenues for future research. The first possibility is the application on very large-scale problems with hundreds or more parameters, some of which may be nuisance parameters. It is useful to note that though we have always configured NNE to estimate all the parameters of a structural model, doing so is not always necessary. There is nothing preventing one to configure the neural net to learn estimating only a subset of parameters. One can exclude nuisance parameters from this subset. The second unexplored possibility is that it may be possible to pre-train a generally applicable NNE for some widely used, standard structural models. When applying one of these structural models to a new dataset, researchers can directly use the pre-trained NNE for that model. This approach will significantly reduce the costs for researchers applying structural estimation.

# A Appendix

## A.1 Neural net configuration and training

A neural network is a vector function $\boldsymbol{f}$ parameterized by a set of "edge weights." To facilitate our discussion, we describe a neural network with two hidden layers (adding more layers is not difficult conceptually). Let $\boldsymbol{z}_0$ denote the input vector and let $\widetilde{\boldsymbol{z}}_0 \equiv (\boldsymbol{z}_0', 1)'$ denote $\boldsymbol{z}_0$ augmented with the constant 1. The input $\boldsymbol{z}_0$ is first linearly mapped into $\boldsymbol{w}_1\widetilde{\boldsymbol{z}}_0$ for some matrix $\boldsymbol{w}_1$. Then we apply a nonlinear function $\psi : \mathbb{R} \to \mathbb{R}$ elementwise on $\boldsymbol{w}_1\widetilde{\boldsymbol{z}}_0$ to obtain $\boldsymbol{z}_1 = \psi(\boldsymbol{w}_1\widetilde{\boldsymbol{z}}_0)$. This $\psi$ is known as the activation function. Next, let $\boldsymbol{z}_2 = \psi(\boldsymbol{w}_2\widetilde{\boldsymbol{z}}_1)$ for some matrix $\boldsymbol{w}_2$. Finally, we compute the output of the neural net as $\boldsymbol{f}(\boldsymbol{z}_0) = \boldsymbol{w}_3\widetilde{\boldsymbol{z}}_2$ for some matrix $\boldsymbol{w}_3$.

Figure 3 represents this neural net visually as the input-output operations among a collection of nodes (or units). Each unit of the first layer represents a dimension of the vector $\boldsymbol{z}_0$. The middle two layers represent $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ (i.e., the hidden layers). The last layer represents $\boldsymbol{f}(\boldsymbol{z}_0)$. Each edge (or arrow) assigns a unit as the input of another unit. The entries in matrices $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, and $\boldsymbol{w}_3$ can be represented as weights on these edges (and thus are known as the edge weights of the neural net).

**Configuration** The numbers of layers and units in each layer are the choices of researchers. As we add more layers and units, the neural net becomes more flexible and capable of learning more complex mappings. However, very large neural networks are difficult to train and may "overfit" to the training set, particularly when the size of training set $L$ is relatively small. For one-hidden-layer neural nets, a general rule that we find useful in practice and also supported by theory (see Appendix A.4) is to let the number of hidden units grow roughly proportionally to $\sqrt{L}$. In applications, one chooses among alternative configurations of neural net by looking for optimal performance in the test set (as measured by the value of the loss function in the test set).

In our Monte Carlo studies, we find it sufficient to use a single hidden layer network – using two layers results in about the same test performance but a slightly higher training cost. As to the number of hidden units, it is common to use the powers of 2. For $L^* = 5000$, we use 128 units – using 64 units gives a slightly worse test performance whereas using 256 units gives almost the same test performance. For $L^* = 1000$, we use 64 units as roughly suggested by the $\sqrt{L}$ rule (recall $L = 0.9L^*$). Overall, we find that the parameter estimates as

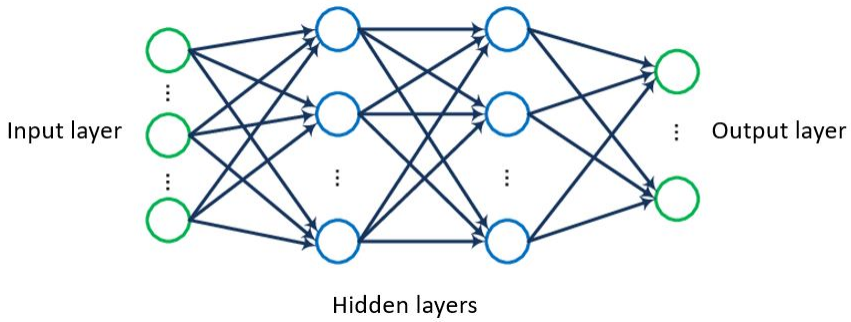



Figure 3: A Neural Network with Two Hidden Layers

well as test performance are largely not sensitive to the neural net configuration, so one does not have to find the precise configuration that optimizes the test performance. In addition, note that different neural net configurations can be trained on the same training set, without the need to simulate additional datasets. This observation means that in heavy structural estimation scenarios where the most significant source of costs is model simulations, choosing the configuration of the neural net should impose a relatively small incremental cost.

**Training** The training of a neural net adjusts the edge weights to minimize a loss function. A basic training algorithm for neural nets is stochastic steepest descent, which basically moves each edge weight in the direction opposite of the derivative of the loss function w.r.t. that weight. These derivatives can be quickly computed using a method called back-propagation. A more recent training algorithm is known as "Adam," which is an extension of the stochastic steepest descent. We use the implementation of Adam in Matlab's deep learning package (the specific Matlab function is `trainNetwork`).

## A.2 Likelihood smoothing

In an entry game where firms enter by the order of profitability, we can smooth the likelihood by exploiting a set of inequalities that directly characterize the sufficient and necessary condition for Nash equilibrium. Note this approach may not be feasible under alternative equilibrium selection rules, which makes MLE more difficult to implement in these cases.

Let $\widetilde{\pi}_{j,k} \equiv \boldsymbol{x}'_{j,k}\boldsymbol{\beta} + \varepsilon_{j,k}$. A given outcome $\boldsymbol{s}_k$ is the Nash equilibrium given a draw of $\boldsymbol{\varepsilon}_k$ if and only if the following inequalities hold. (We omit the cases where some inequalities hold

with equality, which happen with probability zero).

$$\begin{aligned} \widetilde{\pi}_{j,k} - \delta N_k > 0, \quad & \forall j : s_{j,k} = 1, \\ -\widetilde{\pi}_{j,k} + \delta(1 + N_k) > 0, \quad & \forall j : s_{j,k} = 0, \\ \min_{j:s_{j,k}=1} \widetilde{\pi}_{j,k} - \max_{j:s_{j,k}=0} \widetilde{\pi}_{j,k} > 0. & \end{aligned}$$

To smooth the likelihood function, we take the left hand sides of the above $J+1$ inequalities, denote them as $\xi_{1,k}$, ..., $\xi_{J,k}$, and $\xi_{0,k}$, and compute the "extent" to which $\boldsymbol{s}_k$ is the Nash equilibrium as:

$$\frac{1}{1 + \sum_{j=0}^{J} e^{-\lambda \xi_{j,k}}}.$$

In the above, $\lambda$ calibrates the degree of smoothing. For $\lambda \to +\infty$, the above expression is 1 iff $\xi_{j,k} > 0$ for all $j = 0, 1, ..., J$, which means there is no smoothing. For $\lambda = 0$, the above expression becomes a constant, which corresponds to maximal smoothing. The econometric literature suggests choosing $\lambda \propto R^{1/5}$, but the exact choice of $\lambda$ tends to be problematic in practice (Horowitz, 1992; Geweke and Keane, 2001). We choose $\lambda = 3R^{1/5}$ for our application. We reach this choice by inspecting plots of the likelihood function under various values of $\lambda$, and selecting the value under which the likelihood function appears sufficiently smooth yet not overly flat.

