Estimating Parameters of Structural Models Using Neural Networks

Yanhao 'Max' Wei and Zhenling Jiang*

May 1, 2024

Abstract

We study an alternative use of machine learning. We train neural nets to provide the parameter estimate of a given (structural) econometric model, e.g., discrete choice, consumer search. Training examples consist of datasets generated by the econometric model under a range of parameter values. The neural net takes the moments of a dataset as input and tries to recognize the parameter value underlying that dataset. Besides the point estimate, the neural net can also output statistical accuracy. This neural net estimator (NNE) tends to limited-information Bayesian posterior as the number of training datasets increases. We apply NNE to a consumer search model. It gives more accurate estimates at lighter computational costs than the prevailing approach. NNE is also robust to redundant moment inputs. In general, NNE offers the most benefits in applications where other estimation approaches require very heavy simulation costs. We provide code at: `https://nnehome.github.io`.

Keywords: neural networks, machine learning, structural estimation, simulation costs, redundant moments, sequential search.

*Marshall School of Business, University of Southern California and Wharton School, University of Pennsylvania. Email: yanhaowe@usc.edu; zhenling@wharton.upenn.edu. We thank the constructive comments from Eric Bradlow, Tat Chan, Vineet Kumar, Greg Lewis, Laura Liu, Nitin Mehta, Aviv Nevo, Holger Sieg, Andrey Simonov, K. Sudhir, Raphael Thomadsen, Kosuke Uetake, Shunyuan Zhang, and the participants at the Northwestern marketing seminar, UCLA marketing seminar, Columbia marketing camp, Junior faculty forum at WashU, Yale marketing seminar, Rochester marketing seminar, Tech Adoption and Human-Algorithm Interaction Workshop at Harvard, 42nd Marketing Science Conference, QME 2020 Conference, and SICS 2021. Max thanks Sam Boysel for research assistance.

1 Introduction

Machine learning algorithms are increasingly applied in empirical research of marketing and economics. There are two streams of applications: (i) processing traditionally intractable data, such as texts and images, and (ii) building more flexible economic models, e.g., predicting user choices based on user attributes and histories.[1] At the core of these applications is the exceptional capability of machine learning algorithms to learn functions. An example is object recognition from images, which uses neural nets to learn a mapping from image pixels to the object in the image (e.g., aircraft, boat, cat).

This paper studies an alternative use of machine learning algorithms that capitalizes on their ability to learn functions. We train a machine learning model to "recognize" the parameter value of a given econometric model (e.g., discrete choice, consumer search, games). In this paper, we focus on shallow neural nets as the machine learning model.[2] We construct training examples by using the econometric model to simulate datasets under a range of parameter values. The neural net takes a dataset as input and tries to recognize the parameter value underlying that dataset. We refer to this approach as the neural net estimator (NNE).

As such, NNE is a tool that exploits existing machine learning techniques to estimate current econometric models. It is particularly helpful in structural estimation. Structural models are useful for empirical research (e.g., enabling counterfactuals) but can be very difficult to estimate. Particularly, as larger and richer data allow for more complex structural models, many of them require increasingly heavy simulations to evaluate the objective function for estimation. The objective function can also be difficult to optimize (e.g., non-smooth) or code. As we will show, in such problems NNE can provide large computational savings and more accurate estimates, which makes it a useful addition to the toolbox for empirical research.

Section 2 formulates NNE in detail. We start with a fairly general setting of structural estimation. Let $i \in 1, ..., n$ index the individuals in data (which may or may not be i.i.d.) Let $\boldsymbol{x}_i$ denote the observed attributes of i and $\boldsymbol{y}_i$ denote the outcome of i. Let $\boldsymbol{x} \equiv \{\boldsymbol{x}_i\}_{i=1}^n$ and $\boldsymbol{y} \equiv \{\boldsymbol{y}_i\}_{i=1}^n$. So $\{\boldsymbol{y}, \boldsymbol{x}\}$ denotes a dataset. An econometric model specifies $\boldsymbol{y}$ as a function of $\boldsymbol{x}$, a set of unobservables $\boldsymbol{\varepsilon}$, and parameter $\boldsymbol{\theta}$. We write $\boldsymbol{y} = \boldsymbol{q}(\boldsymbol{x}, \boldsymbol{\varepsilon}; \boldsymbol{\theta})$. Examples of $\boldsymbol{q}$ are discrete choices, games, diffusion on networks, etc. To estimate $\boldsymbol{\theta}$, common approaches include MLE and GMM. In many applications, integrals over the unobservable $\boldsymbol{\varepsilon}$ are evaluated via simulations, which gives rise to simulated maximum likelihood (SMLE) and simulated method of moments (SMM).

NNE offers an alternative approach to estimate $\boldsymbol{\theta}$. We train a neural net to recognize the value of $\boldsymbol{\theta}$ when it is given a dataset generated by the econometric model under that value of $\boldsymbol{\theta}$. To construct training examples, NNE only requires that it is possible to simulate the outcome $\boldsymbol{y}$ using the econometric model. Specifically, let $\ell = 1, ..., L$ index training examples. For each ℓ, we draw $\boldsymbol{\theta}^{(\ell)}$ from a parameter space. Then, we draw $\boldsymbol{\varepsilon}^{(\ell)}$ and compute $\boldsymbol{y}^{(\ell)}$ by the econometric model, i.e.,

[1] See, e.g., Timoshenko and Hauser (2019), Liu et al. (2019), Chiong and Shum (2019), Zhu et al. (2020), Yoganarasimhan (2020), Zhang and Luo (2023), Yoganarasimhan et al. (2023). Also see Athey (2018) for a review.

[2] In principle, one may choose other machine learning models. We find neural nets to perform well in applications.

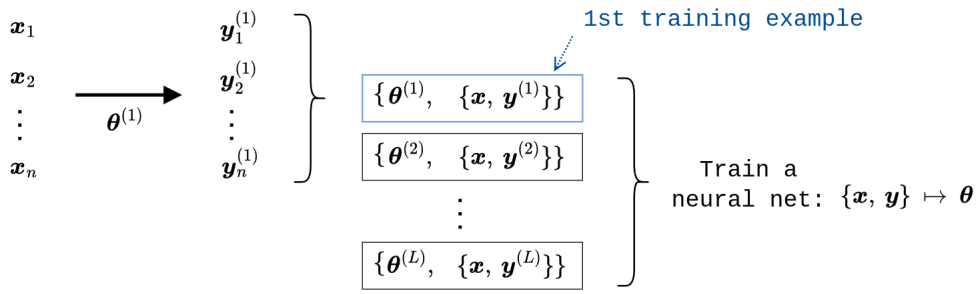


Figure 1: Training Examples in NNE

$\boldsymbol{y}^{(\ell)} = \boldsymbol{q}(\boldsymbol{x}, \boldsymbol{\varepsilon}^{(\ell)}; \boldsymbol{\theta}^{(\ell)})$. The collection $\{\boldsymbol{\theta}^{(\ell)}, \{\boldsymbol{y}^{(\ell)}, \boldsymbol{x}\}\}$ constitutes a training example. An illustration is given in Figure 1. Additional examples can be generated for validation purposes. Finally, we apply the trained neural net to the real data to get an estimate for $\boldsymbol{\theta}$.

We focus on two variations of the above approach. First, we allow using data moments (instead of the entire dataset) as the input to the neural net. Examples of moments include the mean of $\boldsymbol{y}_i$, cross-covariance between $\boldsymbol{y}_i$ and $\boldsymbol{x}_i$, etc. Economic intuition often provides guidance for choosing moments. The reduction of a dataset into moments allows us to use simpler neural nets that are computationally easy to train. Second, we configure the neural net so that it outputs not only point estimate of $\boldsymbol{\theta}$ but also a measure of statistical accuracy for the point estimate.

In Section 2.4, we borrow from the econometric literature on properties of neural nets to provide several theoretical properties for NNE. Let $\boldsymbol{m}$ collect the data moments. As the size of training set $L \to \infty$, the point estimate by NNE converges to $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$. In Bayesian language, $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ is known as the limited-information posterior mean. Here, "limited-information" signifies that the posterior is conditioned on the moments instead of the entire data. Further, the measure of statistical accuracy given by NNE converges to $\mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})$ or $\mathbb{C}\text{ov}(\boldsymbol{\theta}|\boldsymbol{m})$.

One implication of these results is robustness to redundant moments. To see it, note that in a given application, the accuracy of $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ in estimating $\boldsymbol{\theta}$ will not decrease as $\boldsymbol{m}$ includes more moments. Thus, with a sufficiently large L, NNE will not become less accurate as $\boldsymbol{m}$ includes more moments. This is different from SMM, where even with sufficiently many simulations, redundant moments are known to increase finite-sample biases.[3] Intuitively, this difference is because NNE can learn from training examples whether a moment contributes to recognizing $\boldsymbol{\theta}$. SMM and GMM lack such a mechanism (optimal weighting matrix accounts for only the variance-covariance *among* moments). As such, NNE gives researchers more flexibility to include moments for estimation.

We study NNE in two applications. Section 3 uses a simple AR(1) model to visualize the working of NNE and its differences from SMM. The simplicity of the AR(1) model also allows us to see whether a moment is redundant. As we include redundant or almost redundant moments, the root mean square error (RMSE) of GMM increases substantially. In contrast, the increases in the RMSE of NNE are much smaller.

[3]See Chen and Liao (2015) and references there within. Also see Newey (2007).

Section 4 studies an application to sequential search model. The search model has gained popularity as data on consumer online search journeys become increasingly available. The prevailing method to estimate search model is SMLE. However, the estimation is challenging because of the very large number of possible search and purchase combinations. Evaluating the probability of any particular combination requires an infeasibly large number of model simulations. To make SMLE feasible, studies in literature have applied smoothing to the likelihood function.

In Monte Carlo studies, we find that NNE recovers the search model parameter well. Over a wide range of computational costs, NNE has a smaller RMSE than SMLE. The advantage of NNE is especially large at low computational costs. In addition, SMLE's estimates are sensitive to the degree of smoothing. The sensitivity makes it particularly important to choose the appropriate smoothing factor in SMLE. However, the choice of smoothing factor lacks established guidance and has been known to be very difficult for consumer search models in general. When we apply both NNE and SMLE to a real dataset of consumer search, the search model estimated by NNE produces a better fit to key data statistics. The estimates and model fit by SMLE are again sensitive to smoothing.

We evaluate additional properties of NNE in the context of the search model. First, we find that the estimation accuracy of NNE does not deteriorate as $\boldsymbol{m}$ includes more moments, which again suggests robustness to redundant moments. Second, the accuracy of NNE improves with n faster than smoothed SMLE. Third, the estimate by NNE is not sensitive to neural net configuration, and the optimal configuration can be easily selected using validation examples. Fourth, NNE remains reasonably well-behaved even when Θ fails to include the true value of $\boldsymbol{\theta}$; the estimates mostly fall between Θ and the true value.

Overall, our results suggest that NNE is mostly suitable for applications where otherwise heavy simulations are needed to integrate out the unobservables for estimation. In these cases, NNE provides a computationally light alternative to obtain accurate estimates. For applications where the main estimation burden is not simulations, NNE may still be applied but is unlikely to show clear gains. Examples include applications where closed-form likelihood is available and dynamic choices or games where solving the economic model constitutes the main burden in estimation.

We summarize the takeaways of NNE in Table 1. To facilitate applications, we provide our code at: `https://nnehome.github.io`. Finally, we discuss several extensions for future research in Section 5. In particular, further development of NNE may make it practical to rely on aggregate moments to estimate certain structural models that normally require individual-level data. The extension will allow for privacy-preserving structural estimation.

1.1 Literature

This paper adds to a fast-growing literature at the intersection of machine learning and marketing/economics. Machine learning algorithms have been applied to various marketing and economic questions. A particular stream of research focuses on leveraging machine learning to tackle estimation problems. Lewis and Syrgkanis (2018) use the adversarial approach to select from a continuum

Table 1: Summary of NNE

Main properties
1. It does not require computing integrals over unobservables in the structural econometric model; it only requires being able to simulate data using the econometric model.
2. It does not require optimizing over an (potentially non-smooth) objective function as in extremum estimators (e.g., SMLE, SMM, indirect inference).
3. It is more robust to redundant moments when compared to SMM/GMM.
4. It computes a measure of statistical accuracy as a byproduct.

Suitable applications	**Less suitable applications**
A large number of simulations are needed to evaluate likelihood/moments. The SMLE/SMM objective is difficult to optimize. There lacks clear guidance on moment choice. Formulas for standard errors are not yet established.[4]	Closed-form expressions are available for likelihood/moments. The main estimation burden comes from sources other than the simulations to evaluate likelihood/moments.
Examples: sequential search, discrete choices with rich unobserved heterogeneity, matching games, choices on networks.	Examples: dynamic choices or games where the main burden is solving policy functions.

of moment conditions. Wager and Athey (2018) apply random forests to estimate heterogeneous treatment effects. Chernozhukov et al. (2018) study using machine learning models to capture high-dimensional nuisance parameters. Farrell et al. (2021b) use neural nets to model individual heterogeneity. The literature so far has focused on applying machine learning to build more flexible models between economic variables. Our paper explores a different direction, which uses machine learning to estimate the parameter of an econometric model. A recent work sharing this direction is Kaji et al. (2023) that apply the adversarial approach. They propose training a discriminator to construct the objective function of extremum estimator. Their approach is very different from ours (e.g., NNE is not an extremum estimator).

We borrow from the econometric literature on neural nets to derive theoretical properties of NNE. This literature started with White (1989) and White (1990), which establish that single-layer neural nets can learn arbitrary conditional expectations. Later studies mostly examine neural nets under the class of sieve estimators (see Chen 2007). More recently, Farrell et al. (2021a) study multi-layer neural nets. This literature assumes the setting where the goal is to learn relations between data variables. Our paper assumes a different setting where the goal is to learn the relation between data and an econometric model's parameter. We adapt the results in the literature and derive conditions under which they can continue to hold in our setting.

This paper joins a literature that develops computationally light estimators (e.g., Bajari et al. 2007, Pakes et al. 2007, Su and Judd 2012). A particular estimator which NNE shares a similarity

[4] An example is data with network dependence, where asymptotic theory is an active research area. See Lee and Ogburn (2021) and Graham (2020) for surveys. In this case, NNE provides an alternative for inferences.

with is indirect inference (Gourieroux et al. 1993, Collard-Wexler 2013, Bao and Ni 2017). Both approaches avoid the integrals over unobservables in the econometric model. However, the two approaches are conceptually different. As discussed in Gourieroux et al. (1993) and Smith (2008), indirect inference is an extension of SMM, with the usual moments replaced by the parameter of an auxiliary model. As a result, it falls under extremum estimators and inherits issues from SMM. For example, indirect inference has a non-smooth objective function in discrete choice models, which creates difficulty for optimization (Bruins et al. 2018). NNE is not an extremum estimator and faces no such difficulty. In addition, similar to how SMM can be sensitive to moment choice, indirect inference can be sensitive to the choice of auxiliary model. NNE puts less burden on researchers to select moments. We discuss indirect inference more in Appendix A.6.

2 Learning parameter value from data

In this section, we start by describing a framework of structural estimation. Under this framework, we describe how NNE computes the parameter estimate of the structural econometric model and the statistical accuracy of the estimate. We also provide the limit of NNE as L increases.

2.1 Setup

In most applications, we are interested in modeling an outcome $\boldsymbol{y}_i$, with $i = 1, ..., n$ indexing the individuals in data. For example, in a consumer search model, $\boldsymbol{y}_i$ includes both the search and purchase choices of consumer i. We also observe some attributes $\boldsymbol{x}_i$. In the consumer search model, $\boldsymbol{x}_i$ collects the attributes of the products available to consumer i. Let $\boldsymbol{y} \equiv \{\boldsymbol{y}_i\}_{i=1}^n$ and $\boldsymbol{x} \equiv \{\boldsymbol{x}_i\}_{i=1}^n$ collect the data across all individuals. For the purposes of NNE, we consider econometric models that are parametric and allow researchers to simulate the outcome $\boldsymbol{y}$. Formally, denote the econometric model as $\boldsymbol{q}$ and

$$\boldsymbol{y} = \boldsymbol{q}(\boldsymbol{x}, \boldsymbol{\varepsilon}; \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\varepsilon}$ collects the unobserved attributes (sometimes interpreted simply as error terms), and $\boldsymbol{\theta}$ denotes the parameter vector (finite-dimensional in this paper). Throughout this paper, the word "model" when used alone always refers to an econometric model. When referring to a machine learning "model," we will specifically point it out.

