Submitted to Statistical Science

Measurement error models: from nonparametric methods to deep neural networks

Zhirui Hu, Zheng Tracy Ke and Jun S Liu

Abstract. The success of deep learning has inspired a lot of recent interests in exploiting neural network structures for statistical inference and learning. In this paper, we review some popular deep neural network structures and techniques under the framework of nonparametric regression with measurement errors. In particular, we demonstrate how to use a fully connected feed-forward neural network to approximate the regression function $f(x)$, explain how to use a normalizing flow to approximate the prior distribution of X, and detail how to construct an inference network to generate approximate posterior samples of X. After reviewing recent advances in variational inference for deep neural networks, such as the importance weighted autoencoder, doubly reparametrized gradient estimator, and non-linear independent components estimation, we describe an inference procedure built upon these advances. An extensive numerical study is presented to compare the neural network approach with classical nonparametric methods, which suggests that the neural network approach is more flexible in accommodating different classes of regression functions and performs superior or comparable to the best available method in many settings.

Key words and phrases: Doubly reparametrized gradient estimator, error-in-variables, fully connected feed-forward neural network, importance weighted autoencoder, normalizing flow, SIMEX, variational autoencoder.

1. INTRODUCTION

The nonparametric regression with measurement errors is a classical topic in statistics and has received much attention (Carroll et al., 2006). In a common setting, the response $Y \in \mathbb{R}$ satisfies that $\mathbb{E}[Y \mid X] = f(X)$, where $X \in \mathbb{R}^d$ contains the covariates and $f : \mathbb{R}^d \to \mathbb{R}$ is an unknown regression function. The covariates are observed with additive errors: we observe

Zhirui Hu is Assistant Professor at Center for Statistical Science, Tsinghua University, Beijing, China (e-mail: zhiruihu33@gmail.com). Zheng Tracy Ke is Assistant Professor, Department of Statistics, Harvard University, Cambridge, U.S. (e-mail: zke@fas.harvard.edu). Jun S Liu is Professor, Department of Statistics, Harvard University, Cambridge, U.S. (e-mail: jliu@stat.harvard.edu).

$W = X + U$ instead of X, where U is a mean-zero random vector whose distribution is known. The goal is estimating the nonparametric function f from observations of (W, Y).

Nonparametric and semiparametric methods are popular for fitting measurement error models. Some notable examples include the deconvolution method (Fan and Truong, 1993), the simulation extrapolation method (Carroll, Maca and Ruppert, 1999), and the estimating equation method (Jiang, Ma and Carroll, 2018). These methods usually enjoy nice theoretical properties but they only apply to particular smooth function classes of f. Bayesian methods are also widely used. In a Bayesian framework, the joint distribution of (X, W, Y) is fully specified, and unknown parameters are estimated by maximizing the marginal likelihood of (W, Y) (Richardson and Gilks, 1993; Dellaportas and Stephens, 1995). These methods are flexible in accommodating various types of response variables and measurement errors that arise in real applications, but they are usually computationally expensive. If the distribution of U is unknown but there exists an observed variable T that is dependent of X and independent of U, the instrumental variable approach can be employed for measurement error models (Carroll and Stefanski, 1994). We will review all these existing methods in Section 2.

Recently, the success of deep learning in artificial intelligence has inspired a surge of interests in exploiting neural network structures for statistical inference and learning. Fan, Ma and Zhong (2021) give a nice review of common neural network models and training techniques from a statistical perspective. It is widely observed, empirically and theoretically, that neural networks have the ability of representing a variety of function classes (Mhaskar, 1996; Maiorov and Meir, 2000; Rolnick and Tegmark, 2018) and that the neural-network-based methods tend to be resistant to overfitting and have a good generalization power (Bauer and Kohler, 2019; Schmidt-Hieber et al., 2020). These nice features motivate us to consider a new direction of fitting measurement error models — by exploiting neural network structures.

However, the neural network models and methods reviewed in Fan, Ma and Zhong (2021) are for the nonparametric regression with error-free covariates. When there are measurement errors, the neural network modeling and training become very different. The purpose of this paper is three-fold. First, we investigate and discuss different ways to exploit neural network structures in the nonparametric regression with measurement errors. We also propose an approach that integrates the fully-connected feed-forward neural networks (FNNs) and the normalizing flow (Tabak and Turner, 2013) into the Bayesian framework of fitting measurement error models. This is presented in Section 3. Second, we demonstrate how to apply the variatonal autoencoder (Kingma and Welling, 2014) to train the above neural network models. In the usual nonparametric regression setting, the training is via stochastic gradient descent algorithms. However, with the presence of measurement errors, we face a hierarchical model with latent variables, and we have to use variational inference techniques. This is presented in Section 4, where we review the recent developments in variational inference for neural networks, especially the importance weighted autoencoder (Burda, Grosse and Salakhutdinov, 2016) and the doubly reparametrized gradient estimator (Tucker et al., 2018), and use them to design training algorithms. Last, we compare the neural network approach with existing

nonparametric/semiparametric methods on simulated data and two real data sets. These are presented in Sections 5-6.