### A.3 Parameter range

Before constructing the training set, one needs to decide the parameter range $\Theta$ from which $\boldsymbol{\theta}^{(\ell)}$ is drawn. Here, we present a way to check whether a $\Theta$ is sufficiently large to cover the true parameter values. The basic idea is to examine whether the parameter estimates by NNE are sensitive to expansions of $\Theta$. If they are, then $\Theta$ is likely not covering the true parameter values.

Table 5 displays a sensitivity analysis of $\widehat{\delta}$ in estimating the entry game. We fix a set of true parameter values. The first row displays the NNE estimates when $\Theta$ has a rather small range for $\delta$. This range is displayed in the first column, and it is gradually expanded as we move down the table. We see that the parameter estimates, especially $\widehat{\delta}$ , stabilize in the last two rows. This pattern indicates that the parameter ranges in the last two rows cover the true parameter values, whereas the parameter range in the first row is too small. In fact, the true parameter value for $\delta$ is 0.5.

Table 5: Sensitivity Analysis of NNE to $\Theta$ in Market Entry

| Parameter range | $\widehat{\delta}$ | $\widehat{\beta}_1$ | $\widehat{\beta}_2$ | $\widehat{\beta}_3$ - $\widehat{\beta}_{11}$ |
|---|---|---|---|---|
| $\delta \in [0,\ 0.25]$ | 0.269 (.02) | 0.189 (.02) | 0.266 (.05) | ... |
| $\delta \in [0,\ 0.5]$ | 0.462 (.03) | 0.193 (.03) | 0.222 (.05) | ... |
| $\delta \in [0,\ 0.75]$ | 0.504 (.04) | 0.192 (.03) | 0.213 (.05) | ... |
| $\delta \in [0,\ 1]$ | 0.497 (.04) | 0.198 (.03) | 0.209 (.05) | ... |

Notes: The table displays the NNE estimates for a simulated dataset under different choices of the parameter range $\Theta$. Specifically, the range of each dimension of $\boldsymbol{\beta}$ is $[-0.5, 0.5]$; the range of $\delta$ is given in the first column. The true value of each dimension of $\boldsymbol{\beta}$ is 0.2; the true value of $\delta$ is 0.5. Numbers in parentheses are SDs estimated by NNE.

## A.4 Proofs

We first set up the preliminaries for proving Proposition 1-3. Because our asymptotics ($L \to \infty$) are stated for any fixed data size $n$, our argument can be made either conditional on $\{\boldsymbol{x}_i\}_{i=1}^n$ or by treating this part of the data as non-random. We adopt the non-random approach, which is conceptually simpler. However, all the ensuing argument would carry through with $P$ re-defined as the conditional probability given $\{\boldsymbol{x}_i\}_{i=1}^n$.

Let $P$ denote the probability distribution for $(\boldsymbol{\theta}, \boldsymbol{m})$, where $\boldsymbol{\theta} \sim \mathcal{U}(\Theta)$ and $\boldsymbol{m}|\boldsymbol{\theta}$ follows the distribution implied by the specification of the structural model, $\boldsymbol{y}_i = \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i; \boldsymbol{\theta})$, as well as the researcher-defined rule that constructs $\boldsymbol{m}$ from $\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n$. Then, the members of the training set $\{\boldsymbol{\theta}^{(\ell)}, \boldsymbol{m}^{(\ell)}\}_{\ell=1}^L$ are i.i.d. samples from $P$.

We now work towards a lemma that states the general conditions for a sequence of trained neural networks to converge to a general target function. This lemma forms the basis of all our proofs. We first define the function space where the target function resides. Let $\mathcal{M}$ denote the support for $\boldsymbol{m}$ under $P$. Let $\mathcal{F}$ denote the function space $\mathcal{F} \equiv \{\boldsymbol{f} : \mathcal{M} \to \Delta, \boldsymbol{f} \text{ is continuous}\}$, where $\Delta$ is some subset of a Euclidean space. Let $\|\cdot\|$ denote the 2-norm, that is, $\|\boldsymbol{f}\|^2 = \int \|\boldsymbol{f}(\boldsymbol{m})\|^2 dP(\boldsymbol{m})$. All our propositions share the common goal of learning some target function $\boldsymbol{f}^* \in \mathcal{F}$ that maps $\mathcal{M}$ to the interior of $\Delta$. For example, in proposition 1, $\boldsymbol{f}^*$ is $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$.

Next, we need to define the sequence of trained neural networks that we want to converge to the target function as $L \to \infty$. A proper definition of a trained neural network requires two elements, the loss function and the class of neural nets within which we minimize the loss function. For our purposes, it is sufficiently general to consider loss function constructed

from an individual loss function $h : \Theta \times \Delta \to \mathbb{R}$:

$$C(\boldsymbol{f}) = L^{-1} \sum_{\ell=1}^{L} h\left[\boldsymbol{\theta}^{(\ell)}, \boldsymbol{f}(\boldsymbol{m}^{(\ell)})\right].$$

We are interested in $\widehat{\boldsymbol{f}}_L$ such that $C(\widehat{\boldsymbol{f}}_L) - \inf_{\boldsymbol{f}\in\mathcal{F}_L} C(\boldsymbol{f})$ is a term that converges to 0 in probability as $L \to \infty$. If the infimum can be attained, then we can simply write $\widehat{\boldsymbol{f}}_L \in \operatorname{argmin}_{\boldsymbol{f}\in\mathcal{F}_L} C(\boldsymbol{f})$. Here, $\mathcal{F}_L$ is the class of neural nets in which we minimize the loss function.