The formulation above includes both cases with i.i.d. and non-i.i.d. observations. When individuals are independent observations, y_i does not depend on $(\boldsymbol{x}_j, \boldsymbol{\varepsilon}_j)$ for any $j \neq i$. However, for example, in a model of peer influence over social network, choices are inter-dependent. So y_i may depend on $(\boldsymbol{x}_j, \boldsymbol{\varepsilon}_j)$ for $j \neq i$ and y_i is generally a function of the entire $(\boldsymbol{x}, \boldsymbol{\varepsilon})$.[5]

The maximum likelihood estimator (MLE) estimates $\boldsymbol{\theta}$ by maximizing $\log P(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$. In the i.i.d. case, we have $\log P(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n \log P(\boldsymbol{y}_i|\boldsymbol{x}_i; \boldsymbol{\theta})$. For many econometric models, there is

[5]For example, let $\boldsymbol{A}$ be the adjacency matrix and $y_i = \lambda \sum_{j=1}^n A_{ij} y_j + \boldsymbol{\beta}' \boldsymbol{x}_i + \varepsilon_i$, then $\boldsymbol{y} = (\boldsymbol{I} - \lambda \boldsymbol{A})^{-1}(\boldsymbol{x}\boldsymbol{\beta} + \boldsymbol{\varepsilon})$.

no explicit expression for $P(\boldsymbol{y}_i|\boldsymbol{x}_i;\boldsymbol{\theta})$, and one has to use Monte Carlo simulations to integrate out $\boldsymbol{\varepsilon}_i$. This is known as the simulated maximum likelihood (SMLE). For subsequent discussions, it is useful to define simulation burden. Fix a data size n. Suppose the researcher uses R draws of $\boldsymbol{\varepsilon}_i$ to evaluate $P(\boldsymbol{y}_i|\boldsymbol{x}_i;\boldsymbol{\theta})$. An optimization routine needs to evaluate the likelihood $P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ many times at different values of $\boldsymbol{\theta}$. We define the simulation burden as R times the number of likelihood evaluations by the optimization routine.

An alternative method to estimate $\boldsymbol{\theta}$ is the generalized method of moments (GMM). As with MLE, for many econometric models the evaluation of the moment function requires simulations to integrate out $\boldsymbol{\varepsilon}_i$. This is known as the simulated method of moments (SMM). Suppose the researcher uses R draws of $\boldsymbol{\varepsilon}_i$ to evaluate the moments. Similarly, the simulation burden is R times the number of moment function evaluations by the optimization routine.

2.2 NNE point estimates

Fix an econometric model $\boldsymbol{q}$ as in equation (1). Recall that $\boldsymbol{y} \equiv \{\boldsymbol{y}_i\}_{i=1}^n$ and $\boldsymbol{x} \equiv \{\boldsymbol{x}_i\}_{i=1}^n$, thus $\{\boldsymbol{y}, \boldsymbol{x}\}$ denotes a dataset. In what follows, we describe NNE as using a machine learning algorithm to learn a mapping from datasets to the parameter values of the econometric model $\boldsymbol{q}$:

$$\{\boldsymbol{y}, \boldsymbol{x}\} \mapsto \boldsymbol{\theta}. \tag{2}$$

That is, given any dataset, NNE tries to "recognize" the value of $\boldsymbol{\theta}$ for the econometric model. Importantly, the purpose of the machine learning algorithm is to estimate $\boldsymbol{\theta}$, not to model how $\boldsymbol{x}$ affects $\boldsymbol{y}$. The relation between $\boldsymbol{y}$ and $\boldsymbol{x}$ is entirely specified by the econometric model $\boldsymbol{q}$.

Building the machine learning model requires a training set composed of training examples. Many machine recognition tasks (e.g., image classification) require laborious human labeling to create training sets. This is not required in NNE. Instead, we use the econometric model to generate the training set.

Specifically, let $\ell \in \{1, 2, ..., L\}$ index the training examples. For each ℓ, draw $\boldsymbol{\theta}^{(\ell)}$ uniformly from a parameter space Θ.[6] One may use a non-uniform distribution if she has a specific prior. Let $\boldsymbol{\varepsilon}^{(\ell)}$ be a random draw of the unobservables in the econometric model and $\boldsymbol{y}^{(\ell)} = \boldsymbol{q}(\boldsymbol{x}, \boldsymbol{\varepsilon}^{(\ell)}; \boldsymbol{\theta}^{(\ell)})$. That is, $\{\boldsymbol{y}^{(\ell)}, \boldsymbol{x}\}$ is a dataset generated by the econometric model given $\boldsymbol{\theta}^{(\ell)}$ and the observed $\boldsymbol{x}$. We call it a training dataset. Note that all training datasets are generated conditional on the same observed $\boldsymbol{x}$. Our training set is

$$\{\boldsymbol{\theta}^{(\ell)}, \{\boldsymbol{y}^{(\ell)}, \boldsymbol{x}\}\}_{\ell=1,2...,L}. \tag{3}$$

We train a machine learning model to recognize $\boldsymbol{\theta}^{(\ell)}$ from $\{\boldsymbol{y}^{(\ell)}, \boldsymbol{x}\}$. After training, we plug the real data into the machine learning model to obtain the estimate for $\boldsymbol{\theta}$.

The machine learning model above uses an entire dataset as the input, so the input dimension is in the order of n. One possible variation is to use data moments as the input instead. For many structural econometric models, data moments are sufficient for pinning down parameters, and

[6] We give more discussion on how to choose Θ in practice in Section 4 and Appendix A.2.

economic theory and intuition often suggest what these moments might be. Let $\boldsymbol{m}$ denote a set of moments that summarizes $\{\boldsymbol{y}, \boldsymbol{x}\}$. For example, $\boldsymbol{m}$ may collect the mean of $\boldsymbol{y}_i$: $\bar{\boldsymbol{y}} \equiv n^{-1}\sum_{i=1}^{n}\boldsymbol{y}_i$, the covariance matrix of $\boldsymbol{y}_i$: $n^{-1}\sum_{i=1}^{n}(\boldsymbol{y}_i-\bar{\boldsymbol{y}})(\boldsymbol{y}_i-\bar{\boldsymbol{y}})'$, the cross-covariance matrix between $\boldsymbol{y}_i$ and $\boldsymbol{x}_i$: $n^{-1}\sum_{i=1}^{n}(\boldsymbol{y}_i-\bar{\boldsymbol{y}})(\boldsymbol{x}_i-\bar{\boldsymbol{x}})'$, etc.[7] Similarly, let $\boldsymbol{m}^{(\ell)}$ denote the moments that summarize the simulated dataset $\{\boldsymbol{y}^{(\ell)}, \boldsymbol{x}\}$. Our training set now becomes

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

We train a machine learning model to recognize $\boldsymbol{\theta}^{(\ell)}$ from $\boldsymbol{m}^{(\ell)}$. After training, we plug in the real data moments to obtain the estimate for $\boldsymbol{\theta}$.

The reduction of data to moments has pros and cons. The dimension of $\boldsymbol{m}$ is much smaller than that of $\{\boldsymbol{y}, \boldsymbol{x}\}$. In addition, the mapping from moments to $\boldsymbol{\theta}$ is usually less complex than the mapping from data to $\boldsymbol{\theta}$. Together, these factors reduce the required size and complexity of our machine learning model, which lowers training cost. While large and complex machine learning models (e.g., deep learning models) are common nowadays, they tend to be costly to train.

Using data moments can result in some loss of information in the data. At minimum, we would require that the parameter $\boldsymbol{\theta}$ is identified by the moments selected into $\boldsymbol{m}$ (in this paper we do not specifically address cases where identification is lacking). This requirement is the same as in GMM, where the moment conditions must identify parameter. To mitigate the potential information loss, one would like to include as many potentially relevant moments as possible. However, it is known that in GMM redundant moments can substantially worsen finite-sample estimation biases (e.g., Chen and Liao 2015, Newey 2007). This problem has been studied in settings with lagged moments (Andersen and Sorensen 1996, Altonji and Segal 1996), higher-order moments (Breusch et al. 1999), and interaction moments (Donald and Newey 2021). In such settings, researchers have to carefully balance the information gain against the adverse effect on bias when deciding whether to include additional moments for estimation.

Fortunately, we find NNE to be more robust to redundant moments. We first reveal this property from a theoretical perspective in Section 2.4, and then illustrate it with applications in Section 3 and 4. Intuitively, the robustness comes from the fact that the machine learning algorithm can learn, from the training set, which moments contribute to predicting the parameter and which moments do not. GMM lacks such a mechanism (optimal weighting matrix accounts for the variance-covariance among moments, not how much each moment contributes to parameter estimation).

We now turn to more details on the machine learning model. Use $\boldsymbol{f} : \boldsymbol{m} \mapsto \boldsymbol{\theta}$ to denote a generic mapping from the space of moments to the space of parameter values. We seek to find a mapping $\widehat{\boldsymbol{f}}$ such that

$$\boldsymbol{\theta}^{(\ell)} \simeq \widehat{\boldsymbol{f}}(\boldsymbol{m}^{(\ell)}),\ \forall \ell.$$

One may use any machine learning algorithm here. In this paper, we construct $\widehat{\boldsymbol{f}}$ using shallow

[7] Here, it is useful to note that we need not include moments that involve $\boldsymbol{x}$ only (e.g., covariance matrix of $\boldsymbol{x}_i$), because such moments would be constant across the training examples.

neural nets. The main reasons are: (i) in our context shallow neural nets provide a good balance between learning capacity and ease of training, and (ii) there are relatively well developed statistical properties of shallow neural nets for us to establish convergence results (later in Section 2.4).

Algorithms for training neural nets are well established and we shall not expand on their details. For the discussion here, it is sufficient to view neural nets as a flexible functional form for $\boldsymbol{f}$ parameterized by a set of "weights." The training chooses these weights by minimizing a loss function. Let k index the dimensions of $\boldsymbol{\theta}$. We first consider the mean squared error (MSE) loss function:

$$C_1(\boldsymbol{f}) = L^{-1} \sum_{\ell=1}^{L} \sum_{k} \left[f_k(\boldsymbol{m}^{(\ell)}) - \theta_k^{(\ell)} \right]^2. \tag{5}$$

We choose $\widehat{\boldsymbol{f}}$ as the neural net $\boldsymbol{f}$ that minimizes C_1.[8] We will consider alternative loss functions later. Readers familiar with machine learning may note that loss function (5) does not have the usual penalty term for regularization. Instead, we regularize the neural net by choosing the number of hidden nodes. We do so for the convenience to establish convergence results (Section 2.4), as the econometric literature on neural nets has mostly focused on regularization via the number of hidden nodes (instead of a penalty term). In applications, however, our experience is that NNE works well with either regularization method.

The final element to be discussed is validation. Specifically, we simulate additional datasets $\ell = L+1, ..., L^*$ to form a validation set $\{\boldsymbol{\theta}^{(\ell)}, \boldsymbol{m}^{(\ell)}\}_{\ell=L+1}^{L^*}$. Note the index here starts with $L+1$. The validation set is not used in the loss function (5). Instead, the validation set can be used to check the accuracy of $\widehat{\boldsymbol{f}}$ as well as choose the number of hidden nodes.

2.3 NNE statistical accuracy

The development so far has focused on training the neural net to give a point estimate of $\boldsymbol{\theta}$. Next, we train the neural net to provide the statistical accuracy of point estimates. Importantly, we will use the same training set as in expression (4). At first glance, this task may appear implausible because in each training example, there is only one parameter value $\boldsymbol{\theta}^{(\ell)}$ associated with one moment value $\boldsymbol{m}^{(\ell)}$. However, across the training examples, we can measure the dispersion of parameter values around any given moment value. Conceptually, this dispersion provides us with an estimate of statistical accuracy.

More formally, later in Section 2.4 we establish that $\widehat{\boldsymbol{f}}$ trained using the loss in equation (5) converges to $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ as $L \to \infty$. Continuing with this result, here we seek to estimate the variance vector $\mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})$ or the covariance matrix $\mathbb{C}\text{ov}(\boldsymbol{\theta}|\boldsymbol{m})$ as a measure of statistical accuracy. We achieve this by modifying the NNE with two changes: (i) the neural net outputs not only a point estimate but also the variance and covariance terms, and (ii) the loss function is specified accordingly to properly train the neural net.

Specifically, we configure the neural net $\boldsymbol{f}$ such that its output has two parts. The first part is

[8] We abstract away from optimization issues in neural net training and assume that the minimum can be attained. Neural network optimization is an active research area (see e.g., Dauphin et al. 2014, Li et al. 2018, Du et al. 2019).

a vector $\boldsymbol{\mu}$ for the point estimate and the second part specifies a covariance matrix $\boldsymbol{V}$. We write $\boldsymbol{f} = (\boldsymbol{\mu}, \boldsymbol{V})$. In practice, we need to ensure that $\boldsymbol{V}$ is positive definite. One way to do so is Cholesky decomposition. We let $\boldsymbol{f}$ compute the Cholesky factor of $\boldsymbol{V}$ and then transform it into $\boldsymbol{V}$. A special case arises in applications when researchers are interested in only the variance terms. As we will show below, in this case we may simplify $\boldsymbol{V}$ to be a diagonal matrix with the variance terms only. To ensure positive variances, we can let $\boldsymbol{f}$ compute the log of the diagonal elements and then map them into $\boldsymbol{V}$.

Using the above notation, we write $(\boldsymbol{\mu}^{(\ell)}, \boldsymbol{V}^{(\ell)}) = \boldsymbol{f}(\boldsymbol{m}^{(\ell)})$ for each ℓ in the training set. We use the following loss function to train the neural net. This loss function is a generalization of C_1 and known as cross-entropy loss in the machine learning literature.

$$C_2(\boldsymbol{f}) = L^{-1} \sum_{\ell=1}^{L} \left[\log\left(|\boldsymbol{V}^{(\ell)}| \right) + \left(\boldsymbol{\theta}^{(\ell)} - \boldsymbol{\mu}^{(\ell)} \right)' \left(\boldsymbol{V}^{(\ell)} \right)^{-1} \left(\boldsymbol{\theta}^{(\ell)} - \boldsymbol{\mu}^{(\ell)} \right) \right], \\ \text{where } (\boldsymbol{\mu}^{(\ell)}, \boldsymbol{V}^{(\ell)}) = \boldsymbol{f}(\boldsymbol{m}^{(\ell)}). \tag{6}$$

We train a neural net $\widehat{\boldsymbol{f}}$ that minimizes this loss function C_2 instead of C_1.

One may note that C_2 uses the normal p.d.f. However, this does *not* mean that we must assume normality for the underlying distribution $P(\boldsymbol{\theta}|\boldsymbol{m})$. Formally, we will show in Section 2.4 that $\widehat{\boldsymbol{f}}$ converges to $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ and $\mathbb{C}\text{ov}(\boldsymbol{\theta}|\boldsymbol{m})$ as $L \to \infty$. If $\boldsymbol{V}$ is specified as diagonal, then $\widehat{\boldsymbol{f}}$ converges to $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ and $\mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})$ as $L \to \infty$. Importantly, these results hold for general $P(\boldsymbol{\theta}|\boldsymbol{m})$ that may or may not be normal.[9]

To summarize, we use the following steps to obtain both the point estimate and a measure of statistical accuracy. First, we follow Section 2.2 to generate the pairs $\{\boldsymbol{\theta}^{(\ell)}, \boldsymbol{m}^{(\ell)}\}_{\ell=1}^{L}$. Then, we use these pairs to train a neural net $\widehat{\boldsymbol{f}} : \boldsymbol{m} \mapsto (\boldsymbol{\mu}, \boldsymbol{V})$ under the loss function C_2. Finally, we plug the moments of the real data into $\widehat{\boldsymbol{f}}$ to obtain the point estimate and the statistical accuracy for the point estimate.

2.4 Convergence results

We provide three convergence results for NNE as well as the intuitions for these results. The first result concerns with the neural net trained under loss function C_1 (see equation 5). The second result concerns with loss function C_2 (see equation 6). The third result concerns with a more general loss function C_3, which we will define. All the convergence results rely on $L \to \infty$ and hold for any finite data size n. Readers more interested in applications may skip this subsection without much loss of continuity. All proofs are given in Appendix B.