How to use neural network techniques in measurement error models is not straightforward. We propose a Bayesian framework for fitting measurement error models, the Neural Network method for Measurement Error models (NNME), where, in contrast to standard Bayesian methods, we replace (i) the class of f, (ii) the model for the prior distribution of X, and (iii) the inference procedure, by their respective neural network counterparts. Components (ii)-(iii) are not needed for developing neural network approaches to the nonparametric regression with error-free covariates, hence not covered in the review of Fan, Ma and Zhong (2021). We point out that (ii) is connected to using normalizing flows to estimate nonparametric densities and (iii) is connected to using variational autoencoder to train deep generative models. There is a nascent literature on these directions in deep learning, but they are not yet familiar to the statistics community. The description of our NNME model and its training technique serves to review and elaborate these ideas from a statistics perspective.

Our numerical exploration suggests that the neural network approach to fitting measurement error models is promising. Thanks to the representation power of neural network function classes, this approach is flexible to accommodate various classes of f without requiring prior knowledge. It is also convenient to deal with multiple covariates ($d > 1$): switching from $d = 1$ to $d > 1$, nonparametric methods often need special treatments (e.g., in the kernel/spline construction), but the neural network approach does not, as the code is identical for $d = 1$ and $d > 1$. In the case of multiple covariates, the neural network approach is also computationally fast. In contrast, even for $d = 2$, we find that the available code of several other methods either does not work or run very slowly. Tuning in the neural network approach is mainly about selecting the network architecture, such as the number of layers and the number of nodes per layer. Thanks to the resistance-to-overfitting of neural network structures, we can set these numbers properly large without fine tuning. On the other hand, neural networks need a large sample size to train, so that its advantage is seen only when n is properly large. Compared with existing methods (especially nonparametric methods), the neural network approach is more suitable if we face a complicated f to estimate but also have a large number of samples. This scenario is common in modern big data applications, making the exploitation of neural network structures in measurement error models a very promising direction.

The remaining part of this article is organized as follows: Section 2 reviews some existing methods for treating measurement error models. Section 3 describes a neural network-based Bayesian framework for estimating measurement error models. Section 4 reviews the literature of variational auto-encoders and proposes an algorithm for training the neural network model. Section 5 compares neural network approaches with classical ones for measurement error models via simulated data. Section 6 presents the application in two real datasets, and Section 7 concludes with a few remarks.

2. CLASSICAL METHODS FOR NONPARAMETRIC MEASUREMENT ERROR MODELS

Let $Y \in \mathbb{R}$ be the response, and let $W \in \mathbb{R}^d$ be the vector of error-prone covariates. We assume that

$$Y = f(X) + \epsilon, \qquad W = X + U, \tag{1}$$

where (X, U, ϵ) are independent of each other, with U being the unobserved measurement error, $\mathbb{E}[\epsilon] = 0$, and $\mathbb{E}[U] = 0$. By convention, the measurement error distribution $p_U(u)$ is assumed known (in practice, it is often estimated or determined by the prior data or knowledge) since otherwise the regression function would be unidentifiable. This assumption can be relaxed, however, if some instrumental variables uncorrelated with the measurement error are available (see Section 2.3). Given independent and identically distributed realizations of (W, Y), $\{(w_i, y_i), i = 1, 2, ..., n\}$, our goal is to estimate the regression function f.

2.1 Nonparametric methods

Such methods typically start from an existing nonparametric method for the error-free case, and aim to address the bias caused by replacing X with W in the presence of measurement errors. The deconvolution approach builds on local smoothing estimators for nonparametric regression in the error-free case. A local smoother takes the form $\hat{f}(x) = \sum_{i=1}^n \ell_i(x) y_i$. Given a kernel function $K(\cdot)$ and a bandwidth h, the Nadaraya-Watson estimator corresponds to taking $\ell_i(x) = K(\frac{x_i - x}{h}) / [\sum_{j=1}^n K(\frac{x_j - x}{h})]$. In the presence of measurement errors, $\{K(\frac{x_j - x}{h})\}_{j=1}^n$ are unobserved. Fan and Truong (1993) proposed to replace $K(\frac{x_j - x}{h})$ by $L(\frac{w_j - x}{h})$, such that $\mathbb{E}[L(\frac{w_j - x}{h}) | x_j] = K(\frac{x_j - x}{h})$. The function $L(\cdot)$ is defined by

$$L(x) = \frac{1}{2\pi} \int e^{-\sqrt{-1}tx} \frac{\phi_K(t)}{\phi_U(t/h)} dt, \qquad \begin{cases} \phi_K(t) : \text{Fourier transform of } K(\cdot), \\ \phi_U(t) : \text{characteristic function of } U. \end{cases}$$

This approach is called 'deconvolution' for it is connected to density deconvolution (Carroll and Hall, 1988; Stefanski and Carroll, 1990). Similar ideas can be developed to combine with other local smoothing estimators, such as the local polynomial estimator (Delaigle, Fan and Carroll, 2009). A nice property of the deconvolution approach is that it requires no structural assumption on $f(\cdot)$, except the smoothness. Another nice property is that it attains the minimax rate of convergence (Fan and Truong, 1993). On the other hand, its numerical performance relies on the selection of bandwidth h. How to select h in the presence of measurement errors is a challenging problem and not well studied in the literature. One notable work is Delaigle and Hall (2008), where they combined cross-validation with the simulation extrapolation idea (to be introduced below) for bandwidth selection.