We shall focus on the sequence $\mathcal{F}_L$ that is constructed from single-hidden-layer neural nets, which are sufficient for our purpose. Deeper neural nets with more hidden layers are known to provide better approximation capacity in applications. But, studies on their statistical properties in the nonparametric literature are only recent (e.g., Farrell et al., 2019). In comparison, studies on single-hidden-layer neural nets have been more extensive. We adopt the same neural nets used in White (1989; 1990). Let $\psi : \mathbb{R} \to \mathbb{R}$ be any sigmoid function (e.g., logistic function), $r$ denote the dimension of $\boldsymbol{m}$, and

$$\mathcal{F}(q,b) \equiv \left\{ \boldsymbol{f} \in \mathcal{F} : \forall k,\ f_k(\boldsymbol{m}) = w_{1k0} + \sum_{j=1}^{q} w_{1kj}\psi\Big(w_{0j0} + \sum_{i=1}^{r} w_{0ji} m_i\Big), \right.$$
$$\left. \sum_{j=0}^{q} |w_{1kj}| \le b,\ \sum_{j=1}^{q}\sum_{i=0}^{r} |w_{0ji}| \le qb \right\}. \tag{10}$$

In the above formulation, $\mathcal{F}(q,b)$ denotes the single-hidden-layer neural networks with $q$ hidden units and weights $\boldsymbol{w}$ bounded in a way by $b$. A finite $b$ makes $\mathcal{F}(q,b)$ a compact space, which allows the infimum of the loss function to be attainable. We relax this bound as $L$ increases. Specifically, our lemma holds for any sequence $\{\mathcal{F}_L\}_{L=1}^{\infty}$ given by $\mathcal{F}_L = \mathcal{F}(q_L, b_L)$ for any $q_L$ and $b_L$ that grow sufficiently slow. By White (1989), one possible choice is $q_L \propto \sqrt{L}$ and $b_L \propto \log(L)$.

We are in a position to spell out the lemma that connects $\widehat{\boldsymbol{f}}_L$ with $\boldsymbol{f}^*$.

**Lemma 1.** *Let $\mathcal{M}$ and $\Theta$ be compact and let $\Delta$ be compact with a non-empty interior. In addition, assume the following:*

*(i) For any $\epsilon > 0$, $\mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}^*(\boldsymbol{m}))\right] < \inf_{\boldsymbol{f}\in\mathcal{F}:\|\boldsymbol{f}-\boldsymbol{f}^*\|\ge\epsilon} \mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m}))\right]$.*

*(ii) $h$ is continuously differentiable over $\Theta \times \Delta$.*

*Then, we have* $\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| \to 0$ *in probability.* □

*Proof.* of **Lemma** 1: We prove the lemma using Chen (2007). Specifically, we need to show the following conditions are satisfied. Recall that $\|\cdot\|$, when applied to a function, denotes the 2-norm on $\mathcal{F}$.

1. $\mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}^*(\boldsymbol{m}))\right] < \inf_{\boldsymbol{f}\in\mathcal{F}:\|\boldsymbol{f}-\boldsymbol{f}^*\|\geq\epsilon} \mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m}))\right]$ for any $\epsilon > 0$.
2. For any $L < L'$, we have $\mathcal{F}_L \subseteq \mathcal{F}_{L'} \subseteq \mathcal{F}$, and there exists a sequence of functions $\boldsymbol{f}_L \in \mathcal{F}_L$ such that $\|\boldsymbol{f}_L - \boldsymbol{f}^*\| \to 0$.
3. $\mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m}))\right]$ is continuous in $\boldsymbol{f}$ under the norm $\|\cdot\|$ on $\mathcal{F}$.
4. Each $\mathcal{F}_L$ is compact under $\|\cdot\|$.
5. $\sup_{\boldsymbol{f}\in\mathcal{F}_L} \left|C(\boldsymbol{f}) - \mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m}))\right]\right| \to 0$ in probability as $L \to \infty$.

To see these conditions indeed give $\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| \to 0$ in probability, we need to make a series of arguments. To ease notations, let $Q(\boldsymbol{f})$ denote the population loss function $\mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m}))\right]$. Given any $\epsilon > 0$, take $\delta \equiv \inf_{\boldsymbol{f}\in\mathcal{F}:\|\boldsymbol{f}-\boldsymbol{f}^*\|\geq\epsilon} Q(\boldsymbol{f}) - Q(\boldsymbol{f}^*)$. By condition 1, we have $\delta > 0$. By condition 2, we can find a sequence $\boldsymbol{f}_L \in \mathcal{F}_L$ that converges to $\boldsymbol{f}^*$ in $\|\cdot\|$. Given the definition $\widehat{\boldsymbol{f}}_L \in \operatorname{argmin}_{\boldsymbol{f}\in\mathcal{F}_L} C(\boldsymbol{f})$, we must have $C(\widehat{\boldsymbol{f}}_L) \leq C(\boldsymbol{f}_L)$. By condition 5, we have $\Pr[|C(\widehat{\boldsymbol{f}}_L) - Q(\widehat{\boldsymbol{f}}_L)| < \delta/3] \to 1$ and $\Pr\left[|C(\boldsymbol{f}_L) - Q(\boldsymbol{f}_L)| < \delta/3\right] \to 1$ as $L \to \infty$. Therefore, $\Pr[Q(\widehat{\boldsymbol{f}}_L) < Q(\boldsymbol{f}_L) + 2\delta/3] \to 1$. By condition 1, 3, and the construction of $\boldsymbol{f}_L$, for any sufficiently large $L$ we have $Q(\boldsymbol{f}_L) \leq Q(\boldsymbol{f}^*) + \delta/3$. Thus, $\Pr[Q(\widehat{\boldsymbol{f}}_L) < Q(\boldsymbol{f}^*) + \delta] \to 1$ for any $\delta > 0$. Using the definition of $\delta$, we get $\Pr[Q(\widehat{\boldsymbol{f}}_L) < \inf_{\boldsymbol{f}\in\mathcal{F}:\|\boldsymbol{f}-\boldsymbol{f}^*\|\geq\epsilon} Q(\boldsymbol{f})] \to 1$. With $\widehat{\boldsymbol{f}}_L \in \mathcal{F}_L \subseteq \mathcal{F}$, the event of $Q(\widehat{\boldsymbol{f}}_L) < \inf_{\boldsymbol{f}\in\mathcal{F}:\|\boldsymbol{f}-\boldsymbol{f}^*\|\geq\epsilon} Q(\boldsymbol{f})$ implies $\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| < \epsilon$. Thus, we have $\Pr[\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| < \epsilon] \to 1$.

We now check these five conditions are satisfied. The first part of condition 1 is directly provided for with assumption (i). They basically require the population loss $\mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m}))\right]$ to have a unique minimum at $\boldsymbol{f}^*$, and it is not approachable elsewhere.

Condition 2 is shown by Hornik et al. (1989) and White (1990). In general, neural nets are dense sieves for the space of continuous functions (and square integrable functions).