Our results build on the existing econometric literature on the asymptotic properties of neural nets (e.g., Chen 2007 and White 1990). The literature studies the setting where neural nets learn about the relation from $\boldsymbol{x}_i$ to $\boldsymbol{y}_i$, such as $\mathbb{E}(\boldsymbol{y}_i|\boldsymbol{x}_i)$. That is, neural nets are used as econometric

[9] Nevertheless, it is useful to note a general result in Bayesian statistics that posteriors tend to normal distributions with $n \to \infty$ (see Gelman et al. 2004, Kim 2002). Thus, in applications with a large number of observations, one may assume $P(\boldsymbol{\theta}|\boldsymbol{m})$ to be normal.

models. This is different from our setting where we use neural nets to recover the parameter $\boldsymbol{\theta}$ of a econometric model. In other words, the relation to be learned is from $\{\boldsymbol{x}, \boldsymbol{y}\}$ to $\boldsymbol{\theta}$. At a high level, our propositions "translate" the asymptotic results in the literature from their original setting to our setting, deriving conditions specific to our setting so that the asymptotics can be upheld. It is useful to note that in principle, our results are not restricted to neural nets. Generically, we expect a machine learning model to converge to the target relation as the training set grows, if its capacity properly adjusts to the training set size and it is flexible enough to include the target relation (as its capacity grows). Our results should still apply if the neural net is replaced by a machine learning model for which such asymptotic property is true.

We start by formalizing several notions to be used in the convergence results. Conditional on the observed $\boldsymbol{x}$, the econometric model together with the prior over Θ (e.g., uniform) implies a distribution of $\boldsymbol{\theta}$ and $\boldsymbol{y}$. This distribution in turn implies a distribution of $\boldsymbol{\theta}$ and $\boldsymbol{m}$, which we denote as $P(\boldsymbol{\theta}, \boldsymbol{m})$. Then, $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ is the best estimate of parameter given the set of moments. In Bayesian language, this is known as the limited-information posterior mean, with "limited-information" signifying that the posterior is conditional on a set of moments instead of the entire data (Kim 2002).[10] To express asymptotic results, we introduce a sequence of function spaces $\{\mathcal{F}_L\}_{L=1}^{\infty}$, with each $\mathcal{F}_L$ denoting a class of neural nets. Because the exact specification of $\mathcal{F}_L$ concerns only technical aspects of proofs, we describe it in the appendix. For our exposition below, the key property of $\{\mathcal{F}_L\}_{L=1}^{\infty}$ is that the capacity of the neural nets in $\mathcal{F}_L$ grows gradually with L. Intuitively, this property formalizes regularization – the neural net's capacity should be appropriate for the size of training set. Let $\widehat{\boldsymbol{f}}_L$ denote a neural net in $\mathcal{F}_L$ that minimizes C_1. Let $\|\cdot\|$ denote the 2-norm of 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) Θ 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 (5), we have $\|\widehat{\boldsymbol{f}}_L - \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})\| \to 0$ in probability as $L \to \infty$.* □

In words, the proposition says that as we increase the number of training datasets, NNE converges to $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ as a function of $\boldsymbol{m}$. The conditions are relatively mild. The compactness of Θ 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. 2021a for some discussion). The continuity condition requires $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ not to change abruptly when the observed data moments change only slightly, which is satisfied in virtually any application.

We note that Proposition 1 lets $L \to \infty$ while holding the data size n constant. Also note that given an application where n is fixed, the accuracy of $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ as an estimate of $\boldsymbol{\theta}$ will not decrease when more moments are added to $\boldsymbol{m}$. Instead, the accuracy will weakly increase. Thus, with a sufficiently large L, the NNE will become weakly more accurate as we add moments to $\boldsymbol{m}$. This is in contrast to SMM, where it is known that even with the number of simulations $R \to \infty$,

[10] There is a relation between our method and approximate Bayesian computation (ABC). Conceptually, NNE can be seen as an implementation of ABC that takes advantages of neural nets.

redundant moments generally lead to larger biases. We have discussed this difference in Section 2.2 from a more applied perspective.

Next, we move to loss function C_2. For the following proposition, the neural net outputs both the point estimate and the measure of statistical accuracy as described in Section 2.3. We use $\widehat{\boldsymbol{f}}_L$ to denote a neural net that minimizes C_2 in $\mathcal{F}_L$.

Proposition 2. *Suppose condition (i) - (iii) in Proposition 1 hold, and in addition: (iv)* $\mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})$ *is continuous in* $\boldsymbol{m}$ *and* $\mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m}) \geq \delta$ *for some* $\delta > 0$*. Under loss (6) with diagonal covariance matrix, we have* $\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| \to 0$ *in probability with* $\boldsymbol{f}^* = [\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m}), \mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})]$.

In addition to condition (i)-(iv) above, suppose: (v) $\mathbb{C}\text{ov}(\boldsymbol{\theta}|\boldsymbol{m})$ *is continuous in* $\boldsymbol{m}$ *and its smallest eigenvalue is bounded below by some positive number. Under loss (6) with full covariance matrix, we have* $\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| \to 0$ *in probability with* $\boldsymbol{f}^* = [\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m}), \mathbb{C}\text{ov}(\boldsymbol{\theta}|\boldsymbol{m})]$. □

In words, the proposition says that as we increase the number of training datasets, NNE converges to $\mathbb{V}\text{ar}(\boldsymbol{\theta}|\boldsymbol{m})$ or $\mathbb{C}\text{ov}(\boldsymbol{\theta}|\boldsymbol{m})$ as a function of $\boldsymbol{m}$, in addition to $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$. Compared to Proposition 1, the additional conditions essentially require the underlying distribution $P(\boldsymbol{\theta}|\boldsymbol{m})$ to be bounded away from a degenerate one. Importantly, note that the proposition does not make any distributional assumption on $P(\boldsymbol{\theta}|\boldsymbol{m})$. In particular, it does not require $P(\boldsymbol{\theta}|\boldsymbol{m})$ to be normal. This point has been previously noted in Section 2.3.

Below, we turn to a more general loss function. The goal is to generalize and provide some high-level intuition behind Proposition 1 and 2. To introduce this loss function, 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 Γ. For example, in case of the normal family, $\boldsymbol{\gamma}$ collects both the mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{V}$ and $\phi(\boldsymbol{\theta};\boldsymbol{\gamma}) = |2\pi\boldsymbol{V}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu})'\boldsymbol{V}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu})}$. Let $\boldsymbol{f}$ be a neural net that outputs the elements of $\boldsymbol{\gamma}$. Now we may define the following loss function.

$$C_3(\boldsymbol{f}) = L^{-1} \sum_{\ell=1}^{L} -\log \phi\left[\boldsymbol{\theta}^{(\ell)};\ \boldsymbol{f}(\boldsymbol{m}^{(\ell)})\right]. \tag{7}$$

Note that C_3 reduces to C_2 if ϕ is the normal family. It further reduces to C_1 if we impose an identity covariance matrix.

To develop intuition, we consider the minimization of the limiting version of C_3. As $L \to \infty$, we would expect C_3 to approach C_3^* defined as follows.

$$C_3^*(\boldsymbol{f}) = -\mathbb{E}\left[\log \phi\left(\boldsymbol{\theta};\ \boldsymbol{f}(\boldsymbol{m})\right)\right].$$

The expectation on the right hand side is taken over $P(\boldsymbol{\theta}, \boldsymbol{m})$. As we know from the theory of MLE (e.g., White 1982), minimization of C_3^* amounts to the minimization of Kullback–Leibler (KL) divergence. To be more precise, suppose that function $\boldsymbol{f}^*$ minimizes C_3^*. Then, for each possible value of $\boldsymbol{m}$, $\boldsymbol{f}^*(\boldsymbol{m})$ should be equal to a value of $\boldsymbol{\gamma}$ that makes the density $\phi(\boldsymbol{\theta};\boldsymbol{\gamma})$ closest to $P(\boldsymbol{\theta}|\boldsymbol{m})$ in terms of the KL divergence. As it turns out, if ϕ is specified as the normal family,

the KL divergence is minimized when ϕ takes the same mean and covariance matrix as $P(\boldsymbol{\theta}|\boldsymbol{m})$. This result explains Proposition 2. If ϕ is specified as the normal family with identity covariance matrix, then the KL divergence is minimized when ϕ takes the same mean as $P(\boldsymbol{\theta}|\boldsymbol{m})$. This result explains Proposition 1.

For subsequent applications in this paper, we will not be using loss function C_3 – loss functions C_1 and C_2 will be sufficient. Nevertheless, for completeness we formalize the above intuition for C_3 in a proposition below. For this proposition, $\widehat{\boldsymbol{f}}_L$ denotes a neural net that minimizes C_3 in $\mathcal{F}_L$. We use $\mathcal{KL}(\cdot\|\cdot)$ to denote the KL divergence.

Proposition 3. *Suppose: (i) Θ is compact and Γ is compact with a non-empty interior, (ii) $\boldsymbol{m}$ has a compact support, (iii) ϕ 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 Γ. Let $\boldsymbol{f}^*(\boldsymbol{m})$ denote the minimizing point in (v). Then $\|\widehat{\boldsymbol{f}}_L - \boldsymbol{f}^*\| \to 0$ in probability.* □

In words, as the number of training datasets increases, the NNE will converge to a function that brings the parameterized family ϕ as close to $P(\boldsymbol{\theta}|\boldsymbol{m})$ as possible. The key assumptions are sufficient levels of continuity and a uniqueness condition. The uniqueness condition (i.e., assumption (v)) puts a restriction on ϕ, requiring that there are not multiple members of ϕ being equally closest to $P(\boldsymbol{\theta}|\boldsymbol{m})$. We note that this uniqueness condition is not imposed on the neural net. A neural net is usually over-parameterized by a large number of weights, and there are always multiple combinations of weights that equally minimize the loss function. We only require $\widehat{\boldsymbol{f}}_L$ to take one of these weight combinations that minimize the loss function.[11]

3 Illustration by a Simple Example

In this section, we illustrate NNE through a simple AR(1) model. The simplicity of the model permits us to graphically present NNE and compare it to SMM. We note that for this simple econometric model, closed-form expressions of likelihood and moments are available. Thus, simulation-based estimators like NNE do not really offer computational gains. However, our goal here is not to compare computational performances, but to illustrate how NNE works and develop intuitions that extend to more complex structural econometric models. Readers more interested in NNE's performance in a structural application may skip this section and go to Section 4.

3.1 Model setup and estimation

We consider a simple AR(1) model. Let $\varepsilon_i \sim \mathcal{N}(0,1)$ and $\beta \in [0,1)$,

$$y_i = \beta y_{i-1} + \varepsilon_i. \tag{8}$$

The model has a single parameter β. As we will see, this simple setting makes it possible for us to graphically illustrate estimation procedures. We assume that the initial value y_1 is drawn

[11] We thank a referee for pointing this out.

Algorithm 1 NNE for AR(1)

Suppose we have a dataset $\{y_i\}_{i=1}^n$, where y_i follows the AR(1) process. NNE uses the steps below to obtain an estimate for β.

1. Given an index ℓ, draw $\beta^{(\ell)}$ uniformly from an interval such as $[0, 0.9]$. Draw $y_1^{(\ell)} \sim \mathcal{N}(0, 1/(1-\beta^2))$. Draw $\varepsilon_i^{(\ell)} \sim \mathcal{N}(0,1)$ and compute $y_i^{(\ell)} = \beta^{(\ell)} \cdot y_{i-1}^{(\ell)} + \varepsilon_i^{(\ell)}$ for $i = 2, ..., n$.
2. Compute the moment $m^{(\ell)} = \frac{1}{n-1} \sum_{i=2}^n y_i^{(\ell)} y_{i-1}^{(\ell)}$.
3. Repeat the above for all $\ell \in \{1, ..., L^*\}$ to obtain the set $\{\beta^{(\ell)}, m^{(\ell)}\}_{\ell=1}^{L^*}$. Split the set 90/10 for training and validation. Train a neural net to predict $\beta^{(\ell)}$ from $m^{(\ell)}$.
4. Apply the neural net to the moment of the original dataset, $m = \frac{1}{n-1} \sum_{i=2}^n y_i y_{i-1}$, to obtain the estimate $\widehat{\beta}$.

from the stationary distribution, $\mathcal{N}(0, 1/(1-\beta^2))$. We set the sample size $n = 100$, so that a dataset is $\{y_i\}_{i=1}^{100}$. Below we describe how NNE and SMM estimate β. Both NNE and SMM are moment-based methods and we compare the two methods to highlight their similarities and differences.

For NNE, we list the steps in Algorithm 1. In particular, note here we use a single data moment, $m = \frac{1}{n-1} \sum_{i=2}^n y_i y_{i-1}$, which is the covariance between y_i and its own lag. This single-moment case allows straightforward graphical presentation, which we will show in Section 3.2. We will examine using multiple moments in Section 3.3.

For SMM, we start with GMM using the same moment used by NNE above. By equation (8), we have

$$\mathbb{E}(y_i y_{i-1}) = \underbrace{\beta/(1-\beta^2)}_{g(\beta)}. \tag{9}$$

We denote the right hand side as function $g(\beta)$. Because $g(\beta)$ is strictly increasing over $\beta \in [0, 1)$, one could solve for the precise value of β if she knew the value of $\mathbb{E}(y_i y_{i-1})$. However, in estimation, we do not know $\mathbb{E}(y_i y_{i-1})$ but only a sample realization of it. This sample realization is $m = \frac{1}{n-1} \sum_{i=2}^n y_i y_{i-1}$. GMM estimates β by solving:

$$g(\beta) - m = 0.$$

We know the closed-form expression for $g(\beta)$ in this simple AR(1) model. However, in general structural estimation, closed-form expressions of moments are often unavailable. To mirror this more general scenario, suppose for a moment that we do not know the closed form of $g(\beta)$ and have to evaluate it via simulations. For any given β we use the AR(1) model to simulate R copies of the data: $\{y_i^{(r)}\}_{i=1}^n$, $r = 1, ..., R$. Then, we approximate $g(\beta)$ by $\widehat{g}(\beta) = \frac{1}{R} \sum_{r=1}^R (\frac{1}{n-1} \sum_{i=2}^n y_i^{(r)} y_{i-1}^{(r)})$. A larger R makes the approximation more accurate but also more costly. SMM estimates β by

solving

$$\widehat{g}(\beta) - m = 0.$$

Next, we illustrate NNE and SMM graphically to compare the two approaches.

3.2 Graphic illustration of estimation

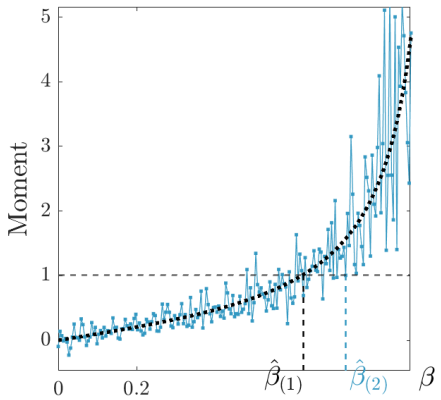
Figure 2 uses six plots to graphically illustrate SMM and NNE in the estimation of AR(1). Below we discuss the plots in order. The goal is to develop a better understanding of how NNE approaches parameter estimation.

Plot (a) and (b) illustrate SMM. The dotted black curve in plot (a) shows $g(\beta)$ as defined in equation (9). As a hypothetical example, the horizontal dashed line shows the value of $m = \frac{1}{n-1}\sum_{i=2}^{n} y_i y_{i-1}$ in the real data. Under this example, GMM's estimate for β is given by the intersection between this horizontal dashed line and $g(\beta)$, indicated as $\widehat{\beta}_{(1)}$ on the axis. The jagged blue curve shows $\widehat{g}(\beta)$ evaluated with $R = 1$. The SMM's estimate for β is given by the point on the jagged blue curve that is closest to the horizontal dashed line, marked as $\widehat{\beta}_{(2)}$ on the axis. The gap between $\widehat{\beta}_{(1)}$ and $\widehat{\beta}_{(2)}$ is the estimation error due to simulations. This error can be reduced by a larger R at the expense of a higher computational cost. As an example, plot (b) shows $\widehat{g}(\beta)$ evaluated with $R = 5$, which we see approximates $g(\beta)$ more closely.

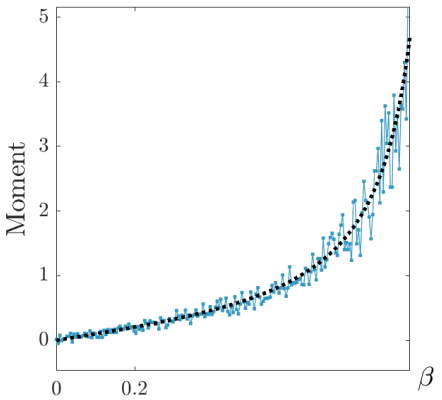
Plot (c) and (d) illustrate a simplified version of NNE that replaces neural net with a linear function. The simplification helps to demonstrate the core idea of NNE. We use $L = 25$ training points: $\{\beta^{(\ell)}, m^{(\ell)}\}_{\ell=1}^{25}$. In plot (c), these training points are shown as red circles and are randomly sampled from the same jagged blue curve $\widehat{g}(\beta)$ as in plot (a). The red line is a linear fit of $\beta^{(\ell)}$ onto $m^{(\ell)}$. Plot (d) reproduces the horizontal dashed line that represents a hypothetical value of m in real data. The NNE's estimate for β is at the intersection between the red line and horizontal dashed line, marked as $\widehat{\beta}_{(3)}$ on the axis. The difference between $\widehat{\beta}_{(1)}$ and $\widehat{\beta}_{(3)}$ is a *learning* error, caused by us using a parameterized function (a linear fit in this case) to learn $g(\beta)$.