The regression spline approach approximates $f(\cdot)$ by spline bases. Different from the error-free case, fitting splines with measurement errors requires either estimating the posterior distribution of X or calculating an unbiased score function. Carroll, Maca and Ruppert (1999) assumed that $f(x)$ has an expansion on the truncated power basis $\{1, x, \ldots, x^p, (x - \xi_1)_+^p, \ldots, (x - \xi_k)_+^p\}$, where $\xi_1, \xi_2, \ldots, \xi_k$ are the fixed knots. They replaced the unobserved quantities, x_i^m and $(x_i - \xi_j)_+^m$, by $\mathbb{E}(x_i^m \mid w_1, \ldots, w_n)$ and $\mathbb{E}[(x_i - \xi_j)_+^m \mid w_1, \ldots, w_n]$, respectively. To compute these conditional moments, they assumed that the distribution of X is

a mixture of normals and developed a Gibbs sampling algorithm to estimate the posterior distribution of the x_i's. Jiang, Ma and Carroll (2018) assumed that $f(x)$ has an expansion on the B-spline basis and proposed an estimating equation method. Denote by β the spline coefficients and let $S^*_\beta(W, Y, \beta)$ be the gradient of the log-likelihood of (W, Y), assuming a working density for X. This method estimates β by solving $\sum_{i=1}^n \{S^*_\beta(w_i, y_i, \beta) - g(w_i, y_i, \beta)\} = 0$, where $g(w, y, \beta)$ is a function such that $\mathbb{E}[S^*_\beta(W, Y, \beta) - g(W, Y, \beta) \mid X] = 0$. The spline approach attains a faster rate than the minimax one if the density of X has a compact support. For Gaussian measurement errors, when $f(\cdot)$ has a continuous kth derivative, the minimax rate for the mean squared error is as slow as $\log(n)^{-k}$ (Fan and Truong, 1993). In comparison, assuming a compact support for X, the spline approach can attain the standard nonparametric rate $n^{-\frac{2k}{2k+1}}$ (Hall and Qiu, 2005; Jiang, Ma and Carroll, 2018). The spline approach still requires choosing tuning parameters— knots in the spline construction. Following Eilers and Marx (1996), one can use a large number of equally spaced knots and add a penalty on the difference of coefficients of adjacent splines. It reduces to choosing the penalization parameter λ, which can be done by using doubling smoothing (Carroll, Maca and Ruppert, 1999).

The simulation extrapolation (SIMEX) approach builds upon an arbitrary nonparametric regression method for the error-free case. For every $\lambda > 0$, it constructs $w_i(\lambda) = w_i + \sigma_0\sqrt{\lambda}\delta_i$, $1 \le i \le n$, where $\delta_i \sim \mathcal{N}(0, 1)$ independent of data, and σ_0^2 is the variance of the measurement error U in model (1). Let $\hat{f}(x; \lambda)$ be the estimated nonparametric regression function with $\{w_i(\lambda)\}_{1\le i\le n}$ being used as covariates. In a linear measurement error model, $\hat{f}(x; \lambda)$ gives a consistent estimator of the true $f(x)$ when being extrapolated to $\lambda = -1$ (Cook and Stefanski, 1994). This idea was generalized by Carroll, Maca and Ruppert (1999) to nonparametric measurement error models: It first estimates $\hat{f}(x; \lambda)$ for a few values of $\lambda \ge 0$ using an available nonparametric regression method, fits a quadratic curve of λ using the values of $\hat{f}(x; \lambda)$, and then extrapolates the fitting to $\lambda = -1$. SIMEX provides a super convenient way to extend an arbitrary nonparametric regression method from the error-free case to the error-in-variable case. But its consistency needs strong assumptions such as the variance of U converging to zero as $n \to \infty$ (Cook and Stefanski, 1994). SIMEX needs to select tuning parameters in the original nonparametric regression method. Staudenmayer and Ruppert (2004) introduced the empirical bias bandwidth selector (EBBS), which estimates the bias of SIMEX for each fixed h and selects h that minimizes the mean-squared error.

Nonparametric methods were mostly studied for $d = 1$. When $d > 1$, they have natural extensions, but the computational cost and the selection of bandwidth or knots can be much more challenging. One of our motivations of introducing the neural network approach in Section 3 is to provide a computationally cheaper solution for $d > 1$.

2.2 Bayesian methods

Bayesian methods are also widely used for estimating measurement error models. These methods typically assume that the regression function f, the prior distribution of X, and the distribution of ϵ are all from some restrictive families. The marginal likelihood of (W, Y) is

$$L(\alpha, \beta, \gamma; W, Y) = \int p_U(x - W) \cdot p_X(x; \beta) \cdot p_\epsilon\big(Y - f(x; \alpha); \gamma\big) dx,$$

where $f(\cdot;\alpha)$ is the regression function, $p_X(\cdot;\beta)$ and $p_\epsilon(\cdot;\gamma)$ are parametric densities, and $p_U(\cdot)$ is assumed known. The parameter estimation and inference are usually conducted by Markov chain Monte Carlo (MCMC) (Crainiceanu, Ruppert and Wand, 2005) or the Monte Carlo EM algorithm (Ganguli, Staudenmayer and Wand, 2005).