Condition 3 needs the population loss function to be continuous in $\boldsymbol{f}$. Our assumption (ii) says $h$ is Lipschitz continuous (because it is continuously differentiable over a compact set). Therefore, there exist some $c > 0$ such that $\left|h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m})) - h(\boldsymbol{\theta}, \boldsymbol{f}'(\boldsymbol{m}))\right| \leq c\|\boldsymbol{f}(\boldsymbol{m}) - \boldsymbol{f}'(\boldsymbol{m})\|$ for any $(\boldsymbol{\theta}, \boldsymbol{m})$. Thus, $\left|\mathbb{E}h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m})) - \mathbb{E}h(\boldsymbol{\theta}, \boldsymbol{f}'(\boldsymbol{m}))\right| \leq c\mathbb{E}\|\boldsymbol{f}(\boldsymbol{m}) - \boldsymbol{f}'(\boldsymbol{m})\| \leq c\sqrt{\mathbb{E}\|\boldsymbol{f}(\boldsymbol{m}) - \boldsymbol{f}'(\boldsymbol{m})\|^2} = c\|\boldsymbol{f} - \boldsymbol{f}'\|$ (the second step uses Jensen's inequality). Therefore, the population loss is continuous in $\boldsymbol{f}$.

For condition 4, we note that the derivatives of the functions in $\mathcal{F}_L$ are bounded, and thus are Lipschitz continuous with a common Lipschitz constant. By the Arzela-Ascoli theorem, any function sequence in $\mathcal{F}_L$ must have a convergent subsequence. Thus, $\mathcal{F}_L$ is compact.

Condition 5 is a high-level condition. It can be implied by the lower-level condition 3.5M in Chen (2007). We note that condition 5 does not inherit the specific metric between functions used in condition 3.5M. We will use the sup norm $\|\cdot\|_\infty$ as this metric when applying condition 3.5M.

Condition 3.5M(i) requires i.i.d. samples for the computation of $C$. This requirement is satisfied by the construction of our training datasets. In addition, the condition requires $\mathbb{E}\left[\sup_{\boldsymbol{f}\in\mathcal{F}_L}|h(\boldsymbol{\theta},\boldsymbol{f}(\boldsymbol{m}))|\right]<\infty$ for all $L$. This requirement is provided for by our assumption (ii), which says that $h$ is continuous and thus bounded over the compact $\Theta\times\Delta$.

Condition 3.5M(ii) is satisfied if there is a constant $c>0$ such that for any $\boldsymbol{\theta}$ and $\boldsymbol{m}$, we have $|h(\boldsymbol{\theta},\boldsymbol{f}(\boldsymbol{m}))-h(\boldsymbol{\theta},\boldsymbol{f}'(\boldsymbol{m}))|\leq c\|\boldsymbol{f}-\boldsymbol{f}'\|_\infty$. To get this result, note the Euclidean distance between $\boldsymbol{f}(\boldsymbol{m})$ and $\boldsymbol{f}'(\boldsymbol{m})$ is bounded by $\sqrt{\dim(\Delta)}\times\|\boldsymbol{f}-\boldsymbol{f}'\|_\infty$. In addition, assumption (ii) says $h$ is Lipschitz continuous. Thus, we only need to take $c$ as the product of $\sqrt{\dim(\Delta)}$ and the Lipschitz constant of $h$.

Condition 3.5M(iii) says the number of balls required to cover $\mathcal{F}_L$ cannot grow too fast. This is satisfied with any sufficiently slow rates of $q_L$ and $b_L$.

We have checked all the conditions for $\|\widehat{\boldsymbol{f}}_L-\boldsymbol{f}^*\|$ to converge to zero in probability. □

*Proof.* of **Proposition** 1: We apply Lemma 1, with the target function as $\boldsymbol{f}^*=\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$. As to $\Delta$, we can use any compact set that contains $\text{conv}(\Theta)$ in its interior. The reason for taking the convex hull is to accommodate cases where $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ may take values outside $\Theta$, which happens if some dimensions of $\boldsymbol{\theta}$ take discrete values. In particular, with a compact $\Theta$, we may let $\Delta=[\underline{\theta},\overline{\theta}]^p$, where $p$ denotes the dimension of $\boldsymbol{\theta}$ and $\underline{\theta}<\theta_k<\overline{\theta}$ for all $\boldsymbol{\theta}\in\Theta$ and all $k=1,...,p$.

Assumption (ii) of Lemma 1 is immediately provided by the square-error form of $h$, that is, $h(\boldsymbol{\theta},\boldsymbol{f})=\sum_k(\theta_k-f_k)^2$.

As to assumption (i), we need to show $\boldsymbol{f}^*$ minimizes the population loss in $\mathcal{F}$ and further, $\|\boldsymbol{f}-\boldsymbol{f}^*\|\geq\epsilon$ can bound the population loss's value at $\boldsymbol{f}$ away from this minimum. It suffices to consider the case where $\boldsymbol{\theta}$ is single-dimensional, because the population loss simply sums

across each dimension. Note

$$\mathbb{E}(\theta - f(\boldsymbol{m}))^2 = \int \mathbb{E}\left[(\theta - f(\boldsymbol{m}))^2 | \boldsymbol{m}\right] dP(\boldsymbol{m}).$$

Take any $f \neq f^*$ . We have

$$\begin{aligned}
&\mathbb{E}\left[(\theta - f(\boldsymbol{m}))^2 | \boldsymbol{m}\right] - \mathbb{E}\left[(\theta - f^*(\boldsymbol{m}))^2 | \boldsymbol{m}\right] \\
&\qquad = \mathbb{E}\left[-2\theta f(\boldsymbol{m}) + 2\theta f^*(\boldsymbol{m}) + f(\boldsymbol{m})^2 - f^*(\boldsymbol{m})^2 | \boldsymbol{m}\right] \\
&\qquad = -2f^*(\boldsymbol{m})f(\boldsymbol{m}) + 2f^*(\boldsymbol{m})^2 + f(\boldsymbol{m})^2 - f^*(\boldsymbol{m})^2 \\
&\qquad = \left[f(\boldsymbol{m}) - f^*(\boldsymbol{m})\right]^2.
\end{aligned}$$

The second equality uses the fact $f^* = \mathbb{E}(\theta|\boldsymbol{m})$. As a result,

$$\mathbb{E}(\theta - f(\boldsymbol{m}))^2 - \mathbb{E}(\theta - f^*(\boldsymbol{m}))^2 = \|f - f^*\|^2.$$

Thus, the difference between the population loss at $f$ and the the population loss at $f^*$ is exactly $\|f - f^*\|^2$. In particular, $\inf_{f \in \mathcal{F}: \|f - f^*\| \geq \epsilon} \mathbb{E}\left[h(\theta, f(\boldsymbol{m}))\right] = \mathbb{E}\left[h(\theta, f^*(\boldsymbol{m}))\right] + \epsilon^2$. Thus, assumption (i) is satisfied. □

*Proof.* of **Proposition** 2 - part (i): We apply Lemma 1, with $\boldsymbol{f}^* = [\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m}), \mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})]$. Again let $p$ be the dimension of $\boldsymbol{\theta}$. Because $\mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})$ is assumed to be bounded away from zero and $\Theta$ is bounded, we may choose $\underline{v}, \overline{v} > 0$ such that $\underline{v} < \mathbb{V}\text{ar}(\theta_k|\boldsymbol{m}) < \overline{v}$ for all $k$. Let $\underline{\theta}$ and $\overline{\theta}$ be defined as in the proof of Proposition 1, we may choose $\Delta$ to be $[\underline{\theta}, \overline{\theta}]^p \times [\underline{v}, \overline{v}]^p$.