Plot (e) and (f) illustrate NNE. We use the same $L = 25$ training points as above. Plot (e) fits the points by a neural net with 4 hidden neurons and ReLu activation function. We see a much better fit than the linear function. The learning error is not marked out in the plot, but one can easily see that it will be much smaller than in plot (d). Aside from ReLu, another popular activation function is the sigmoid. Plot (f) uses a neural net with 4 hidden neurons and sigmoid activation function. The fit is on par with that in plot (e).

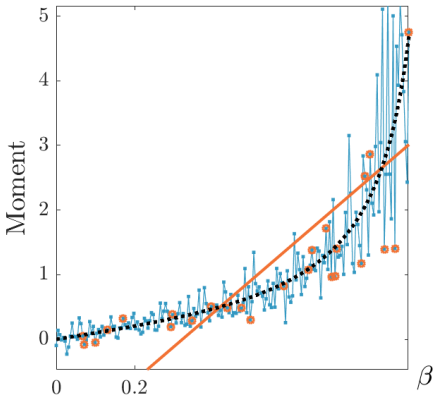
Overall, plots (c) - (f) demonstrate the key idea of NNE that it attempts a *functional* approximation of the relation between moment and parameter. Under this idea, an important detail which we have not emphasized above is that NNE approximates the inverse function g^{-1} instead of g. That is, the neural net predicts $\beta^{(\ell)}$ from $m^{(\ell)}$ (not $m^{(\ell)}$ from $\beta^{(\ell)}$). The purpose is to facilitate estimation. Specifically, suppose $\widehat{f}$ approximates g^{-1}, then we may estimate β directly as $\widehat{f}(m)$. However, suppose $\widehat{f}$ approximated g instead, then we would have to solve $\widehat{f}(\beta) = m$ to obtain an estimate of β. This problem will complicate further when we use multiple moments because



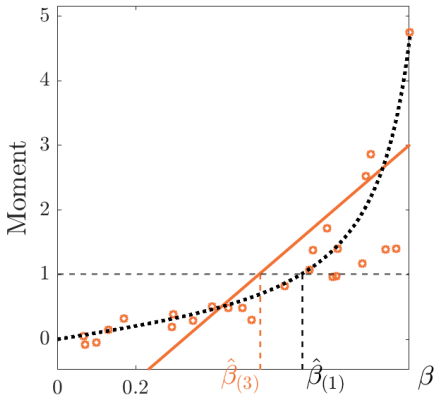
(a) Black dotted curve shows $\mathbb{E}(y_t y_{t-1})$ as a function of β. Jagged blue curve shows the approximation by SMM with $R = 1$.



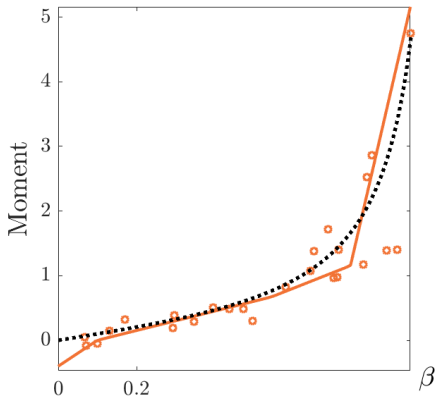
(b) Jagged blue curve shows the approximation by SMM with $R = 5$.



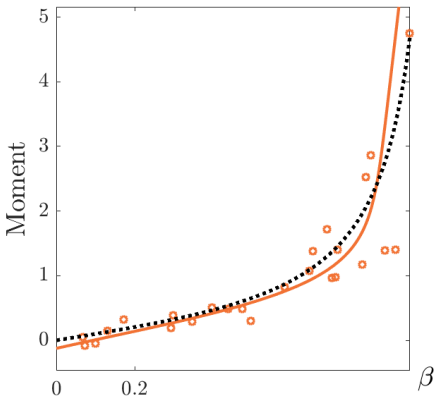
(c) A linear fit with $L = 25$ training points sampled from jagged blue curve. These training points are shown as red circles.



(d) Estimation of β based on the linear fit.



(e) A fit by neural net with ReLu activation, using $L = 25$ training points.



(f) A fit by neural net with sigmoid activation, using $L = 25$ training points.

Figure 2: Estimation of AR(1)

Table 2: Estimation of AR(1) under Different Moments

Moments	SMM ($R=\infty$)		NNE ($L=1e3$)	
	Bias	RMSE	Bias	RMSE
1) y_iy_{i-1}	-0.019 (.003)	0.093 (.002)	-0.017 (.003)	0.091 (.002)
2) y_iy_{i-1}, y_i^2	-0.015 (.003)	0.088 (.002)	-0.013 (.003)	0.086 (.002)
3) $y_iy_{i-k}, k=1,2,3$	-0.026 (.003)	0.099 (.003)	-0.015 (.003)	0.091 (.002)
4) $y_iy_{i-k}, k=1,2,...,10$	-0.043 (.003)	0.111 (.003)	-0.014 (.003)	0.095 (.002)
5) y_iy_{i-1}, $y_i^2y_{i-1}$, $y_iy_{i-1}^2$	-0.075 (.003)	0.130 (.003)	-0.014 (.003)	0.095 (.002)
6) y_iy_{i-k}, $y_i^2y_{i-k}$, $y_iy_{i-k}^2$, $k=1,2,3$	-0.086 (.003)	0.135 (.003)	-0.013 (.003)	0.096 (.002)

Notes: SMM ($R=\infty$) is the same as GMM and follows the usual 2-step procedure. The 1st step constructs optimal weighting matrix accounting for serial correlations. NNE ($L=1e3$) uses a shallow neural net of 32 hidden nodes. Reported numbers are based on 1000 Monte Carlo datasets. Numbers in parentheses are standard errors.

$\widehat{\boldsymbol{f}}(\beta)=\boldsymbol{m}$ typically has no exact solution.

To summarize, Figure 2 shows that SMM and NNE take different approaches and subsequently incur different types of estimation errors. While both rely on approximations of the mapping between moment and parameter, SMM makes a *point-by-point approximation from parameter to moment* whereas NNE makes a *functional approximation from moment to parameter*. It is not difficult to see that in cases where the moment evaluation has sizable simulation errors (so that the point-by-point approximation is noisy), the functional approach by NNE can have advantages. A similar comparison applies to NNE and indirect inference (see Appendix A.6).

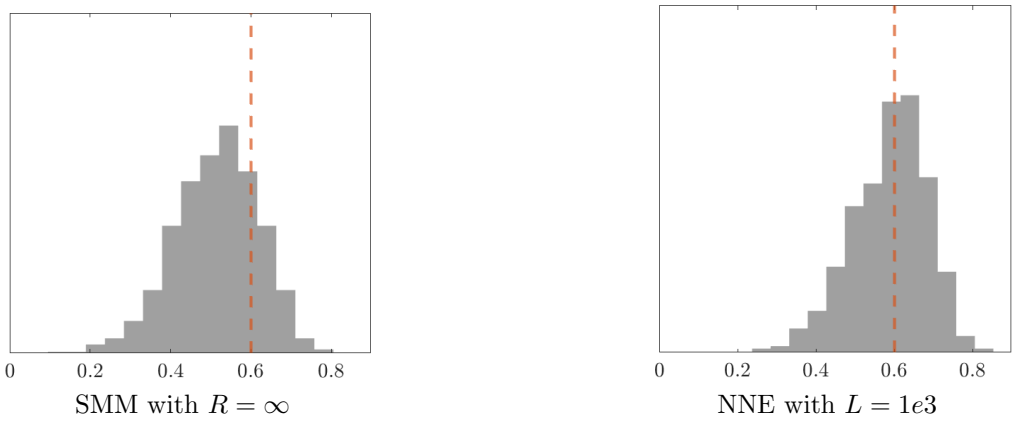
3.3 Robustness to redundant moments

So far, we have focused on the estimation relying a single moment for the purpose of visual illustration. Next, we move to estimation using multiple moments to examine NNE's robustness to redundant moments in the context of the AR(1) model. As discussed in Section 2, it is known that redundant moments can introduce substantial biases to SMM in finite samples. In comparison, NNE should be less affected by redundant moments thanks to its learning mechanism.

We use Monte Carlo experiments to compare SMM and NNE with multiple moments. The simplicity of AR(1) allows us to choose $R=\infty$ for SMM, which is just GMM. This choice removes simulation errors and thus helps to isolate the impact of redundant moments. For NNE, we use $L=1000$. Our Monte Carlo exercise sets $\beta=0.6$ and generates 1000 datasets by the AR(1). We estimate β from each of these Monte Carlo datasets.

Table 2 shows the results under six different moment specifications. Row 1 is our benchmark relying on the single moment $\frac{1}{n-1}\sum_{i=2}^{n} y_iy_{i-1}$. Row 2 adds the moment $\frac{1}{n}\sum_{i=1}^{n} y_i^2$. Note that $\mathbb{E}(y_i^2)=1/(1-\beta^2)$ so the moment is informative about β. We see a slight reduction in the RMSE for GMM, suggesting that in this case the information gain from the added moment overcomes the potential bias increase. We also see a slight reduction in the RMSE for NNE, showing that NNE is able to capitalize on the information from the added moment too.

Row 3 and 4 add moments involving further lags to the benchmark case. Note that $\mathbb{E}(y_iy_{i-k})=$



Notes: Histograms of the estimates of β across 1000 Monte Carlo datasets. The vertical dashed red lines display the true value of β. The moment specification follows row 6 in Table 2.

Figure 3: Distributions of Estimates for AR(1) with Redundant Moments

$\beta^k/(1-\beta^2)$, so each of these added moments by itself is informative about β. However, moments with more lags (i.e., larger k) may become almost redundant after we include moments with fewer lags (i.e., smaller k). In both of these rows, GMM has larger bias and RMSE than the benchmark case (row 1). In comparison, NNE's performance remains close to the benchmark case.

Row 5 and 6 add higher-order moments to the benchmark case. While these moments may appear informative at glance, a bit of algebra will show that $\mathbb{E}(y_i^2 y_{i-k}) = 0$ and $\mathbb{E}(y_i y_{i-k}^2) = 0$ regardless of β. Therefore, these moments are irrelevant and redundant. In both rows, GMM now has much larger bias and RMSE than the benchmark case. In comparison, NNE's performance remains close to the benchmark case. For a visual comparison, Figure 3 plots the histograms of the estimates using the nine moments in row 6. The bias of GMM is clear.

To summarize, Table 2 shows that NNE is less sensitive to redundant moments, consistent with our discussion in Section 2. Note, in particular, that the simplicity of AR(1) allows us to derive closed-form expressions of moments and see whether a moment may be redundant. However, doing so more generally in structural estimation is not feasible. In applications of GMM/SMM, researchers often need to carefully select moments rather than just "throwing in" moments that might or might not be useful for recovering parameters. In this regard, the robustness of NNE lessens the burden on researchers to select moments.

4 Application to Consumer Sequential Search

This section studies an application to the sequential search model (Weitzman, 1979, Ursu 2018). The model has attracted growing interests as data on consumer online purchase journey become increasingly available (i.e., what alternative options consumers viewed before purchases). The model is typically estimated by SMLE and the estimation is known to be difficult. The main

challenge comes from the large number of possible combinations of search and purchase choices. The challenge makes likelihood evaluation difficult and allows potential gains from using NNE.

Below, we start by briefly describing the search model and how to estimate it with NNE and SMLE. Then, we present Monte Carlo studies of both estimation approaches. We focus on estimation accuracy as well as computational costs. Next, we estimate the search model using real data. With real data, we can no longer calculate estimation accuracy (because the true value of search model parameter is unknown). So instead, we focus on model fit. Finally, we examine how NNE's performance varies with the choice of moments as well as data size.

4.1 Search model

We describe the search model in the context of hotel bookings online, largely following Ursu (2018). A consumer is shown a list of J hotel options. She can make a search by clicking into a hotel option. After searches, she may make a purchase by booking one of the hotels. Let consumer i's utility from booking hotel j be

$$u_{ij} = \underbrace{\boldsymbol{z}_{ij}'\boldsymbol{\beta}}_{v_{ij}} + \varepsilon_{ij},$$

where vector $\boldsymbol{z}_{ij}$ collects hotel attributes, including star rating, review score, location score, log price, whether it belongs to a chain, and whether a promotion is offered. We use the log price because price has a skewed distribution. Consumer i observes $v_{ij} \equiv \boldsymbol{z}_{ij}'\boldsymbol{\beta}$ before searching. However, she observes ε_{ij} only after searching hotel j. An outside option where she makes no purchase offers an utility of $u_{i0} = \eta + \varepsilon_{i0}$. This outside utility is known to her without search. It is assumed that $\varepsilon_{ij} \sim \mathcal{N}(0,1)$. The distribution of ε_{ij} is assumed known to consumers before search.

Consumers incur costs for searching. The search cost may vary by the ranking position of a hotel in the hotel list. Let $s_{ij} \in \{1, 2, ..., J\}$ be the ranking of hotel j. We specify the cost of searching hotel j as

$$c_{ij} = e^{\delta_0 + \delta_1 \log(s_{ij})}.$$

The exponential function ensures that c_{ij} is always positive. Parameter δ_1 captures the effect of ranking. We take the log of s_{ij} because doing so improves model fit. One convention in the literature is to allow one "free search," which we follow. That is, consumers do not incur search cost for her first searched hotel. The purpose is to accommodate data where observations are recorded only if consumer has made at least one search.

The optimal search and purchase strategy is characterized in Weitzman (1979). Briefly speaking, the strategy associates each product option j with a "reservation utility," which is a function of v_{ij} and c_{ij}. The consumer orders her searches by reservation utility, starting with the option with the highest reservation utility. The consumer stops searching when the highest utility from the searched options is higher than the maximum of the reservation utilities in the remaining options.

In data, we observe each consumer's purchase and searches. Let $\boldsymbol{y}_{ij}$ collect two dummy variables $y_{ij}^{(\text{buy})}$ and $y_{ij}^{(\text{search})}$, where $y_{ij}^{(\text{buy})} = 1$ if option j is purchased and $y_{ij}^{(\text{search})} = 1$ if option j is searched.

Algorithm 2 NNE for consumer search

Suppose we have a dataset $\{\{\boldsymbol{y}_{ij}, \boldsymbol{x}_{ij}\}_{j=1.}^{J}\}_{i=1}^{n}$, where $\boldsymbol{y}_{ij}$ collects search and purchase choices and $\boldsymbol{x}_{ij}$ collects hotel attributes and rankings. We use the following steps to estimate the search model.

1. Given an index ℓ, draw $\boldsymbol{\theta}^{(\ell)}$ uniformly from the parameter space Θ. Given this parameter draw and the observed $\boldsymbol{x}_{ij}$ for all i and j, simulate the search and purchase outcomes $\boldsymbol{y}_{ij}^{(\ell)}$ for all i and j.
2. Compute the moments $\boldsymbol{m}^{(\ell)}$ that summarize $\{\{\boldsymbol{y}_{ij}^{(\ell)}, \boldsymbol{x}_{ij}\}_{j=1.}^{J}\}_{i=1}^{n}$.
3. Repeat the above for each $\ell \in \{1, ..., L^*\}$ to obtain the set $\{\boldsymbol{\theta}^{(\ell)}, \boldsymbol{m}^{(\ell)}\}_{\ell=1}^{L^*}$. Split the set 90/10 for training and validation. Train a neural net to predict $\boldsymbol{\theta}^{(\ell)}$ from $\boldsymbol{m}^{(\ell)}$.
4. Apply the neural net to the moments of the original dataset to obtain $\widehat{\boldsymbol{\theta}}$.

Let $\boldsymbol{x}_{ij} = (\boldsymbol{z}_{ij}, \log s_{ij})'$. So the data can be denoted as $\{\{\boldsymbol{y}_{ij}, \boldsymbol{x}_{ij}\}_{j=1.}^{J}\}_{i=1}^{n}$. The parameter vector $\boldsymbol{\theta}$ includes $\boldsymbol{\beta}$, η, δ_0, and δ_1. A few key statistics will be useful in our exposition: buy rate, number of searches per consumer, and search ranking. The buy rate is the fraction of consumers who made purchases and the search ranking is the average ranking of searched options.

4.2 Estimation of search model

We describe how NNE and SMLE estimate the consumer search model.