Different Bayesian methods differ in their choices of the model families for f, p_X, p_ϵ, and p_U. The choices of p_ϵ and p_U are typically motivated by the application in consideration. For $f(\cdot)$, earlier works used linear functions (Richardson and Gilks, 1993) or parametric nonlinear functions (polynomial regression, segmented regression or other forms for specific scientific applications) (Dellaportas and Stephens, 1995). High-degree polynomials can be unstable especially at the boundaries. Later works used better nonparametric function approximations such as penalized splines (Berry, Carroll and Ruppert, 2002; Ruppert, Wand and Carroll, 2003) to reduce the estimation instability. For the model of $p_X(\cdot)$, some assumes it to be a multivariate normal distribution with unknown parameters (Richardson and Gilks, 1993). However, this simple prior can be too restrictive in many real applications, resulting in non-negligible biases for estimating the regression function. Others employed more flexible models such as a k-component Gaussian mixture (Carroll, Roeder and Wasserman, 1999), a Dirichlet process Gaussian mixture (Müller and Roeder, 1997), or a discrete distribution (Gustafson, Le and Vallée, 2002).

A main advantage of Bayesian methods is that it is flexible in accommodating both continuous and discrete response variables and various kinds of measurement errors, such as discrete errors for categorical covariates (Gustafson, 2003) (e.g., in case-control studies), multiplicative errors (where $\log(W) = \log(X) + U$), Berkson errors (Dellaportas and Stephens, 1995), or a mixture of different types of errors from several measuring instruments (Mallick, Hoffman and Carroll, 2002). Other advantages include the convenience of incorporating multiple sources of information as priors to improve parameter estimation and the possibility of combining different experimental designs in a single framework (e.g., repeated measures, the existence of a validation group where error-free covariates are observed, or the existence of ancillary data from a separate source) (Gilks and Richardson, 1992).

On the other hand, Bayesian methods require the user to specify every component of the full model. It is sometimes unclear how to decide their forms when limited information is available. Sensitivity analysis is often needed to check the robustness of model assumptions. Bayesian methods are also computationally much more expensive than nonparametric methods, as they often require using expensive MCMC algorithms for posterior inference. The neural network framework described in Section 3 can be viewed as a new form of the Bayesian nonparametric method, in which function f and the prior of X are modeled by neural networks. Instead of using MCMC, we employ variational inference and gradient descent to achieve fast computation in complex models.

2.3 Instrumental variable methods

If the data contains observations on an instrumental variable T for the covariates, instrumental variable methods are commonly used for measurement error models when the measurement error variance is unknown (Carroll et al., 2006). Although T does not need to be

a replicate measurement, T has to depend on X, and is uncorrelated (or independent in the nonlinear setting) with the measurement error U and the response error ϵ.

In the linear regression setting with classical measurement errors, the instrumental variable estimate can be constructed as $\hat{\beta}^{IV} = (\hat{\Sigma}_{TW}^T \hat{\Sigma}_{TW})^{-1} \hat{\Sigma}_{TW}^T \hat{\Sigma}_{TY}$, where $\hat{\Sigma}_{TY}$ and $\hat{\Sigma}_{TW}$ are the sample covariance matrices between the instrumental variables T and the response variable Y and the covariates W, respectively. The matrix inverse can be replaced by the generalized inverse if we have more instrumental variables than the number of covariates.

The instrumental variable method can be extended to generalized linear models in which the mean and the variance of the response depend on a linear combination of the covariates. Regression calibration approximations are commonly used given that T and X have an approximately linear relationship (Carroll and Stefanski, 1994). Assuming that $\mathbb{E}(Y|X) = f(\beta X)$ and $E(Y|T) \approx f(\beta \mathbb{E}(X|T)) \approx f(\beta\gamma T)$, regression calibration methods first run a linear regression between W and T to obtain $\hat{\gamma}$, then perform a generalized linear regression of Y on $\hat{\gamma}T$ to get $\hat{\beta}^{IV}$. When the relationship between T and X is nonlinear, Buz as (1997) developed an adjusted score method for certain generalized linear models with scalar X, and Schennach (2007) developed hybrid classical and regression calibration approaches. Carroll et al. (2004) used instrumental variables to estimate the variance of the measurement error and combined it with a nonparametric regression method for measurement error models with known variance. Bayesian methods can be used to study non-linear or nonparametric regression when instrumental variables are available (Carroll et al., 2004).

However, those requirements for a variable T to be an instrumental variable are difficult to validate in practice and limit the applicability of instrumental variable methods. Also, the estimates tend to have a large variation if the dependence between T and X is weak and could be biased if T is correlated with any of the errors.