The individual loss function is $h(\boldsymbol{\theta}, \boldsymbol{f}) = \sum_k -\log(V_k) - V_k^{-1}(\theta_k - \mu_k)^2$, where $\{\mu_k, V_k\}_{k=1}^p$ are collected in $\boldsymbol{f}$. Assumption (ii) of Lemma 1 is satisfied with this choice of $h$.

As to assumption (i), we use a similar argument as in the proof of Proposition 1. As before, we consider the case where $p = 1$, so that we may write $\boldsymbol{f} \equiv (\mu, V)$. The cases with $p > 1$ follow because the population loss function again simply sums across each dimension $k$. Fix a small $\epsilon > 0$. Our argument starts by noting

$$\mathbb{E}h(\theta, \boldsymbol{f}(\boldsymbol{m})) = \int \mathbb{E}\left[h(\theta, \boldsymbol{f}(\boldsymbol{m}))|\boldsymbol{m}\right] dP(\boldsymbol{m}).$$

We first examine the part inside the integral. We have

$$\begin{aligned}
&\mathbb{E}\left[h(\theta, \boldsymbol{f}(\boldsymbol{m}))|\boldsymbol{m}\right] - \mathbb{E}\left[h(\theta, \boldsymbol{f}^*(\boldsymbol{m}))|\boldsymbol{m}\right] \\
&\quad = \mathbb{E}\left[-\log(V(\boldsymbol{m})) - \frac{(\theta - \mu(\boldsymbol{m}))^2}{V(\boldsymbol{m})} + \log(V^*(\boldsymbol{m})) + \frac{(\theta - \mu^*(\boldsymbol{m}))^2}{V^*(\boldsymbol{m})} \,\Big|\boldsymbol{m}\right] \\
&\quad = \frac{V^*(\boldsymbol{m})}{V(\boldsymbol{m})} - \log\left[\frac{V^*(\boldsymbol{m})}{V(\boldsymbol{m})}\right] - 1 + \frac{V^*(\boldsymbol{m})}{V(\boldsymbol{m})} \cdot (\mu^*(\boldsymbol{m}) - \mu(\boldsymbol{m}))^2. \qquad (11)
\end{aligned}$$

The last equality uses the definitions $\mu^*(\boldsymbol{m}) = \mathbb{E}(\theta|\boldsymbol{m})$ and $V^*(\boldsymbol{m}) = \mathbb{V}\text{ar}(\theta|\boldsymbol{m})$. We want to show that the integral of (11) is bounded away from zero if $\|\boldsymbol{f} - \boldsymbol{f}^*\| \geq \epsilon$. We do this in two steps.

First, let $Q : \Delta^2 \to \mathbb{R}$ be defined as $Q(\boldsymbol{t}, \boldsymbol{t}') = t_2'/t_2 - \log(t_2'/t_2) - 1 + (t_2'/t_2) \cdot (t_1' - t_1)^2$. Then (11) equals $Q(\boldsymbol{f}(\boldsymbol{m}), \boldsymbol{f}^*(\boldsymbol{m}))$. Define $d(\boldsymbol{m}) \equiv \inf_{\boldsymbol{t} \in \Delta : \|\boldsymbol{t} - \boldsymbol{f}^*(\boldsymbol{m})\| \geq \epsilon/2} Q(\boldsymbol{t}, \boldsymbol{f}^*(\boldsymbol{m}))$. Note $d$ is positive because $Q$ reaches its minimum zero only when $\boldsymbol{t} = \boldsymbol{t}'$. In addition, by Berge's theorem, $d$ is a continuous function. This, together with the compactness of $\mathcal{M}$, gives $\delta \equiv \inf_{\boldsymbol{m} \in \mathcal{M}} d(\boldsymbol{m}) > 0$. We have the result that for any $\boldsymbol{f}$ and $\boldsymbol{m}$, $\|\boldsymbol{f}(\boldsymbol{m}) - \boldsymbol{f}^*(\boldsymbol{m})\| \geq \epsilon/2$ implies (11) is no less than $\delta$.

Second, let $A \equiv \{\boldsymbol{m} \in \mathcal{M} : \|\boldsymbol{f}(\boldsymbol{m}) - \boldsymbol{f}^*(\boldsymbol{m})\| \geq \epsilon/2\}$. We have $\|\boldsymbol{f} - \boldsymbol{f}^*\|^2 = \int \|\boldsymbol{f}(\boldsymbol{m}) - \boldsymbol{f}^*(\boldsymbol{m})\|^2 dP(\boldsymbol{m}) \leq P(A)c^2 + (1 - P(A))\epsilon^2/4$, where $c$ denotes an upper bound for $\|\boldsymbol{f}(\boldsymbol{m}) - \boldsymbol{f}^*(\boldsymbol{m})\|$ implied by the compactness of $\Delta$. A result of this inequality is that $P(A)$ cannot be too small if $\|\boldsymbol{f} - \boldsymbol{f}^*\|$ is not too small. More precisely, there exists some $\tau > 0$ such that for any $\boldsymbol{f}$, $\|\boldsymbol{f} - \boldsymbol{f}^*\| \geq \epsilon$ implies $P(A) \geq \tau$.

Now, pick any $\boldsymbol{f}$ with $\|\boldsymbol{f} - \boldsymbol{f}^*\| \geq \epsilon$, we have

$$\begin{aligned}
&\mathbb{E}\left[h(\theta, \boldsymbol{f}(\boldsymbol{m}))\right] - \mathbb{E}\left[h(\theta, \boldsymbol{f}^*(\boldsymbol{m}))\right] \\
&\qquad = \int \left\{\mathbb{E}\left[h(\theta, \boldsymbol{f}(\boldsymbol{m}))|\boldsymbol{m}\right] - \mathbb{E}\left[h(\theta, \boldsymbol{f}^*(\boldsymbol{m}))|\boldsymbol{m}\right]\right\} dP(\boldsymbol{m}) \\
&\qquad \geq \int_A \delta dP(\boldsymbol{m}) \\
&\qquad \geq \tau\delta.
\end{aligned}$$

In words, the population loss function at $\boldsymbol{f}$ is at least $\tau\delta > 0$ larger than the population loss at $\boldsymbol{f}^*$, where neither $\tau$ or $\delta$ depends on $\boldsymbol{f}$. Therefore, assumption (i) is satisfied.