NNE for search model

Algorithm 2 lists the main steps of NNE applied to the consumer search model. Below, we give further details of the algorithm (on the parameter space Θ, moment choice $\boldsymbol{m}$, and neural net training) for readers who would like to replicate our results. Readers who are more interested in the results may skip these details without loss of continuity.

We specify the parameter space Θ as follows: $\eta \in [2, 5]$, $\delta_0 \in [-5, -2]$, $\delta_1 \in [-0.25, 0.25]$, and $\beta_k \in [-0.5, 0.5]$ for all k. This Θ covers wide ranges of key data statistics. In a typical training set, the purchase rate ranges from 0 to 1, number of searches per consumer ranges from 1 to 23.3, and search ranking ranges from 1.5 to 25.5. In addition, the results in this section are not sensitive to the specification of Θ. As a related issue, in Appendix A.2 we examine the behavior of NNE when Θ fails to contain the true $\boldsymbol{\theta}$ and suggest a way to check whether Θ contains the true $\boldsymbol{\theta}$.

We specify the moments $\boldsymbol{m}$ as follows. First, we include: (i) the mean of $\boldsymbol{y}_{ij}$ and (ii) the cross-covariance matrix between $\boldsymbol{y}_{ij}$ and $\boldsymbol{x}_{ij}$.[12] We construct additional moments by aggregating $\boldsymbol{y}_{ij}$ to the consumer level. Specifically, let $\widetilde{\boldsymbol{y}}_i$ collect three variables: whether consumer i made any non-free search, how many searches the consumer made, and whether the consumer made a

[12] We do not include the covariance matrix of $\boldsymbol{y}_{ij}$ because it is implied by the mean of $\boldsymbol{y}_{ij}$. To see it, note that $\boldsymbol{y}_{ij}$ consists of two dummy variables and the purchase dummy can be 1 only when the search dummy is 1.

purchase. We add to $\boldsymbol{m}$: (iii) the mean of $\widetilde{\boldsymbol{y}}_i$, (iv) the cross-covariance matrix between $\widetilde{\boldsymbol{y}}_i$ and $J^{-1}\sum_{j=1}^{J}\boldsymbol{x}_{ij}$, and (v) the covariance matrix of $\widetilde{\boldsymbol{y}}_i$. In total, there are $2+14+3+21+6=46$ moments in $\boldsymbol{m}$. Alternative specifications of $\boldsymbol{m}$ are examined in Section 4.5.

The NNE in this section uses loss function C_2 with a diagonal $\boldsymbol{V}$ (see Section 2.3). Unless stated otherwise, we use $L^*=1e4$ training and validation datasets. The neural net uses ReLu activation function (sigmoid gives qualitatively the same results). We use 64 hidden nodes (it is a convention to use powers of 2). This number of hidden nodes is chosen by the validation loss. The estimation accuracy of NNE is not sensitive to the number of hidden nodes, so a coarse search (e.g., a few powers of 2) should be sufficient. See Appendix A.1 for more details.

A final detail on the implementation of NNE is that we trim the training and validation sets by excluding datasets where: nobody makes a purchase, everyone makes a purchase, nobody makes non-free searches, or everyone searches all options. These "corner cases" lack identification of $\boldsymbol{\theta}$. Although trimming reduces the training set size, we find it slightly improves the estimation accuracy of NNE. Intuitively, this is because the training can focus on realistic instead of corner cases.

SMLE for search model

The prevailing approach to estimate search model is SMLE. The main challenge of estimation lies in the enormous number (in millions/billions) of possible search and purchase choices by a consumer. To feasibly evaluate the likelihood over these choices, the literature applies smoothing on the likelihood function. Yet, choosing the right amount of smoothing is difficult (Geweke and Keane 2001, Ursu 2018). Below, we give more details on SMLE. Readers who are more interested in the results may skip these details without loss of continuity.

The most straightforward approach to evaluate likelihood by simulations is the accept-reject method. Specifically, fix a consumer i. We simulate R draws of $\{\varepsilon_{ij}\}_{j=0}^{J}$ and count the proportion of draws under which the model produces the same outcome as observed in the data. A problem here is that this proportion can often be zero. To see it, note the outcome of a consumer includes both her search and purchase choices. Conditional on the consumer searching k options, there are C_k^J possible search combinations and $k+1$ possible purchase choices. So, in total there are $\sum_{k=1}^{J}C_k^J\cdot(k+1)$ possible outcomes for a consumer. This number is over ten billion for $J=30$. However, a computationally feasible value of R is in the tens or hundreds. As a result, the observed outcome can often be evaluated as a probability-zero event.

The literature uses smoothing to address the probability-zero problem. Instead of counting only the simulations that exactly match the observed outcome, it allows a "partial match" between simulations and observed outcome. The definition of partial match is based on a set of inequalities that characterize optimal search and purchase choices. As an example, one inequality is that the maximum utility among searched options exceeds the reservation utility of any unsearched option.

We refer readers to Ursu (2018) for complete description of the inequalities.[13] A draw of $\{\varepsilon_{ij}\}_{j=0}^{J}$ is counted as partially matching the observed outcome if the inequalities are close to being satisfied. A smoothing factor λ calibrates the partial match. A larger λ is more lenient on unsatisfied inequalities. $\lambda \to 0$ is equivalent to the accept-reject method.

A difficulty in likelihood smoothing is choosing λ. The estimate $\widehat{\boldsymbol{\theta}}$ is often sensitive to λ. But there is no established procedure for choosing λ (see discussion in Geweke and Keane 2001). In practice, researchers resort to Monte Carlo experiments to try out different values of λ. The experiments require repeating SMLE many times for each trial value of λ, which is computationally expensive. Even then, it is unclear how to set $\boldsymbol{\theta}$ in these Monte Carlo experiments. One possibility is to use a preliminary estimate of SMLE under a guess of λ. But this preliminary estimate can be far from the true $\boldsymbol{\theta}$ due to the sensitivity of SMLE to λ.

In this section, the optimization in SMLE starts at the center of Θ specified above in NNE. The optimization result is not sensitive to the starting point – using other start points tends to lead to almost the same result. For the number of simulations, we use $R = 50$ (as in Ursu 2018) unless stated otherwise.

4.3 Monte Carlo studies

In the Monte Carlo studies, we first choose a "true" parameter value for the search model. Then, we use the search model to simulate a dataset and try to recover the parameter value from the dataset. We will refer to this simulated dataset as a "Monte Carlo dataset," in order to distinguish it from the also-simulated datasets used in training NNE.

Specifically, we generate a Monte Carlo dataset as follows. We set $n = 1000$ and $J = 30$ to resemble the real data that we will estimate later in Section 4.4. For the hotel attributes $\boldsymbol{x}_{ij}$, we draw each attribute from a distribution similar to its distribution in the real data.[14] We draw ε_{ij} from $\mathcal{N}(0, 1)$. Then, we compute $\boldsymbol{y}_{ij}$ using the optimal search and purchase strategy. The true parameter is set at $\boldsymbol{\beta} = (0.1,\ 0,\ 0.2,\ -0.2,\ 0.2,\ -0.2)'$, $\eta = 3$, and $\boldsymbol{\delta} = (-4, 0.1)$.

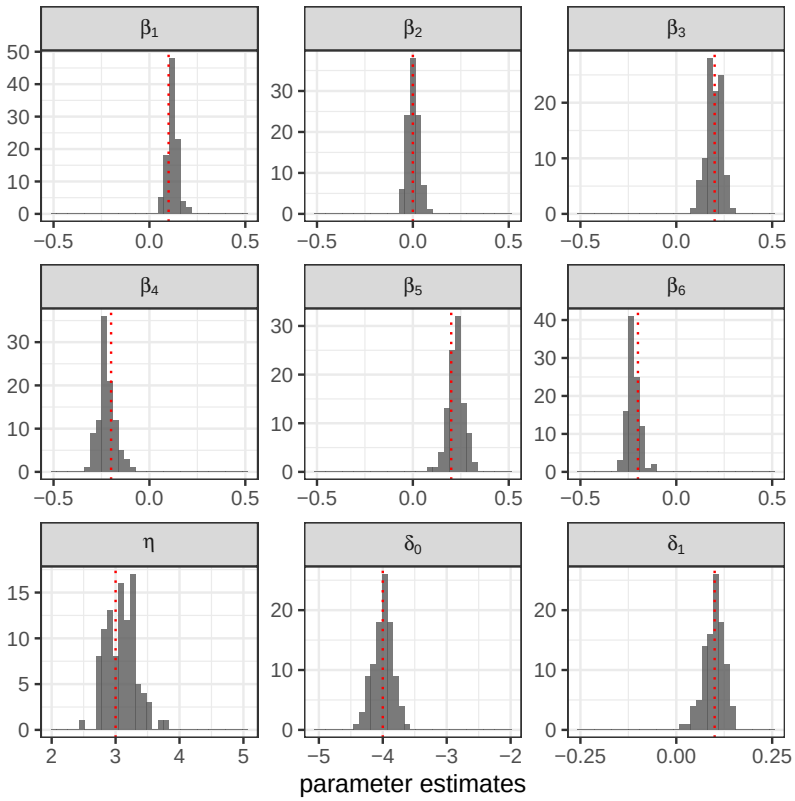
Below, we first examine the accuracy of parameter estimates and associated computational costs. We then examine the accuracy of the economic predictions implied by the parameter estimates.

Estimation accuracy and costs

For either NNE or SMLE, we repeat the estimation across 100 Monte Carlo datasets to assess the distribution of the estimates. Figure 4 plots the estimates by NNE. The red dotted lines show the true parameter values. We see that in all plots the estimates concentrate around the true values.

[13] A detail here is that some inequalities require knowing the search order. In the real data, this order is not observed and we follow Ursu (2018) to assume that higher-ranked options are searched first. In Monte Carlo, we make the search order observable to SMLE (which gives SMLE an advantage over NNE).

[14] The star rating takes 2, 3, 4, and 5 with probabilities 0.05, 0.25, 0.4, and 0.3, respectively. The review score takes 3, 3.5, 4, 4.5, and 5 with probabilities 0.08, 0.17, 0.4, 0.3, and 0.05, respectively. The location score follows $\mathcal{N}(4, 0.3)$. The dummy for chain hotels takes 1 with probability 0.8. The dummy for promotion takes 1 with probability 0.6. The log price follows $\mathcal{N}(0.15, 0.6)$. The ranking position s_{ij} enumerates from 1 to $J = 30$.



Notes: Each plot corresponds to one parameter in the search model. The red dotted line shows the true parameter value. The histogram shows the parameter estimates across 100 Monte Carlo datasets. The range of the horizontal axis equals the parameter's range in Θ.

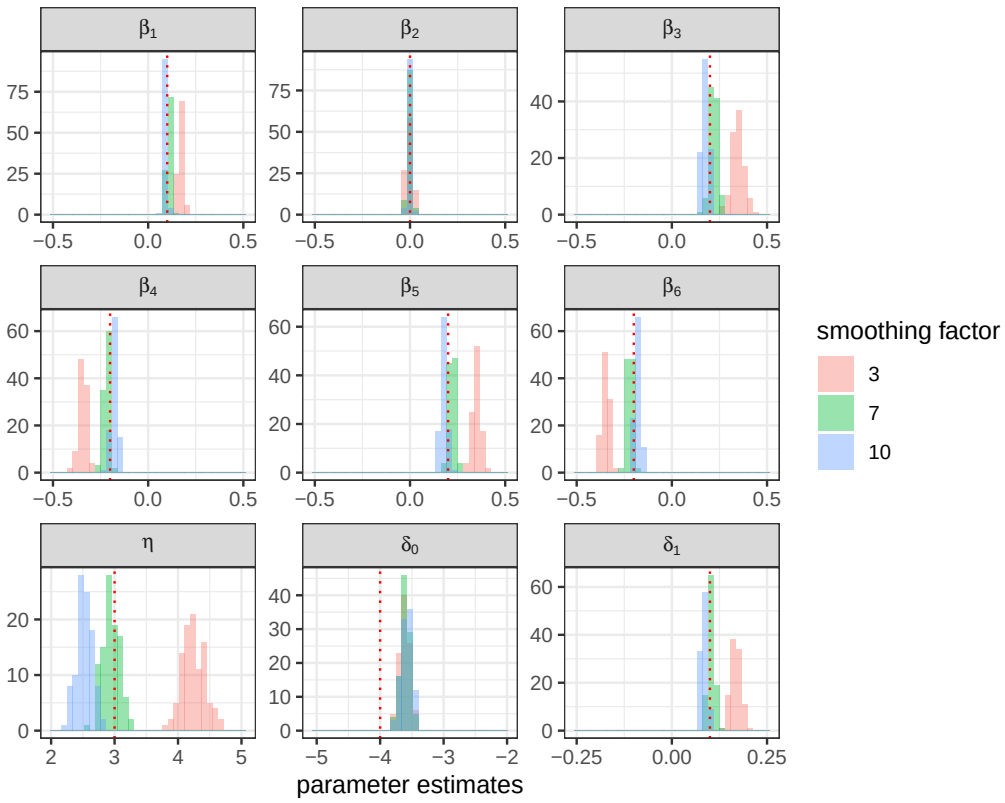
Figure 4: NNE's Estimates for Search Model in Monte Carlo

Some biases may be noticeable but are relatively small.

Figure 5 plots the estimates by SMLE under three smoothing factors: $\lambda = 3$, 7, and 10. In particular, we include $\lambda = 7$ because it is the optimal smoothing factor for SMLE here. Specifically, we use a grid of λ from 1 to 15. For each point on the grid, we repeat SMLE over 100 Monte Carlo datasets to compute a RMSE. The RMSE is smallest at $\lambda = 7$. We give the details in Appendix A.3. It is worth noting that obtaining this optimal λ not only is computationally expensive but also requires knowing the true value of $\boldsymbol{\theta}$ (to calculate RMSE). More generally, it is known that finding optimal λ is difficult in the estimation of search models.

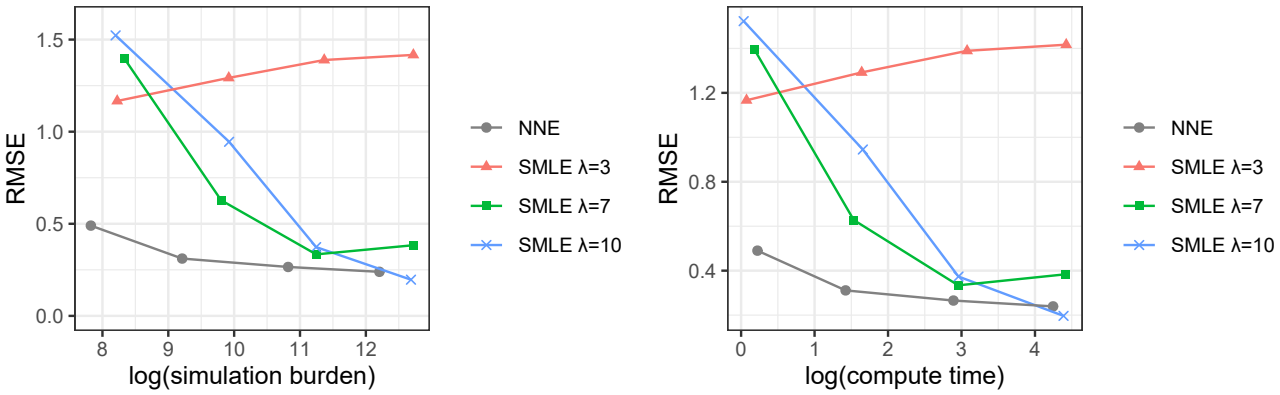
Figure 5 shows that the estimates by SMLE are sensitive to λ. We also see that all three values of λ lead to some biases. As we will show in a moment, both model fit and counterfactual analysis are more accurate with NNE than SMLE, even when SMLE uses the optimal $\lambda = 7$. It is also worth noting that the SMLE here has a higher computational cost than the NNE in Figure 4. We now turn to a more detailed comparison of computational costs.

Figure 6 plots the estimation accuracy of SMLE and NNE as we vary computation costs (by varying L^* and R). We measure estimation accuracy by RMSE. We measure computational costs in two ways: simulation burden and compute time. The left plot of Figure 6 shows the simulation



Notes: Same as Figure 4.

Figure 5: SMLE's Estimates for Search Model in Monte Carlo



Notes: The error in RMSE is defined as the Euclidean distance between the true $\boldsymbol{\theta}$ and its estimate. Simulation burden is the number of search model simulations required to carry out estimation. Simulation burden and compute time are varied by changing L^* and R: $L^* \in \{2500, 1e4, 5e4, 2e5\}$, $R \in \{5, 25, 100, 400\}$. The number of hidden nodes increases with L^* and we set it to be roughly $\propto \sqrt{L}$ (see Section 2.4 and Appendix B).