3. NEURAL NETWORK MODELING FOR MEASUREMENT ERRORS

Using neural networks for nonparametric estimation has received much recent interest (see Fan, Ma and Zhong (2021) for a nice review). In these works, a nonparametric function f is approximated by a feed-forward neural network (FNN). As depicted in Figure 1, an L-layer FNN is a composition of simple nonlinear functions:

$$f(x) = h^{(L)} \circ h^{(L-1)} \circ \ldots \circ h^{(1)} \circ h^{(0)}(x), \tag{2}$$

where $h^{(\ell)}$ is a mapping from $\mathbb{R}^{d_\ell}$ to $\mathbb{R}^{d_{\ell+1}}$, with $d_0 = d$ and $d_{L+1} = 1$. Let $\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$ be an almost-everywhere differentiable function, called the *activation function* in the literature. For any vector $u \in \mathbb{R}^d$, we denote by $\boldsymbol{\sigma}(u) \in \mathbb{R}^d$ the vector of applying $\sigma(\cdot)$ entry-wise. Each $h^{(\ell)}$ has the form

$$h^{(\ell)}(u) = \boldsymbol{\sigma}\big(A^{(\ell)}u + b^{(\ell)}\big), \qquad \text{where} \quad A^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell},\ b^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}, \tag{3}$$

except that the last layer $h^{(L)}$ is a linear mapping. The rectifier linear unit (ReLU) function, $\sigma(x) = \max\{x, 0\}$, is a popular activation function. Other choices include the sigmoid function and the tanh function. Let $\theta = \{A^{(\ell)}, b^{(\ell)}\}_{0 \le \ell \le L}$ denote the parameters of an FNN. Following the statistical tradition, we write $f = f_\theta$. The nonparametric estimation of f now

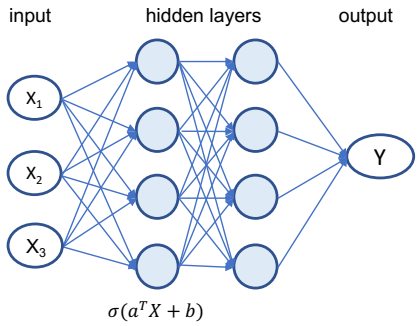


Fig 1: An example of an FNN with $L = 2$ and $d = 3$.

reduces to the estimation of θ of the FNN, which is often achieved by minimizing an empirical loss function, such as the residual sum of squares plus a regularization term.

It has been shown that the multi-layer FNN class defined in (2)-(3) can provide satisfactory approximations to a wide range of continuous functions (Lin, Tegmark and Rolnick, 2017; Rolnick and Tegmark, 2018; Bauer and Kohler, 2019; Schmidt-Hieber et al., 2020). At the same time, although deep FNN's have a large number of parameters to estimate, FNN-based algorithms typically do not suffer much from over-fitting. This is hard to explain by classical learning theory, and many recent theoretical studies have been devoted to understanding this phenomenon. One explanation is the norm regularization. In the training of neural networks, it is common to add penalties on the norms $\|A^{(\ell)}\|_F$. Assuming that these norms are properly bounded, the Rademacher complexity of the FNN function class can be nicely controlled (Golowich, Rakhlin and Shamir, 2018). Another explanation hinges on properties of the stochastic gradient descent (SGD) algorithm for training neural networks. Instead of considering the global minimum of the empirical loss, such theory focuses on local minimums found by SGD and shows that these solutions are properly regularized. Related works include the mean field approximation (Mei, Montanari and Nguyen, 2018), stability of SGD (Hardt, Recht and Singer, 2016), and implicit regularization of SGD (Gunasekar et al., 2018).

The representation power of FNN and the resistance to over-fitting of neural network training together motive us to use them for measurement error models. Specifically, we will model the regression function f in a measurement error model by an FNN as defined in (2)-(3). However, unlike the nonparametric regression, the measurement error model is a hierarchical model with latent variables. We face extra challenges when using neural networks. First, the prior distribution $p_X(\cdot)$ is unknown. We wish to also use a neural network to approximate $p_X(\cdot)$, but it cannot be a regular FNN since $p_X(\cdot)$ needs to be a valid density. Second, following a conventional Bayesian framework (see Section 2.2), we should use the marginal log-likelihood as the empirical loss. Then, the standard SGD is not directly applicable because the objective function involves an integral with respect to the density of X. In Section 3.1 below, we resolve the first challenge by approximating $p_X(\cdot)$ with a normalizing flow, and resolve the second challenge by employing the variational auto-encoder for inference.

3.1 A neural network structure for measurement error models

We assume that ϵ is normally distributed in this section, although the normal family can be easily replaced by any given parametric family. Let $f = f_\theta$ be parametrized by FNN as in (2). We re-write the measurement error model as

$$\begin{aligned} &Y = f_\theta(X) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2) \text{ and } X \sim p_X(x), \\ &W = X + U, \qquad U \sim p_U(u). \end{aligned} \tag{4}$$

We use a normalizing flow (Tabak and Turner, 2013; Papamakarios, Pavlakou and Murray, 2017; Rezende and Mohamed, 2015) to represent the prior distribution p_X. A normalizing flow is a sequence of transformations $g_1, g_2, \ldots, g_m$, where each $g_j : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible mapping parametrized by a few neural network layers and m is a tuning integer (adequate to have $m \geq 3$ (Dinh, Krueger and Bengio, 2015)). Let $V \in \mathbb{R}^d$ be a random vector whose density has a simple analytic form (e.g., $V \sim \mathcal{N}(0, I_d)$). We assume that

$$X = G_\gamma^{-1}(V), \qquad \text{where} \quad G_\gamma = g_m \circ g_{m-1} \circ \ldots \circ g_1,$$