Part (ii): We again apply Lemma 1. By the condition of the proposition, we can find

some $\underline{\lambda} > 0$ such that the smallest eigenvalue of $\mathbb{C}\text{ov}(\boldsymbol{\theta}|\boldsymbol{m})$ is larger than $\underline{\lambda}$. Let $\nabla$ denote the subset of Euclidean space that contains all possible values of the lower triangular part a covariance matrix $\boldsymbol{V}$ such that $\underline{v} \leq \text{diag}(\boldsymbol{V}) \leq \overline{v}$ and the smallest eigenvalue of $\boldsymbol{V}$ is no less than $\underline{\lambda}$. Note $\nabla$ is a compact subset of the convex cone that contains all the positive definite matrices. We can then take $\Delta = [\underline{\theta}, \overline{\theta}]^p \times \nabla$.

The specification of $h(\boldsymbol{\theta}, \boldsymbol{f}) = -\log(|\boldsymbol{V}|) - (\boldsymbol{\theta} - \boldsymbol{\mu})'\boldsymbol{V}^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu})$, where $\boldsymbol{f} = (\boldsymbol{\mu}, \boldsymbol{V})$, satisfies assumption (ii) in Lemma 1. Note, in particular, that matrix inverse is a continuously differentiable operation on positive definite matrices.

The proof to show assumption (i) is satisfied can use a similar argument as the proof for part (i). The target function $\boldsymbol{f}^*$ is $\boldsymbol{\mu}^* = \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ and $\boldsymbol{V}^* = \mathbb{C}\text{ov}(\boldsymbol{\theta}|\boldsymbol{m})$. Again we note the population loss can be written as

$$\mathbb{E}h(\theta, \boldsymbol{f}(\boldsymbol{m})) = \int \mathbb{E}\left[h(\theta, \boldsymbol{f}(\boldsymbol{m}))|\boldsymbol{m}\right] dP(\boldsymbol{m}),$$

where the integrand satisfies, with $\boldsymbol{f} = (\boldsymbol{\mu}, \boldsymbol{V})$,

$$\begin{aligned}
&\mathbb{E}\left[h(\boldsymbol{\theta}; \boldsymbol{f}(\boldsymbol{m}))\middle|\boldsymbol{m}\right] - \mathbb{E}\left[h(\boldsymbol{\theta}; \boldsymbol{f}^*(\boldsymbol{m}))\middle|\boldsymbol{m}\right] \\
&\qquad = \mathbb{E}\left[\log(|\boldsymbol{V}^*(\boldsymbol{m})|) + (\boldsymbol{\theta} - \boldsymbol{\mu}^*(\boldsymbol{m}))'\boldsymbol{V}^*(\boldsymbol{m})^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}^*(\boldsymbol{m})) \,\middle|\, \boldsymbol{m}\right] \\
&\qquad\quad - \mathbb{E}\left[\log(|\boldsymbol{V}(\boldsymbol{m})|) + (\boldsymbol{\theta} - \boldsymbol{\mu}(\boldsymbol{m}))'\boldsymbol{V}(\boldsymbol{m})^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}(\boldsymbol{m})) \,\middle|\, \boldsymbol{m}\right]. \qquad (12)
\end{aligned}$$

Because $\boldsymbol{\theta}$ enters the above expectations in a quadratic way, (12) depends on the first and second moments, but not the higher moments or distributional form, of $P(\boldsymbol{\theta}|\boldsymbol{m})$. These first and second moments are given by $\boldsymbol{f}^*(\boldsymbol{m})$. Therefore, we can write a function $Q : \Delta^2 \to \mathbb{R}$ like in the proof of part (i) such that $Q(\boldsymbol{f}(\boldsymbol{m}), \boldsymbol{f}^*(\boldsymbol{m}))$ equals (12) for any $\boldsymbol{f}$ and $\boldsymbol{m}$. This $Q$ does not depend on the distributional form of $P(\boldsymbol{\theta}|\boldsymbol{m})$, and it is continuous over $\Delta^2$ (note both the matrix determinant and matrix inverse are continuous operations).

In addition, $\min_{\boldsymbol{t} \in \Delta} Q(\boldsymbol{t}, \boldsymbol{f}^*(\boldsymbol{m})) = 0$ and is achieved at $\boldsymbol{t} = \boldsymbol{f}^*(\boldsymbol{m})$. To see this, suppose $P(\boldsymbol{\theta}|\boldsymbol{m})$ is normal for a moment. By (12), we see $Q(\boldsymbol{t}, \boldsymbol{f}^*(\boldsymbol{m}))$ is the Kullback–Leibler divergence from a normal density parameterized by $\boldsymbol{t} \in \Delta$ to the normal $P(\boldsymbol{\theta}|\boldsymbol{m})$. As a result, $Q(\boldsymbol{t}, \boldsymbol{f}^*(\boldsymbol{m}))$ as a function of $\boldsymbol{t}$ is always positive except when $\boldsymbol{t}$ takes the mean and covariance of $P(\boldsymbol{\theta}|\boldsymbol{m})$, that is, $\boldsymbol{t} = \boldsymbol{f}^*(\boldsymbol{m})$. However, because $Q$ does not depend on the distributional form of $P(\boldsymbol{\theta}|\boldsymbol{m})$, this result holds for non-normal $P(\boldsymbol{\theta}|\boldsymbol{m})$ as well.

With these properties of $Q$ established, the rest of the argument follows that in the proof

for part (i). So we shall not repeat it here. □

*Proof.* of **Proposition** 3: We apply Lemma 1, with $\Delta = \Gamma$. With this choice of $\Delta$, the target $\boldsymbol{f}^*$ as defined in the proposition is a member of $\mathcal{F}$ iff it is continuous. We will show $\boldsymbol{f}^*$ is continuous below.

Assumption (ii) of Lemma 1 is satisfied with $h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m})) = -\log \phi(\boldsymbol{\theta}; \boldsymbol{f}(\boldsymbol{m}))$ and condition (iii) of the proposition.

We proceed to verify assumption (i). To ease notation, let $Q(\boldsymbol{\gamma}, \boldsymbol{m}) \equiv \mathbb{E}\left[-\log \phi(\boldsymbol{\theta}; \boldsymbol{\gamma}) \middle| \boldsymbol{m}\right]$. By the definition of Kullback–Leibler divergence, we have

$$\mathcal{KL}\left[P(\boldsymbol{\theta}|\boldsymbol{m}) \parallel \phi(\boldsymbol{\theta}; \boldsymbol{\gamma})\right] = Q(\boldsymbol{\gamma}, \boldsymbol{m}) - \mathbb{E}\left[-\log P(\boldsymbol{\theta}|\boldsymbol{m}) \middle| \boldsymbol{m}\right].$$

Note the second term on the right side does not involve $\boldsymbol{\gamma}$. Therefore, by condition (v), we know $\boldsymbol{f}^*(\boldsymbol{m})$ is equal to $\operatorname{argmin}_{\gamma \in \Gamma} Q(\boldsymbol{\gamma}, \boldsymbol{m})$ for every $\boldsymbol{m}$. Now, with the continuity provided by condition (iv), Berge's theorem says that $\boldsymbol{f}^*$ is a continuous function.