Figure 6: RMSE vs. Computational Cost in Search Model

burden, which is the number of search model simulations required to carry out the estimation (for NNE this burden equals L^*; for SMLE this burden equals R times the number of likelihood evaluations in optimization). The simulation burden is invariant to hardware or code implementation. The right plot of Figure 6 shows the total compute time, with our hardware and code.[15]

The results from the two plots in Figure 6 are consistent. NNE has better estimation accuracy than SMLE for a wide range of computation costs, and the advantage is especially large at low computation costs. We see that the estimation accuracy of SMLE is sensitive to λ and the optimal λ changes with computational cost (or R). The sensitivity to λ makes it important to choose the right λ. However, as discussed before, choosing λ is difficult (a comparison like Figure 6 requires repeating SMLE many times as well as knowing the true $\boldsymbol{\theta}$).

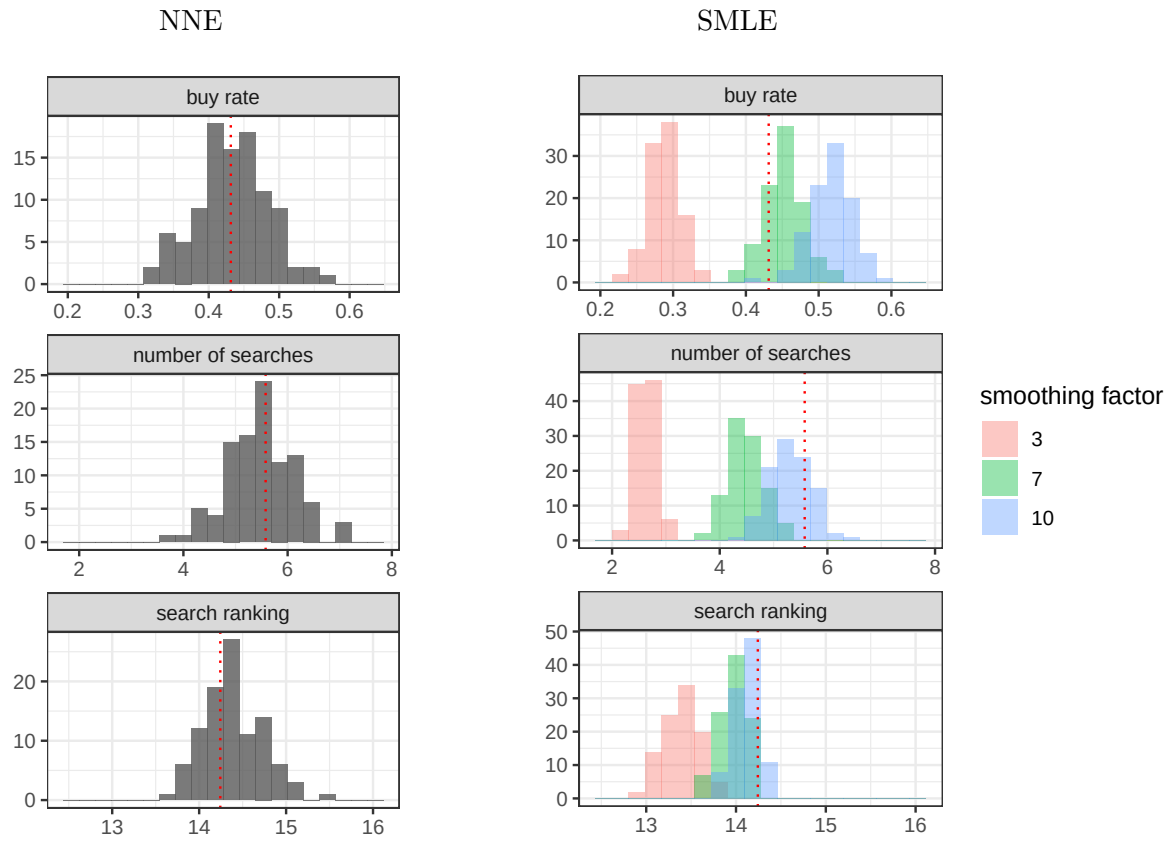
Implications by estimates

We examine how the parameter estimates by NNE and SMLE translate into economic predictions about consumer behaviors. We first look at model fit and then counterfactual analysis.

Figure 7 shows the model fit on three key statistics. In each plot, the red dotted line shows the average value observed across 100 Monte Carlo datasets. The histogram shows the values predicted by the search model estimated with these Monte Carlo datasets. The histogram's spread captures the statistical uncertainty in both $\widehat{\boldsymbol{\theta}}$ and the key statistic.

From Figure 7 we see that the predictions under NNE all center around the observed averages. The SMLE again shows sensitivity to the smoothing factor λ, and it shows clear biases under all three smoothing factors. Biases are present even with the optimal $\lambda = 7$ obtained with expensive computations and knowledge of true $\boldsymbol{\theta}$.

[15] A further computational benefit of NNE, not illustrated in Figure 6, is that it is highly parallelizable. All training datasets can be computed at once in parallel, which is useful with computer clusters.



Notes: Plots are based on 100 Monte Carlo datasets (all generated under the true $\boldsymbol{\theta}$). Take the 1st plot as example. On each Monte Carlo dataset, we estimate the search model and calculate the buy rate of a dataset simulated by the estimated search model. The histogram plots the buy rates calculated as such across the 100 Monte Carlo datasets. The red dotted line shows the average of the buy rates directly observed in the Monte Carlo datasets.

Figure 7: Model Fit on Key Statistics in Monte Carlo

Table 3: Search Model Counterfactual – What if No Search Costs

	Truth	NNE	SMLE ($\lambda = 3$)	SMLE ($\lambda = 7$)	SMLE ($\lambda = 10$)
Δ in buy rate	0.141	0.135	0.161	0.173	0.168
	(.001)	(.001)	(.001)	(.001)	(.001)

Notes: Results are averaged across 100 Monte Carlo datasets. On each Monte Carlo dataset, we first estimate the search model and then use the estimated search model to simulate the counterfactual. Numbers in parentheses are standard errors for the averages.

We now turn to counterfactual analysis. We focus on a counterfactual that computes the impact of search cost on purchases. Specifically, given a value of $\boldsymbol{\theta}$, we can use the search model to compute the increment in buy rate if consumers no longer incur search costs (with everything else unchanged). This increment measures the purchases lost due to search costs. It provides the platform with an upper bound on the effect of interventions to reduce search costs (e.g., better web design, better ranking algorithms).

Table 3 displays the counterfactual results under the true $\boldsymbol{\theta}$, NNE, and SMLE. The increment in buy rate is closest to the truth when we use NNE's estimate. SMLE gives us a considerably higher increment than the truth. Intuitively, this is because search costs are over-estimated with SMLE (see Figure 5).

4.4 Real data estimation

We estimate the search model on a real dataset. Specifically, we use the Expedia hotel search data in the main analysis of Ursu (2018). There are $n = 1055$ search sessions (which can be treated as individual consumers in this search model). The number of options $J = 33$ for some consumers and $J = 34$ for the others. Table 4 reports the results.[16] We examine model fit later in Figure 8.

The top panel of Table 4 reports the computational costs, measured in terms of simulation burden and compute time. As defined before, simulation burden is the number of search model simulations required to carry out estimation, and compute time is total time used to carry out estimation by our machine and code. Overall, we see the computational cost of NNE is a number of times smaller than SMLE.

In the lower panel of Table 4, the first column reports the estimates by NNE. We see that the hotel star rating, location score, and promotion have positive effects on the consumer utility. The review score has a negative effect (as in Ursu 2018), though the estimate is not statistically significant. Price has a negative effect and it is statistically significant. As to search costs, the ranking effect (δ_1) is positive but not statistically significant. The lack of significance for δ_1 is likely because we allow a free search and there are only 8.8% (or 93) consumers who searched beyond the free search, which makes it statistically difficult to pin down how ranking affects search cost. Overall, the NNE's estimates are reasonable.

[16] Our model setup largely follows Ursu (2018) but there are a few differences, so the estimates are not directly comparable. Specifically, we assume free first search and take the logs of ranking and price. See Section 4.1.

Table 4: Search Model Estimates on Real Data

	NNE		SMLE ($\lambda = 3$)		SMLE ($\lambda = 7$)		SMLE ($\lambda = 10$)	
Simul. burden ($\times 10^4$)	1.00		9.09		8.28		8.24	
Compute time (mins)	3.83		33.27		30.34		30.25	
Stars, β_1	0.215	(.078)	0.218	(.109)	0.111	(.063)	0.081	(.047)
Review, β_2	-0.089	(.107)	-0.046	(.188)	-0.005	(.106)	0.001	(.076)
Location, β_3	0.111	(.108)	0.178	(.196)	0.080	(.108)	0.056	(.077)
Chain, β_4	0.018	(.093)	-0.012	(.119)	-0.007	(.067)	-0.007	(.049)
Promotion, β_5	0.183	(.091)	0.105	(.108)	0.050	(.062)	0.037	(.045)
Price, β_6	-0.240	(.080)	-0.390	(.117)	-0.208	(.071)	-0.155	(.052)
Outside, η	4.038	(.594)	3.670	(.927)	2.803	(.51)	2.557	(.361)
Search, δ_0	-2.953	(.259)	-0.307	(.260)	-1.290	(.191)	-1.567	(.151)
Search, δ_1	0.049	(.036)	-0.147	(.093)	-0.083	(.067)	-0.061	(.053)

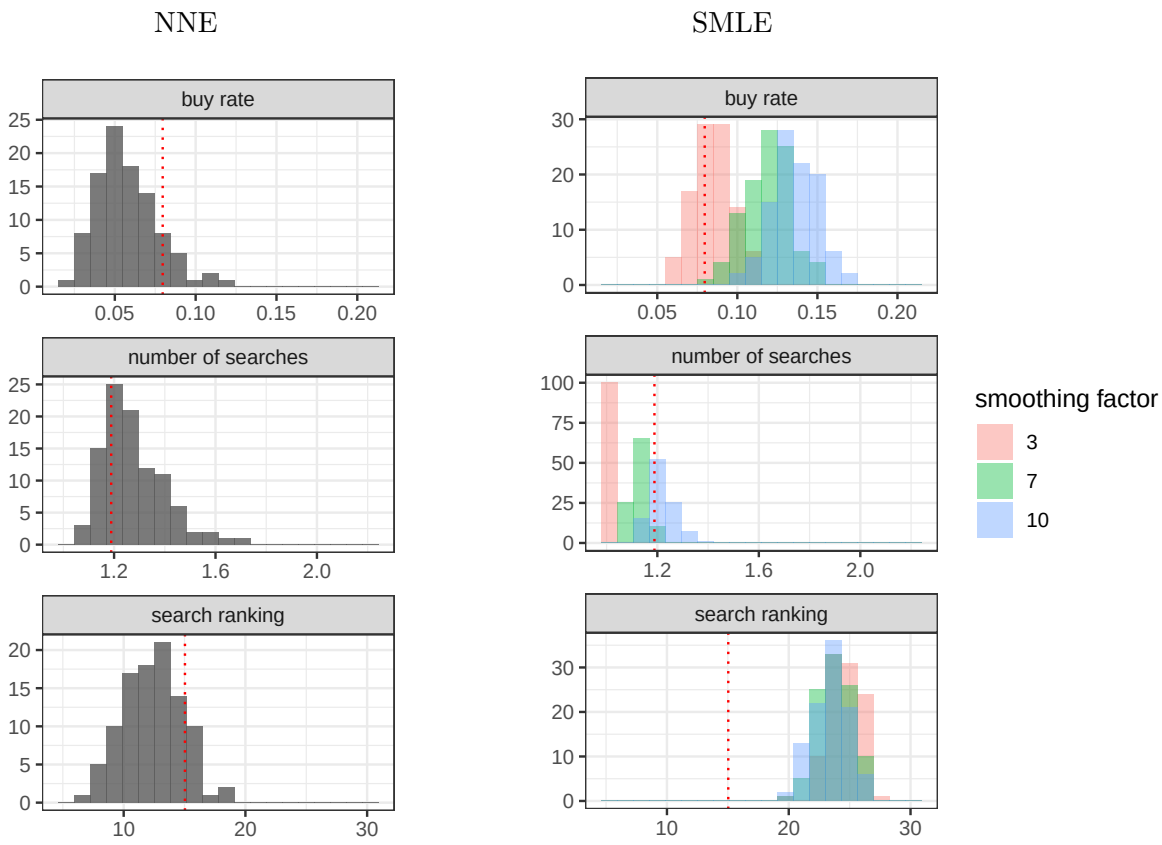
Notes: Numbers in parentheses are asymptotic standard errors (for SMLE) and estimates of statistical accuracy outputted by neural net (for NNE). All reported numbers are averaged across 100 estimation runs on the real data (estimates vary slightly between estimation runs because SMLE and NNE are both simulation-based).

The next columns in Table 4 report the estimates by SMLE under different smoothing factors. Consistent with what we have seen in the Monte Carlo studies, estimates are sensitive to the smoothing factor λ. In particular, SMLE's estimates for $\boldsymbol{\beta}$ and η tend to be closest to NNE's when λ is small. However, SMLE's estimates for $\boldsymbol{\delta}$ tend be to farthest from NNE's when λ is small. This sensitivity to λ is problematic because, as discussed before, choosing the optimal λ is practically difficult.

Unlike in Monte Carlo studies, we do not know the true value of the search model parameter $\boldsymbol{\theta}$. Therefore, we cannot calculate the RMSE of either NNE or SMLE. So instead, we focus on model fit on three key statistics. Figure 8 gives the results. In each plot, the red dotted line shows the value observed in the real data. The histogram shows the predicted values by estimated search model. The histogram's spread captures the statistical uncertainty in both $\widehat{\boldsymbol{\theta}}$ and the key statistic.

Figure 8 shows that NNE leads to a reasonable (albeit far from ideal) fit on the key statistics. With SMLE, the fit varies substantially by the smoothing factor λ. This sensitivity of SMLE to λ is problematic because, as discussed before, it is practically difficult to choose λ. More specifically, SMLE can give a good fit on either the buy rate or the number of searches, but not both. As to the search ranking, SMLE shows a substantially worse fit than NNE. Overall, NNE seems to give a better model fit.

The main findings from real data estimation are consistent with our findings in Monte Carlo studies. The root of the difficulty of SMLE lies in the enormous number of search and buy combinations. Without smoothing, an infeasibly large number of simulations will be needed to integrate out the unobservables for likelihood evaluation. Smoothing makes SMLE feasible, but it also introduces biases. In contrast, NNE does not require evaluating integrals over unobservables.



Notes: The histograms are based on 100 datasets bootstrapped from the real data. Take the 1st plot as example. On each bootstrap dataset, we estimate the search model and calculate the buy rate of a dataset simulated by the estimated search model. The histogram plots the buy rates calculated as such across the 100 bootstrap datasets. The red dotted line shows the buy rate observed in real data.

Figure 8: Model Fit on Key Statistics in Real Data

Table 5: NNE in Search Model with Alternative Moment Specifications

Number of moments	Total |bias|	RMSE
16	1.210 (.027)	0.649 (.017)
32	1.201 (.026)	0.630 (.015)
40	0.914 (.029)	0.508 (.015)
46	0.183 (.029)	0.311 (.017)
60	0.186 (.035)	0.291 (.015)
81	0.171 (.033)	0.279 (.015)

Notes: Each row corresponds to one specification of moments. Total |bias| is the sum of absolute biases across the parameters of the search model. Results are based on 100 Monte Carlo datasets. Numbers in parentheses are standard errors.

4.5 Additional results on search model

We provide further results on the performance of NNE in estimating the consumer search model. In particular, we examine the impacts of moment choice and data size.

Choice of moments

We measure how the estimation accuracy of NNE varies as the moment choice varies. This exercise is related to the general discussion in Section 2 on NNE's robustness to redundant moments. (Also see Section 3 for a similar exercise under the much simpler AR(1) model.)