with γ denoting the parameters employed in the mappings. Let $J(x)$ denote the Jacobian of the mapping G_γ. Then, the implied distribution for X is $p_X(x) = p_V(G_\gamma(x)) \cdot |\det J(x)|$. Intuitively, this strategy obtains the target prior $p_X(\cdot)$ by transforming a simple analytic density (e.g., normal) via a "flow" of invertible mappings. A specific simple flow called the Non-linear Independent Components Estimation (NICE) (Dinh, Krueger and Bengio, 2015) is employed in our framework. NICE parametrizes each invertible mapping $v = g_j(x)$ as follows: $v_{I_{j1}} = x_{I_{j1}}$ and $v_{I_{j2}} = x_{I_{j2}} + h(x_{I_{j1}})$, where $\{I_{j1}, I_{j2}\}$ is a partition of $\{1, \ldots, d\}$ and h is a neural network with $|I_{j1}|$ input units and $|I_{j1}|$ output units. Such a mapping has a trivial inverse and the identity Jacobian regardless of the choice of h. Different g_j corresponds to different partitioning $\{I_{j1}, I_{j2}\}$. It thus follows that

$$p_X(x) = p_V(G_\gamma(x)), \tag{5}$$

where γ denotes the parameters of NICE. We call the fitting based on (2), (4), and (5) the neural network method for measurement error models (NNME).

Under the NNME, the joint density of (W, Y, X) can be expressed explicitly:

$$p(w, y, x; \theta, \gamma, \sigma^2) = p_V(G_\gamma(x)) \times p_U(w - x) \times p_\epsilon(y - f_\theta(x); \sigma^2). \tag{6}$$

In a similar spirit as empirical Bayes methods, we estimate the parameters $(\theta, \gamma, \sigma^2)$ by maximizing the marginal log-likelihood of the observed variables:

$$L(\theta, \gamma, \sigma^2) \equiv \log \int p(w, y, x; \theta, \gamma, \sigma^2) dx. \tag{7}$$

The likelihood is an integral over the latent variable x and cannot be evaluated analytically. We follow the idea of variational auto-encoder (VAE) (Kingma and Welling, 2014; Rezende, Mohamed and Wierstra, 2014) to pair this model with another neural network for conducting variational inference, which can be viewed as a generalization (or relaxation) of the EM algorithm (Dempster, Laird and Rubin, 1977). More specifically, we consider a "proposal" distribution for the latent variables X given (W, Y), that is $X \mid (W, Y) \sim \mathcal{N}(\mu_\phi(W, Y), \Sigma_\phi(W, Y))$,

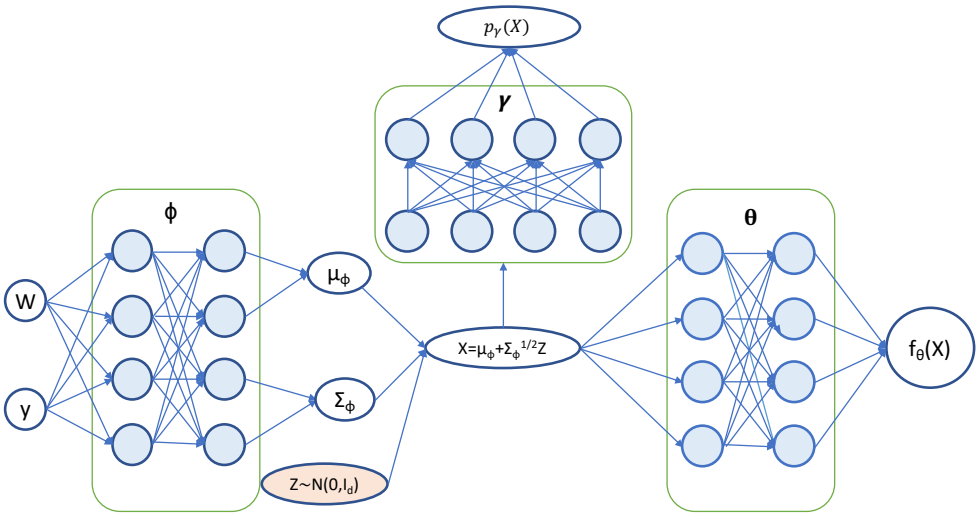


Fig 2: A neural network structure for NNME. The input is w and y, and the output is the estimated regression function $f_\theta(x)$. The left green block is an encoder, which consists of several fully connected layers with ReLU activation functions and the last layer with a linear function; the output of the encoder are parameters for the proposal distribution. The right green block is a decoder, which has the same network structure as the encoder; the input are random samples of x, and the output are estimated values of $f_\theta(x)$. The top green block is another decoder, which consists of a few coupling layers of a normalizing flow; the input are random samples of x, and the output is the estimated prior density of X.

where μ_ϕ and Σ_ϕ are represented by a neural network with parameter ϕ. Let $q_\phi(x \mid w, y)$ denote this distribution. By Jensen's inequality, the log-likelihood in (7) has a lower bound:

$$Q_{\text{VAE}}(\theta, \gamma, \sigma^2, \phi) = \int \log\left(\frac{p(w, y, x; \theta, \gamma)}{q_\phi(x|w, y)}\right) q_\phi(x|w, y)dx, \tag{8}$$

which is often called the *evidence lower bound* (ELBO) in the literature (Blei, Kucukelbir and McAuliffe, 2017). The key idea of variational inference is to maximize this ELBO simultaneously over both the model parameters $(\theta, \gamma, \sigma^2)$ and the parameters ϕ of the proposal distribution. Note that distribution $q_\phi(x \mid w, y)$ is not part of our model since x's conditional distribution is already implied by the joint distribution in (6). The role of $q_\phi(x \mid w, y)$ is purely computational, being analogous to the trial distribution in an importance sampling approximation. In fact, it is useful to realize that ELBO (8) is derived by applying Jensen's inequality to an importance sampling expression of (7). With $q_\phi(x \mid w, y)$ and its associated ELBO, the optimization task is easier to carry out. See Section 4 for more detailed discussions.