Next, fix any small $\epsilon > 0$. Consider the following function $d$:

$$d(\boldsymbol{m}) \equiv \inf_{\boldsymbol{\gamma} \in \Gamma : \|\boldsymbol{\gamma} - \boldsymbol{f}^*(\boldsymbol{m})\| \geq \epsilon/2} Q(\boldsymbol{\gamma}, \boldsymbol{m}) - Q(\boldsymbol{f}^*(\boldsymbol{m}), \boldsymbol{m}).$$

Again by Berge's theorem, we know $d$ is continuous. Because $\boldsymbol{f}^*(\boldsymbol{m})$ is the unique point in $\Gamma$ that minimizes $Q(\boldsymbol{\gamma}, \boldsymbol{m})$, we also have $d > 0$. As a result, $\inf_{\boldsymbol{m} \in \mathcal{M}} d(\boldsymbol{m})$ is attainable and positive, which we denote as $\delta > 0$.

Let $A \equiv \{\boldsymbol{m} \in \mathcal{M} : \|\boldsymbol{f}(\boldsymbol{m}) - \boldsymbol{f}^*(\boldsymbol{m})\| \geq \epsilon/2\}$. Using the same argument in the proof of Proposition 2 part (i), we can find some $\tau > 0$ such that for any $\boldsymbol{f}$, $\|\boldsymbol{f} - \boldsymbol{f}^*\| \geq \epsilon$ implies $P(A) \geq \tau$.

Now for any $\boldsymbol{f}$ such that $\|\boldsymbol{f} - \boldsymbol{f}^*\| \geq \epsilon$, we have

$$\begin{aligned}
&\mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m}))\right] - \mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}^*(\boldsymbol{m}))\right] \\
&\qquad = \int \Big\{ Q\left[\boldsymbol{f}(\boldsymbol{m}), \boldsymbol{m}\right] - Q\left[\boldsymbol{f}^*(\boldsymbol{m}), \boldsymbol{m}\right] \Big\} dP(\boldsymbol{m}) \\
&\qquad \geq \int_A \Big\{ Q\left[\boldsymbol{f}(\boldsymbol{m}), \boldsymbol{m}\right] - Q\left[\boldsymbol{f}^*(\boldsymbol{m}), \boldsymbol{m}\right] \Big\} dP(\boldsymbol{m}) \\
&\qquad \geq \int_A d(\boldsymbol{m}) \cdot dP(\boldsymbol{m}) \\
&\qquad \geq \tau\delta.
\end{aligned}$$

In words, with $\|\boldsymbol{f} - \boldsymbol{f}^*\| \geq \epsilon$, the population loss function at $\boldsymbol{f}$ is at least $\tau\delta > 0$ larger than the population loss at $\boldsymbol{f}^*$. Thus, assumption (i) is satisfied. □

Lastly, we provide details for the alternative two-step approach of approximating the posterior variance. Recall we have $\widehat{\boldsymbol{f}}$ trained using $C_1$ that converges to $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$. Next, if we can train a second neural net $\widetilde{\boldsymbol{f}}$ that converges to $\mathbb{E}(\boldsymbol{\theta}^2|\boldsymbol{m})$, then we can use $\widetilde{\boldsymbol{f}} - \widehat{\boldsymbol{f}}^2$ for $\mathbb{V}\mathrm{ar}(\boldsymbol{\theta}|\boldsymbol{m})$. This idea is similar to Hardle and Tsybakov (1997) that use a two-step approach to estimate conditional variance in the context of kernel regressions. In the proposition below, $\widetilde{\boldsymbol{f}}_L$ denotes the minimizer of the loss function $L^{-1}\sum_{\ell=1}^{L}\sum_k[(\theta_k^{(\ell)})^2 - f_k(\boldsymbol{m}^{(\ell)})]^2$ with the appropriate $\mathcal{F}_L$.

**Proposition 4.** *Suppose the conditions of Proposition 1 hold and* $\mathbb{V}\mathrm{ar}(\boldsymbol{\theta}|\boldsymbol{m})$ *is continuous in* $\boldsymbol{m}$*. Then* $\|\widetilde{\boldsymbol{f}}_L - \widehat{\boldsymbol{f}}_L^2 - \mathbb{V}\mathrm{ar}(\boldsymbol{\theta}|\boldsymbol{m})\| \to 0$ *in probability as* $L \to \infty$*.* □

*Proof.* of **Proposition** 4: First, because $\Theta$ is assumed to be compact, the set $\Theta^2 \equiv \{\boldsymbol{\theta}^2 : \boldsymbol{\theta} \in \Theta\}$ is also compact. So we can repeat the same argument as in the proof for Proposition 1 to show that $\|\widetilde{\boldsymbol{f}}_L - \mathbb{E}(\boldsymbol{\theta}^2|\boldsymbol{m})\| \to 0$ in probability.

Let $c$ be a positive number such that $\Theta \subseteq [-c, c]^p$, where $p$ is the dimension of $\boldsymbol{\theta}$. For $p = 1$, we have for any $\boldsymbol{m}$, $|\widehat{f}_L(\boldsymbol{m})^2 - \mathbb{E}(\theta|\boldsymbol{m})^2| = |\widehat{f}_L(\boldsymbol{m}) + \mathbb{E}(\theta|\boldsymbol{m})| \cdot |\widehat{f}_L(\boldsymbol{m}) - \mathbb{E}(\theta|\boldsymbol{m})| \leq 2c|\widehat{f}_L(\boldsymbol{m}) - \mathbb{E}(\theta|\boldsymbol{m})|$. Extending this result to $p \geq 1$ and taking the integral over $\mathcal{M}$, we have $\|\widehat{\boldsymbol{f}}_L^2 - \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})^2\| \leq 2c\|\widehat{\boldsymbol{f}}_L - \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})\|$. Therefore, Proposition 1 implies $\|\widehat{\boldsymbol{f}}_L^2 - \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})^2\| \to 0$ in probability as well.