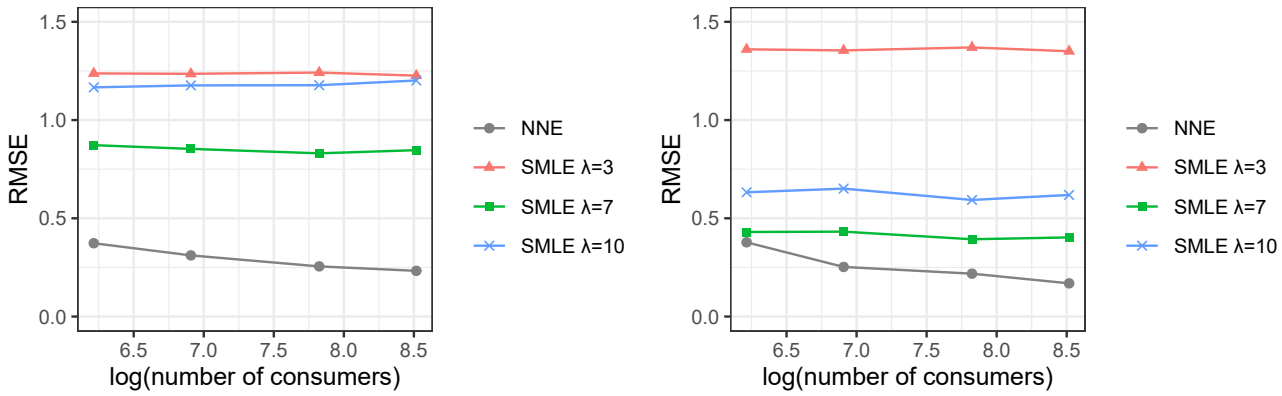
Recall that so far we have included 46 data moments in $\boldsymbol{m}$ (see Section 4.3). We now construct five alternative specifications of $\boldsymbol{m}$ by removing or adding moments. The first specification removes the covariance matrix of $\widetilde{\boldsymbol{y}}_i$ (40 moments remain). The second specification further removes moments by dropping the non-free search dummy from $\widetilde{\boldsymbol{y}}_i$ (32 moments remain). The third specification removes $\widetilde{\boldsymbol{y}}_i$ all together (16 moments remain). The fourth specification adds to the original 46 moments the cross-covariances between $\boldsymbol{y}_{ij}$ and $\boldsymbol{x}_{ij}^2$ (60 moments total). The fifth specification further adds the cross-covariances between $\widetilde{\boldsymbol{y}}_i$ and $J^{-1}\sum_{j=1}^{J}\boldsymbol{x}_{ij}^2$ (81 moments total).

Table 5 reports the estimation accuracy under the six specifications of moments, with each row corresponding to one specification. The first column lists the numbers of moments. We do not see sizable increases in bias or RMSE as we move from 16 to 81 moments. Instead, the bias and RMSE largely decrease. The result suggests that NNE is able to capitalize on the information from additional moments while being robust to potentially redundant moments.

Data size

Our analysis so far has focused around data size $n = 1000$. We now examine how the estimation accuracy of NNE varies with data size and compare it to SMLE. To facilitate the comparison, we keep the computational costs about the same between NNE and SMLE (by using appropriate L^* and R).

Figure 9 reports the results for $n \in \{500, 1000, 2500, 5000\}$. The two plots differ in computational costs, and a higher cost is used in the right plot. Within each plot, NNE and SMLE are



Notes: All reported numbers are averaged across 100 Monte Carlo datasets. Within each plot, L^* and R are chosen so that NNE and SMLE have about the same simulation burdens. In the left plot $L^* = 1e4$ and $R = 15$. In the right plot $L^* = 4e4$ and $R = 50$.

Figure 9: Estimation of Search Model with Different Data Sizes

made to have about the same computational costs. For SMLE we include three smoothing factors: $\lambda = 3, 7$, and 10. As before, we include $\lambda = 7$ because a grid search finds it optimal for SMLE at $n = 1000$ (Figure 9 here suggests that $\lambda = 7$ is also optimal for the other values of n). We again note that finding this optimal λ not only is computationally expensive but also uses the knowledge of the true search model parameter.

The most important observation from Figure 9 is that the RMSE of NNE decreases with n faster than SMLE. In other words, NNE can capitalize on larger data better. Intuitively, this is because the smoothing required by SMLE introduces bias in estimates (as we have seen in Section 4.3). This bias is not a finite-sample bias and tends not to decrease with n. To reduce this bias and subsequently RMSE, one needs to increase R and use a larger λ (the optimal λ increases with R). However, a larger R also implies a higher computational cost.

5 Conclusion

We study a novel approach that leverages machine learning to estimate the parameters of (structural) econometric models. One uses the structural econometric model to generate datasets. These generated datasets, together with the parameter values under which they are generated, can be used to train a machine learning model to "recognize" parameter values from datasets. As such, the approach offers a bridge to connect the existing machine learning techniques and the current structural econometric models.

NNE has its advantages and boundaries. We find that it has a well-defined and meaningful limit, is robust to redundant moments, and achieves good estimation accuracy at light computational costs in suitable applications. The applications that can benefit the most from NNE are where SMLE or SMM would require large numbers of simulations. NNE is unlikely to show gains for applications where the main estimation burden is not simulations. One example is where closed-form likelihood

functions are available. Another example is dynamic choice models where solving the economic model constitutes the main burden in estimation. NNE also requires that the econometric model can be used to simulate data.

The paper leaves several possibilities unexplored, which open avenues for future research. The first possibility is extension to very large-scale problems where the econometric model has hundreds or more parameters, including nuisance parameters. It is useful to note that although in this paper NNE outputs all the parameters of the econometric model, it is straightforward to configure NNE to output only a subset of parameters. One can exclude nuisance parameters from this subset to focus on the main parameters. In addition, while we have focused on shallow neural nets for a balance between learning capacity and ease of training, for very large-scale problems one may explore deep neural nets.

The second unexplored possibility is to identify the relevant moments. This exercise is useful because it can help to clarify the sources of econometric identification and makes structural estimation more transparent. NNE seems particularly well positioned for this exercise. A well-trained neural net should automatically disregard irrelevant inputs. However, a difficulty lies in the interpretability of neural nets. The literature on interpretable machine learning may shed lights on how to identify the inputs that a neural net disregards.

The third unexplored possibility is pre-training NNE. For any given structural econometric model, a researcher can train a NNE and make it available to other researchers who wish to apply the structural model on their own data. One difficulty to overcome is that the pre-trained NNE must not depend on a specific $\boldsymbol{x}$ (recall that $\boldsymbol{x}$ is conditioned on in this paper). However, if made possible, pre-trained NNE packages will significantly reduce the costs of researchers applying structural estimation. An additional benefit exists in handling data privacy. Because NNE only requires data moments as inputs, with a pre-trained NNE one can carry out the estimation without access to the full data. This feature is particularly useful when researchers collaborate with government agencies or companies to study data that contain sensitive personal information. Researchers will not require access to individual-level data but only aggregated data moments to estimate structural models.

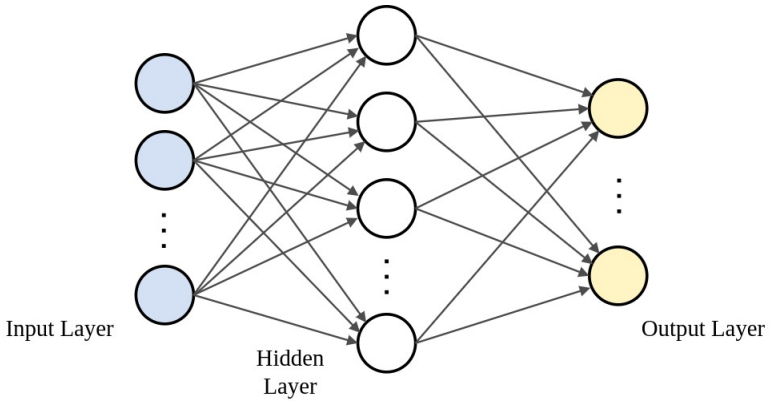


Figure 10: A Shallow Neural Network

A Appendix

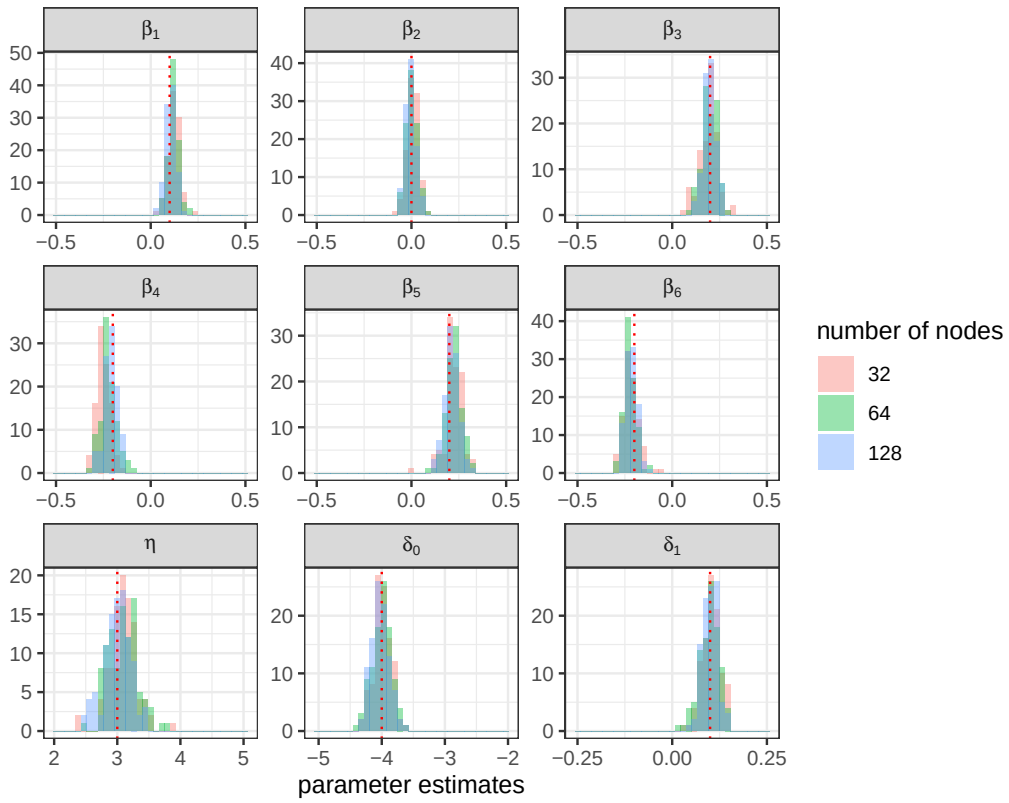
A.1 Number of hidden nodes

Figure 10 visually presents a single-hidden-layer neural net by a collection of nodes and arrows. Neural nets with a single hidden layer are often referred to as shallow neural nets. For shallow neural nets, an important configuration choice is the number of nodes in the hidden layer. As we add more nodes, the neural net becomes more flexible and capable of learning more complex mappings. However, large neural networks may overfit the training set. The appropriate number of hidden nodes is usually chosen by validation loss.

In the search model, we use validation loss to choose the number of hidden nodes for $L^* = 1e4$ (which is the setting in most of our results). We train three neural nets with 32, 64, and 128 hidden nodes, respectively (it is a convention to use powers of 2). The training uses examples $\ell = 1, ..., L$. Then, we calculate validation losses with examples $\ell = L+1, ..., L^*$. The 64-node neural net typically has the smallest validation loss. So we use 64 hidden nodes for $L^* = 1e4$ in all results. In fact, if averaged across 100 Monte Carlo datasets, the validation losses (C_2) with 32, 64, and 128 hidden nodes are -19.32, -20.16, and -19.39.

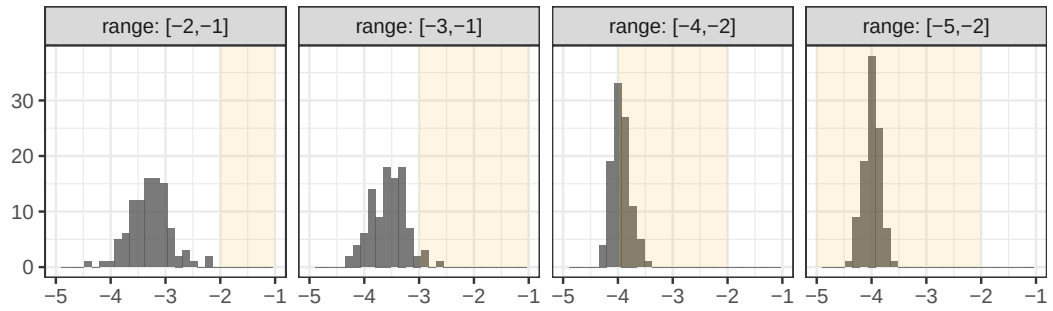
It is useful to note that in choosing the neural net configuration via validation, all the candidate configurations re-use the same training set and validation set. We do not need to simulate additional datasets using the econometric model. So, choosing the neural net configuration is relatively low-cost in applications where simulations of the structural econometric model constitute the bulk of the computational cost.

A related question is how sensitive NNE's estimates are to the neural net configuration? If they are not sensitive, one can use a coarse grid to search for the optimal configuration. Otherwise, one needs to do a fine search. Figure 11 plots the estimates by NNE under different numbers of hidden nodes. The histograms largely overlap, which indicates that estimates are not sensitive to the number of hidden nodes. This result suggests that a coarse grid (e.g., a few powers of 2) should be sufficient for choosing the number of hidden nodes.



Notes: Each plot corresponds to one parameter in the search model. The histogram shows the parameter estimates across 100 Monte Carlo datasets. The range of the horizontal axis equals the parameter's range in Θ.

Figure 11: Estimates with Different Neural Net Configurations



Notes: Each histogram shows the estimates of δ_0 by NNE across 100 Monte Carlo datasets. The four plots differ in the specified range of δ_0 in Θ. The shaded area in each plot shows the specified range of δ_0. The true value of $\delta_0 = -4$.

Figure 12: Sensitivity Analysis of NNE to Θ

A.2 Parameter space Θ

In NNE, the parameter values of the econometric model in training and validation sets are drawn from a parameter space Θ. A practical question is how NNE will behave if Θ fails to contain the true value of $\boldsymbol{\theta}$. We explore this question in consumer search model (Section 4). We will focus on the search cost parameter δ_0, but results with other parameters are similar. Recall that we have specified the range for δ_0 as $[-5, -2]$ and the true value of $\delta_0 = -4$.

Figure 12 shows the distributions of NNE's estimates for δ_0 under different specified ranges of δ_0. The ranges in the first two plots do not include the truth $\delta_0 = -4$. In both plots, most of the estimates fall between the specified range and the true value. The range in the third plot has the truth at its boundary. The range in the last plot contains the truth in its interior. In these two plots, the estimates closely center around the truth.

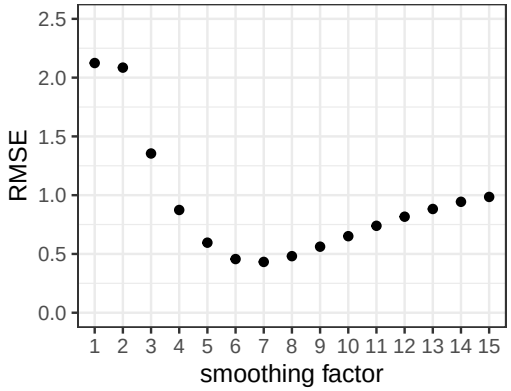
Overall, we see that NNE is well-behaved even when Θ is mis-specified. The estimate typically falls outside Θ in the direction towards the true parameter value. This behavior suggests we check whether NNE's estimate is inside Θ. If not, then Θ likely does not contain the truth and needs to be adjusted.

A.3 Choosing smoothing factor for SMLE

To estimate the consumer search model with SMLE requires a smoothing factor. We use grid search to find the optimal smoothing factor λ for $R = 50$ (which is the setting in most of our results). Specifically, we use a grid of λ from 1 to 15. For each value on the grid, we repeat SMLE across 100 Monte Carlo datasets to calculate the RMSE (so the SMLE is repeated 1500 times in total). Figure 13 reports the results. We see $\lambda = 7$ achieves the lowest RMSE.

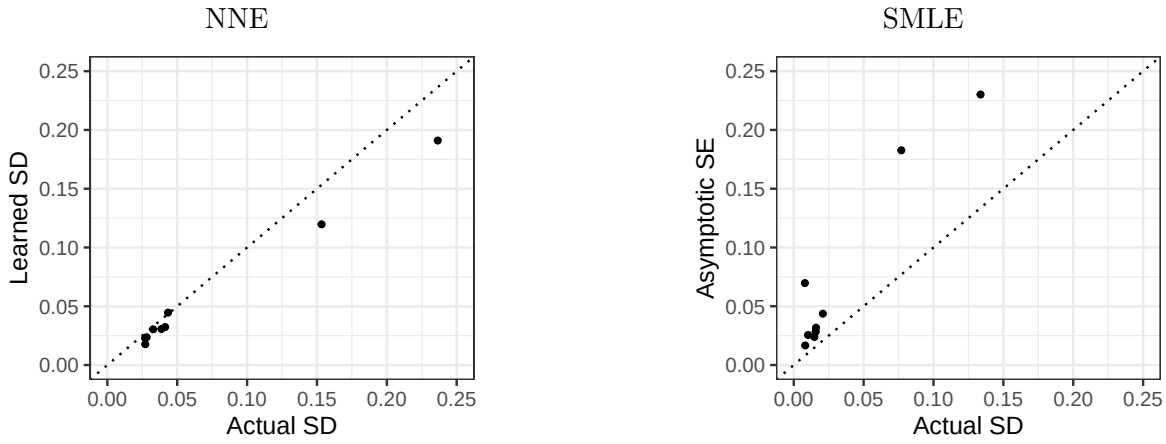
A.4 Estimates of statistical accuracy

Section 2.3 describes how NNE can output an estimate of statistical accuracy in addition to the point estimate. Section 2.4 provides the theoretical properties of this estimate of statistical accuracy.