The neural network structure is shown in Figure 2. The FNN parameterized by ϕ is called an encoder or an inference network. It takes data (W, Y) and outputs the parameters (μ_ϕ, Σ_ϕ) of the proposal distribution for generating latent variable X. The FNN parameterized by θ and the normalizing flow parameterized by γ are both called decoders or generative networks. They take the random samples of X from the proposal distribution and outputs the estimated $f_\theta(x)$ and $p_X(x)$ at those random samples.

3.2 Other neural network models

There are other ways to incorporate neural networks in a measurement error model. The first alternative is to ignore the measurement error and treat the problem as a standard non-

parametric regression with a neural network representing f (we abbreviate this approach as NN). When the variance of U is properly small, NN may have a reasonably good performance. The second approach is to use a parametric density for $p_X(\cdot)$ while keeping other components of NNME unchanged. Consequently, in Figure 2, the decoder (generative network) for $p_\gamma(x)$ becomes an explicit function. This is a simplified version of NNME (we still call it NNME). It works well when true prior distribution can be well approximated by a parametric family. The third approach is to use a similar encoder-decoder structure as in NNME but to maximize the joint likelihood of (X, W, Y) instead of the marginal likelihood (7), abbreviated as MJL. Let $\hat{x}_i(\phi; w_i, y_i) = h_\phi(w_i, y_i)$, $1 \le i \le n$, where h_ϕ is a function parameterized by the encoder. The joint likelihood is $\widetilde{L}(\theta, \phi, \sigma^2 \mid w, y) = p_U\big(w - h_\phi(w, y)\big) \cdot p_\epsilon\big(y - f_\theta(h_\phi(w, y)); \sigma^2\big)$. The parameters are estimated by alternating between computing x_i from the encoder and updating $(\hat{\theta}, \hat{\phi}, \hat{\sigma}^2)$ by maximizing $\widetilde{L}$. MJL is easier to implement than NNME, as the objective has no integral, but it may lose efficiency for not using the prior information.

A numerical comparison. We compare different neural-network-based approaches, NN, MJL, and several versions of NNME, in a setting with $d = 2$ and f generated from a Gaussian process (see Appendix C for details). Let

$$X \sim 0.7 \cdot \mathcal{N}\left(\begin{bmatrix} -0.4 \\ 0.2 \end{bmatrix}, \begin{bmatrix} 0.2^2 & 0 \\ 0 & 0.3^2 \end{bmatrix}\right) + 0.3 \cdot \mathcal{N}\left(\begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}, \begin{bmatrix} 0.3^2 & 0 \\ 0 & 0.2^2 \end{bmatrix}\right) \tag{9}$$

and $U \sim \mathcal{N}(0, \sigma_0^2 I_2)$. NNME-true denotes the benchmark resulting from using the true data-generating prior $p_X(x)$. NNME-GM2, NNME-GM4, and NNME-tdist approximate p_X by a 2-component Gaussian mixture, 4-component Gaussian mixture, and t-distribution, respectively. NNME-NICE is as in Section 3.1. For all methods, we use 9 hidden layers for encoders and 5 hidden layers for decoders, with 32 nodes per layer (see Appendix C for varying the numbers of layers). The performance is measured by the integrated squared error (ISE) $\int_{[-1,0.2]\times[-1,0.5]\cup[-0.5,1]\times[-0.2,1]} [f_{\hat{\theta}}(x) - f_\theta(x)]^2 dx$; the integral is restricted to a region where very few training x_i's fall outside this region.

The results are shown in Figure 3. When the measurement error is small ($\sigma_0 = 0.05$, top panels) and the sample size is large, NN has a reasonable performance; but for all the other cases, NN is considerably worse than NNME. This suggests that ignoring measurement errors is unsatisfactory. MJL and NNME both account for measurement errors, but NNME has a much better performance, especially when n is large (see Appendix B for more explanations). When the measurement error is small, the performance is insensitive to the chosen model of X. In this case, the posterior of the x_i's are mainly determined by the w_i's. When the measurement error is large ($\sigma_0 = 0.2$, bottom panels) using the correct parametric model (i.e., GM2) yields a similar performance as using the true model. For the two misspecified parametric models, GM4 performs as well as (sometimes even better than) the true model, but the (unimodal) t-distribution performs unsatisfactorily. The NICE model performs near-optimally in all settings, demonstrating the robustness of the normalizing flow approach. In summary, the experiment suggests to use either NICE or a Gaussian mixture with a reasonably large number of components for modeling X.