Next, we use the fact $\mathbb{V}\mathrm{ar}(\boldsymbol{\theta}|\boldsymbol{m}) = \mathbb{E}(\boldsymbol{\theta}^2|\boldsymbol{m}) - \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})^2$. By the triangle inequality, we have $\|\widetilde{\boldsymbol{f}}_L - \widehat{\boldsymbol{f}}_L^2 - \mathbb{V}\mathrm{ar}(\boldsymbol{\theta}|\boldsymbol{m})\| \leq \|\widetilde{\boldsymbol{f}}_L - \mathbb{E}(\boldsymbol{\theta}^2|\boldsymbol{m})\| + \|\widehat{\boldsymbol{f}}_L^2 - \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})^2\|$. Our proof is completed by noting both terms on the right side of this inequality converge to zero in probability. □

## References


Athey, Susan (2018) "The impact of machine learning on economics," in *The economics of artificial intelligence: An agenda*: University of Chicago Press, pp. 507–547.

Bajari, Patrick, C Lanier Benkard, and Jonathan Levin (2007) "Estimating dynamic models of imperfect competition," *Econometrica*, Vol. 75, No. 5, pp. 1331–1370.

Berry, Steven (1992) "Estimation of a Model of Entry in the Airline Industry," *Econometrica: Journal of the Econometric Society*, pp. 889–917.

Borkovsky, Ron, Avi Goldfarb, Avery Haviv, and Sridhar Moorthy (2017) "Measuring and Understanding Brand Value in a Dynamic Model of Brand Management," *Marketing Science*.

Chan, Tat Y (2006) "Estimating a continuous hedonic-choice model with an application to demand for soft drinks," *The Rand journal of economics*, Vol. 37, No. 2, pp. 466–482.

Chen, XiaoHong (2007) "Large Sample Sieve Estimation of Semi-nonparametric Models," *Handbook of Econometrics*, Vol. 6B.

Chen, Xu and Zhipeng Liao (2015) "Select the valid and relevant moments: An information-based LASSO for GMM with many moments," *Journal of Econometrics*, Vol. 186, No. 2.

Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins (2018) "Double/debiased machine learning for treatment and structural parameters," *Econometrics Journal*, Vol. 21(1).

Chintagunta, Pradeep K (1993) "Investigating purchase incidence, brand choice and purchase quantity decisions of households," *Marketing Science*, Vol. 12, No. 2, pp. 184–208.

Chiong, Khai and Matt Shum (2019) "Random Projection Estimation of Discrete-Choice Models with Large Choice Sets," *Management Science*, Vol. 65, No. 1, pp. 256–271.

Collard-Wexler, Allan (2013) "Demand Fluctuations in the Ready-Mix Concrete Industry," *Econometrica*, Vol. 81, No. 3.

Dubé, Jean-Pierre (2004) "Multiple discreteness and product differentiation: Demand for carbonated soft drinks," *Marketing Science*, Vol. 23, No. 1, pp. 66–81.

Ellickson, Paul B and Sanjog Misra (2011) "Structural workshop paper—Estimating discrete games," *Marketing Science*, Vol. 30, No. 6, pp. 997–1010.

Farrell, Max, Tengyuan Liang, and Sanjog Misra (2019) "Deep Neural Networks for Estimation and Inference," *Available at arxiv.org/abs/1809.09953*.

Gelman, Andrew, John B. Carlin, Hal S. Stern, and Donald B. Rubin (2004) "Bayesian Data Analysis."

Geweke, John and Michael Keane (2001) "Computationally intensive methods for integration in econometrics," in *Handbook of econometrics*, Vol. 5: Elsevier, pp. 3463–3568.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016) "NIPS 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*.

Gourieroux, Christian, Alain Monfort, and Eric Renault (1993) "Indirect inference," *Journal of applied econometrics*, Vol. 8, No. S1, pp. S85–S118.

Hardle, W. and A. Tsybakov (1997) "Local polynomial estimators of the volatility function in nonparametric autoregression," *Journal of Econometrics*, Vol. 81.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989) "Multilayer feedforward networks are universal approximators," *Neural networks*, Vol. 2, No. 5, pp. 359–366.

Horowitz, Joel L (1992) "A smoothed maximum score estimator for the binary response model," *Econometrica: journal of the Econometric Society*, pp. 505–531.

Kaji, Tetsuya, Elena Manresa, and Guillaume Pouliot (2020) "An adversarial approach to structural estimation," *arXiv preprint arXiv:2007.06169*.

Kim, Jae Young (2002) "Limited information likelihood and Bayesian analysis," *Journal of Econometrics*, Vol. 107.

Kleinberg, Jon, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan (2017) "Human Decisions and Machine Predictions," *Quarterly Journal of Economics*, Vol. 133, No. 1.

Lewis, Greg and Vasilis Syrgkanis (2018) "Adversarial generalized method of moments," *arXiv preprint arXiv:1803.07164*.

Liu, Xiao, Dokyun Lee, and Kannan Srinivasan (2019) "Large-scale cross-category analysis of consumer review content on sales conversion leveraging deep learning," *Journal of Marketing Research*, Vol. 56, No. 6, pp. 918–943.

Pakes, Ariel, Michael Ostrovsky, and Steven Berry (2007) "Simple estimators for the parameters of discrete dynamic games (with entry/exit examples)," *the RAND Journal of Economics*, Vol. 38, No. 2, pp. 373–399.

Seim, Katja (2006) "An empirical model of firm entry with endogenous product-type choices," *The RAND Journal of Economics*, Vol. 37, No. 3, pp. 619–640.

Su, Che-Lin and Kenneth L Judd (2012) "Constrained optimization approaches to estimation of structural models," *Econometrica*, Vol. 80, No. 5, pp. 2213–2230.

Tamer, Elie (2010) "Partial Identification in Econometrics," *Annual Review in Economics*, Vol. 2, No. 1.

Timoshenko, Artem and John R Hauser (2019) "Identifying customer needs from user-generated content," *Marketing Science*, Vol. 38, No. 1, pp. 1–20.

Wager, Stefan and Susan Athey (2018) "Estimation and inference of heterogeneous treatment effects using random forests," *Journal of the American Statistical Association*, Vol. 113, No. 523, pp. 1228–1242.

White, Halbert (1982) "Maximum likelihood estimation of misspecified models," *Econometrica: Journal of the Econometric Society*, pp. 1–25.

——— (1989) "Learning in Artificial Neural Networks: A Statistical Perspective," *Neural Computation*, Vol. 1.

——— (1990) "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings," *Neural Networks*, Vol. 3.

Yoganarasimhan, Hema (2020) "Search Personalization using Machine Learning," *Management Science*, Vol. 66, No. 3.

Zhang, Shunyuan, Nitin Mehta, Param Vir Singh, and Kannan Srinivasan (2019) "Can Lower-quality Images Lead to Greater Demand on Airbnb?" *Working Paper*.

Zhu, Yuting, Duncan Simester, Jonathan A Parker, and Antoinette Schoar (2020) "Dynamic Marketing Policies: Constructing Markov States for Reinforcement Learning," *Available at SSRN 3633870*.