Notes: For each value of the smoothing factor, we repeat SMLE across 100 Monte Carlo datasets to calculate the RMSE. SMLE here uses $R = 50$. The error in RMSE is defined as the Euclidean distance between the true value and the estimate of $\boldsymbol{\theta}$.

Figure 13: Optimal Smoothing Factor in SMLE



Notes: Each point represents one parameter in the search model. Horizontal axis shows the standard deviation of the point estimates across 100 Monte Carlo datasets. For NNE (left plot), vertical axis shows the estimate of statistical accuracy outputted by neural net, averaged across Monte Carlo datasets. For SMLE (right plot), vertical axis shows the asymptotic standard error, averaged across Monte Carlo datasets.

Figure 14: Estimates of Statistical Accuracy

Below, we examine this estimate of statistical accuracy in the consumer search model, by comparing it to the standard deviation of the point estimates across Monte Carlo datasets.

The left plot of Figure 14 shows the results. Each point represents one parameter in the search model (there are 9 in total). The horizontal axis shows the standard deviation of the point estimates. The vertical axis shows the estimate of statistical accuracy given by NNE. Recall that in the search model we have used the loss function C_2 with a diagonal $\boldsymbol{V}$. So the estimates of statistical accuracy are taken as square roots of the diagonal entries. We see that the points are not far from the 45° line, indicating that NNE estimates the statistical accuracy reasonably well.

As a comparison, the right plot of Figure 14 repeats the above exercise but for SMLE ($\lambda = 7$). The vertical axis shows the asymptotic standard error. We see larger deviations from the 45° line. The deviations are likely caused by smoothing. Smoothing affects the likelihood function's slope and curvature, which the asymptotic formula relies on. Thus, it is not surprising that smoothing can distort the asymptotic standard errors of SMLE.

A.5 Regularized polynomial instead of neural net

We examine how the estimation accuracy of NNE is affected if the neural net is replaced by polynomial regression. Specifically, we train a (multivariate) polynomial from $\boldsymbol{m}$ to the econometric model's parameter. For example, if $\boldsymbol{m}$ collects 10 moments, a second-degree polynomial will have 65 terms and a third-degree polynomial will have 285 terms. The polynomial is estimated using lasso. The regularization factor is chosen using the validation set ($\ell = L+1, ..., L^*$).

Table 6 replicates Table 2 but with lasso polynomials. There are two observations when we compare the numbers in Table 2 to Table 6. The first observation is that the lasso shows larger biases than the neural net (while not reported in the table, the biases with fourth-degree polynomial are still about twice the biases with the neural net). This observation shows that the polynomials are not as flexible to capture the relation between $\boldsymbol{m}$ and β. The second observation is that unlike SMM, lasso does not show clear increases in bias as we add moments. So lasso exhibits some robustness to redundant moments that we have seen with neural nets. This observation is intuitive because both the neural net and lasso can use the training set to learn which moments contribute to estimation.

A.6 Indirect inference

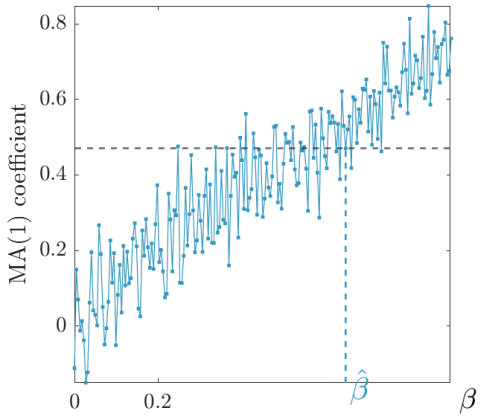
A graphical illustration as in Figure 2 can be made for indirect inference. Indirect inference extends SMM by allowing one to use the parameter of an auxiliary econometric model as moments. We note AR(1) is a simple model that does not require indirect inference to estimate. Nevertheless, the setting works well for illustrating the conceptual difference between indirect inference and NNE.

The auxiliary model is typically mis-specified. We use MA(1) as the auxiliary model: $y_i = \varepsilon_i + \alpha\varepsilon_{i-1}$ with $\varepsilon_i \sim \mathcal{N}(0,1)$. There are two common estimators for MA(1). The first estimator uses the auto-covariance in MA(1): $\mathbb{E}(y_i y_{i-1}) = \alpha$, which suggests estimating α with $\widehat{\alpha}^{AC} = \frac{1}{n-1}\sum_{i=2}^{n} y_i y_{i-1}$. The superscript "AC" signifies auto-covariance. Note that $\widehat{\alpha}^{AC}$ coincides with the

Table 6: Estimation of AR(1) with Different Moments, Lasso

Moments	Lasso degree 2 (L=1e3)		Lasso degree 3 (L=1e3)	
	Bias	RMSE	Bias	RMSE
1) $y_i y_{i-1}$	-0.102 (.003)	0.132 (.002)	-0.055 (.003)	0.114 (.002)
2) $y_i y_{i-1}$, y_i^2	-0.067 (.002)	0.101 (.002)	-0.046 (.003)	0.100 (.002)
3) $y_i y_{i-k}, k = 1, 2, 3$	-0.066 (.003)	0.114 (.002)	-0.044 (.003)	0.106 (.002)
4) $y_i y_{i-k}, k = 1, 2, ..., 10$	-0.055 (.003)	0.112 (.002)	-0.040 (.003)	0.107 (.002)
5) $y_i y_{i-1}$, $y_i^2 y_{i-1}$, $y_i y_{i-1}^2$	-0.095 (.003)	0.130 (.002)	-0.048 (.003)	0.111 (.002)
6) $y_i y_{i-k}$, $y_i^2 y_{i-k}$, $y_i y_{i-k}^2$, $k = 1, 2, 3$	-0.065 (.003)	0.115 (.002)	-0.043 (.003)	0.108 (.002)

Notes: This table replicates Table 2, except that polynomials replace the neural net in NNE. Results are based on 1000 Monte Carlo datasets. Numbers in parentheses are standard errors.



Notes: Jagged blue curve shows the least-square estimate of the MA(1) coefficient as a function of β.

Figure 15: Indirect Inference for AR(1)

moment used by SMM in Section 3.1. Thus, the consequent indirect inference estimator for β is effectively the same as the SMM illustrated in Figure 2(a) in Section 3.

The second estimator for MA(1) uses least squares. It typically specifies $\varepsilon_0 = 0$ and then uses the MA(1) to compute $\varepsilon_1 = y_1$, $\varepsilon_2 = y_2 - \alpha\varepsilon_1$, $\varepsilon_3 = y_3 - \alpha\varepsilon_2$, ... One estimates α with $\widehat{\alpha}^{LS} = \text{argmin}_\alpha \sum_{i=1}^n \varepsilon_i^2$. The consequent indirect inference estimator is illustrated in Figure 15. The jagged curve shows, for each β, the value of $\widehat{\alpha}^{LS}$ estimated from a dataset simulated with AR(1) under that β. Suppose that the horizontal dashed line shows the value of $\widehat{\alpha}^{LS}$ from real data. Then, indirect inference estimates β by the point on the jagged curve closest to the horizontal dashed line, marked as $\widehat{\beta}$ on the horizontal axis.

The discussion above shows the conceptual similarity between indirect inference and SMM that they both attempt approximations *point-by-point* over β. In contrast, NNE attempts to learn a *functional* approximation (discussed in Section 3).

B Proofs

We first set up the preliminaries for proving Proposition 1-3. Our asymptotics ($L \to \infty$) are stated for any fixed data size n. Conditional on the observed $\boldsymbol{x}$, 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 econometric model $\boldsymbol{q}$ and the specification of $\boldsymbol{m}$ as a function of $\{\boldsymbol{y}, \boldsymbol{x}\}$. Then, the training examples $\{\boldsymbol{\theta}^{(\ell)}, \boldsymbol{m}^{(\ell)}\}_{\ell=1}^{L}$ are i.i.d. samples from P.

We now work towards a lemma that states the general conditions for a sequence of 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 Δ 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 Δ. For example, $\boldsymbol{f}^*$ is $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ in Proposition 1.

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 focus on the sequence $\mathcal{F}_L$ constructed from shallow neural nets, which are sufficient for our purpose. Asymptotic properties of shallow neural nets have been well studied in econometric literature (deep neural nets are very recently studied by Farrell et al. 2021a). We adopt the shallow neural nets used in White (1990). Let $\psi : \mathbb{R} \to \mathbb{R}$ be any sigmoid function such as the logistic function. Let v denote the dimension of $\boldsymbol{m}$. Let

$$\mathcal{F}(u, b) \equiv \Bigg\{ \boldsymbol{f} \in \mathcal{F} : f_k(\boldsymbol{m}) = w_{1k0} + \sum_{j=1}^{u} w_{1kj} \cdot \psi\Big(w_{0j0} + \sum_{i=1}^{v} w_{0ji} \cdot m_i\Big), \\ \sum_{j=0}^{u} |w_{1kj}| \leq b, \ \sum_{j=1}^{u} \sum_{i=0}^{v} |w_{0ji}| \leq u \cdot b \Bigg\}. \tag{10}$$

In words, $\mathcal{F}(u, b)$ denotes the single-hidden-layer neural networks with u hidden units and weights $\boldsymbol{w}$ bounded in a way by b. A finite b makes $\mathcal{F}(u, b)$ 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 $\{\mathcal{F}_L\}_{L=1}^{\infty}$ where $\mathcal{F}_L = \mathcal{F}(u_L, b_L)$ with u_L and b_L that grow sufficiently slow.

By White (1989), a permissible choice is $u_L \propto \sqrt{L}$ and $b_L \propto \log(L)$.

We are in a position to state the lemma that links $\widehat{\boldsymbol{f}}_L$ to $\boldsymbol{f}^*$.

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

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

(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. $\sup_{\boldsymbol{f}\in\mathcal{F}_L} |C(\boldsymbol{f}) - \mathbb{E}\left[h(\boldsymbol{\theta}, \boldsymbol{f}(\boldsymbol{m}))\right]| \to 0$ 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 δ, 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.

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 how we construct the training examples in NNE. 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 u_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 $\boldsymbol{f}^* = \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$. We can take Δ as any compact set that contains $\text{conv}(\Theta)$ in its interior, which is always possible given that Θ is compact. The reason for taking the convex hull is to accommodate cases where some dimensions of Θ are discrete so that $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ may take values outside Θ. One simplest choice of Δ is $[\underline{\theta},\overline{\theta}]^p$, 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 Θ 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 Δ 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 δ.

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 Δ. 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 τ or δ 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 ∇ denote the set of all possible values of 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 ∇ specified as such is a compact subset of the convex cone for 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 Δ^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 Δ, 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 Γ 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. □

Funding and Competing Interests

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no funding to report.

References

Altonji, Joseph G and Lewis M Segal (1996) "Small-sample Bias in GMM Estimation of Covariance Structures," *Journal of Business & Economic Statistics*, 14 (3), 353–366.

Andersen, Torben and Bent Sorensen (1996) "GMM Estimation of a Stochastic Volatility Model: A Monte Carlo Study," *Journal of Business and Economic Statistics*, 14 (3).

Athey, Susan (2018) "The Impact of Machine Learning on Economics," in *The Economics of Artificial Intelligence: An Agenda*, 507–547: University of Chicago Press.

Bajari, Patrick, C Lanier Benkard, and Jonathan Levin (2007) "Estimating Dynamic Models of Imperfect Competition," *Econometrica*, 75 (5), 1331–1370.

Bao, Weining and Jian Ni (2017) "Could Good Intentions Backfire? An Empirical Analysis of the Bank Deposit Insurance," *Marketing Science*, 36 (2), 301–319.

Breusch, Trevor, Hailong Qian, Peter Schmidt, and Donald Wyhowski (1999) "Redundancy of Moment Conditions," *Journal of Econometrics*, 91, 89–111.

Bruins, Marianne, James A Duffy, Michael P Keane, and Anthony A Smith Jr (2018) "Generalized Indirect Inference for Discrete Choice Models," *Journal of Econometrics*, 205 (1), 177–203.

Chen, XiaoHong (2007) "Large Sample Sieve Estimation of Semi-nonparametric Models," *Handbook of Econometrics*, 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*, 186 (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*, 21(1).

Chiong, Khai and Matt Shum (2019) "Random Projection Estimation of Discrete-Choice Models with Large Choice Sets," *Management Science*, 65 (1), 256–271.

Collard-Wexler, Allan (2013) "Demand Fluctuations in the Ready-Mix Concrete Industry," *Econometrica*, 81 (3).

Dauphin, Yann, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio (2014) "Identifying and Attacking the Saddle Point Problem in High-dimensional Non-convex Optimization," *Advances in Neural Information Processing Systems*, 27.

Donald, Stephen G. and Whitney K. Newey (2021) "Choosing the Number of Instruments," *Econometrica*, 69 (5).

Du, Simon, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai (2019) "Gradient Descent Finds Global Minima of Deep Neural Networks," *International Conference on Machine Learning*, 97.

Farrell, Max H, Tengyuan Liang, and Sanjog Misra (2021b) "Deep Learning for Individual Heterogeneity: an Automatic Inference Framework," *arXiv preprint arXiv:2010.14694*.

Farrell, Max, Tengyuan Liang, and Sanjog Misra (2021a) "Deep Neural Networks for Estimation and Inference," *Econometrica*, 89 (1), 181–213.

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*, 5, 3463–3568: Elsevier.

Gourieroux, Christian, Alain Monfort, and Eric Renault (1993) "Indirect Inference," *Journal of Applied Econometrics*, 8 (S1), S85–S118.

Graham, Bryan S. (2020) "Network Data," *Handbook of Econometrics*, 7A, 111–218.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989) "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, 2 (5), 359–366.

Kaji, Tetsuya, Elena Manresa, and Guillaume Pouliot (2023) "An Adversarial Approach to Structural Estimation," *Econometrica*.

Kim, Jae Young (2002) "Limited Information Likelihood and Bayesian Analysis," *Journal of Econometrics*, 107.

Lee, Youji and Elizabeth L. Ogburn (2021) "Network Dependence Can Lead to Spurious Associations and Invalid Inference," *Journal of the American Statistical Association*, 116 (535), 1060–1074.

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

Li, Hao, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein (2018) "Visualizing the Loss Landscape of Neural Nets," *Advances in Neural Information Processing Systems*, 32.

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*, 56 (6), 918–943.

Newey, Whitney K. (2007) "Generalized Method of Moments," *MIT OpenCourseWare: New Econometric Methods*.

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*, 38 (2), 373–399.

Smith, Anthony (2008) "Indirect Inference," *The New Palgrave Dictionary of Economics, 2nd Edition*.

Su, Che-Lin and Kenneth L Judd (2012) "Constrained Optimization Approaches to Estimation of Structural Models," *Econometrica*, 80 (5), 2213–2230.

Timoshenko, Artem and John R Hauser (2019) "Identifying Customer Needs from User-generated Content," *Marketing Science*, 38 (1), 1–20.

Ursu, Raluca M (2018) "The Power of Rankings: Quantifying the Effect of Rankings on Online Consumer Search and Purchase Decisions," *Marketing Science*, 37 (4), 530–552.

Wager, Stefan and Susan Athey (2018) "Estimation and Inference of Heterogeneous Treatment Effects using Random Forests," *Journal of the American Statistical Association*, 113 (523), 1228–1242.

Weitzman, Martin L (1979) "Optimal Search for the Best Alternative," *Econometrica: Journal of the Econometric Society*, 641–654.

White, Halbert (1982) "Maximum Likelihood Estimation of Misspecified Models," *Econometrica: Journal of the Econometric Society*, 1–25.

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

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

Yoganarasimhan, Hema (2020) "Search Personalization using Machine Learning," *Management Science*, 66 (3), 1045–1070.

Yoganarasimhan, Hema, Ebrahim Barzegary, and Abhishek Pani (2023) "Design and Evaluation of Optimal Free Trials," *Management Science*, 69 (6), 3220–3240.

Zhang, Mengxia and Lan Luo (2023) "Can Consumer-posted Photos Serve as a Leading Indicator of Restaurant Survival? Evidence from Yelp," *Management Science*, 69 (1), 25–50.

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