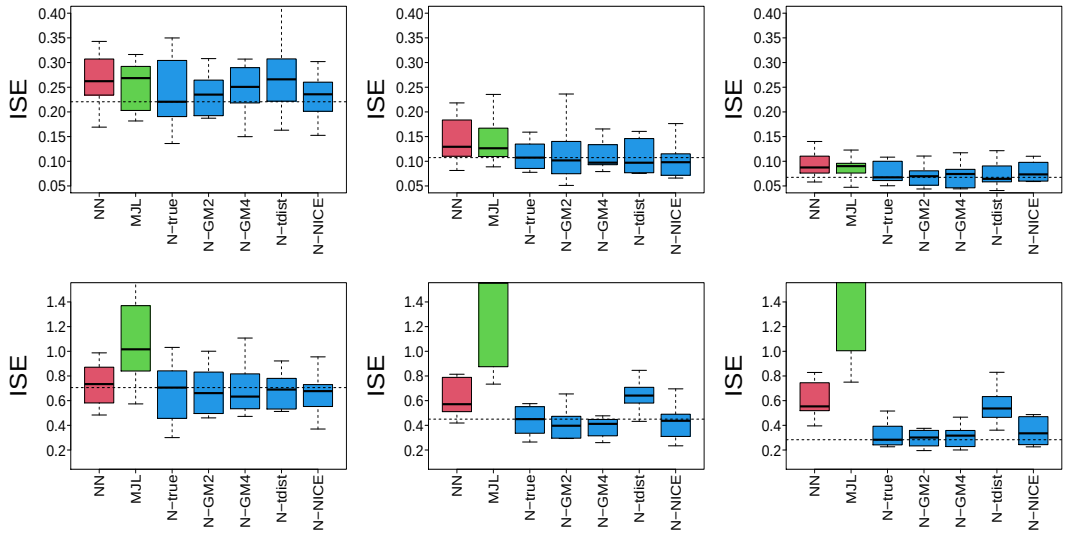


Fig 3: Comparison of different neural network models. X-axis: the neural network structure (see the main text); Y-axis: box plots of ISE based on 10 repetitions. The standard deviation of measurement errors is set as 0.05 (top panel) and 0.2 (bottom patel, respectively; the sample size is (from left to right) 1000, 4000, and 8000, respectively. "N-" stands for "NNME-". The dash line represents the median ISE resulting from NNME-true.

4. TRAINING AND INFERENCE FOR THE NEURAL NETWORK MODEL

The training of NNME is more complicated than a typical Bayesian approach because the standard MCMC is hard to implement. We have to use the variational auto-encoder (VAE). In this section, we review some recent developments of VAE and propose an algorithm for training NNME. For simplicity, we tentatively drop σ^2 from the optimization (and thus from all the expressions of densities), but we will estimate σ^2 along with training in Section 4.4. We also write $p(w, y, x; \theta, \gamma, \sigma^2)$ in (7) as $p_{\theta,\gamma}(w, y, x)$ for short.

4.1 Variational auto-encoder

ELBO in (8) is connected to the marginal likelihood (7) by

$$Q_{\text{VAE}}(\theta, \gamma, \phi) = L\big(\theta, \gamma) - \text{KL}(q_\phi(\cdot \mid w, y) \| p(\cdot \mid w, y)\big), \tag{10}$$

where $\text{KL}(q\|p)$ denotes the Kullback-Leibler divergence between distributions q and p. A key insight of variational inference (Jordan et al., 1999) is that, by optimizing ϕ jointly with other parameters, we force the proposal distribution q_ϕ to be close to the true conditional distribution of X, which is the one used in the standard EM algorithm.

In a stochastic gradient ascent algorithm to maximize Q_{VAE}, one first approximates Q_{VAE} by its Monte Carlo estimate and then computes the gradient of $\widehat{Q}_{\text{VAE}}(\theta, \gamma, \phi)$. Such a gradient estimate is called the REINFORCE gradient estimator (Williams, 1992). Training with the REINFORCE update rule is usually slow, because the effect of ϕ in the Monte Carlo approximation error is not taken into account in the gradient. Kingma and Welling (2014) introduced a re-parametrization trick by writing the samples from the proposal distribution as a deterministic function of some auxiliary variable z. Namely, $x = x(\phi, z)$, where the distribution of z is fixed (e.g., $\mathcal{N}(0, I_d)$) and $x(\phi, \cdot)$ is a non-stochastic function. Then, $q_\phi(x \mid w, y)$ becomes

$q_\phi(x(\phi,z) \mid w,y)$, and (8) is re-written as $Q_{\text{VAE}}(\theta,\gamma,\phi) = \mathbb{E}_{z\sim p_Z(\cdot)}\big[\log\big(\frac{p_{\theta,\gamma}(w,y,x(\phi,z))}{q_\phi(x(\phi,z)|w,y)}\big)\big]$. Now, if we approximate Q_{VAE} by its Monte Carlo estimate and then take the gradient, such a gradient estimate is called the re-parametrized gradient estimator. Compared to the REINFORCE gradient, this gradient estimator has a smaller variance and speed up the training.

In the setup of NNME, we let the proposal distribution be a multivariate normal distribution, i.e., $q_\phi(x \mid w,y)$ is the density function of $\mathcal{N}\big(\mu_\phi(w,y),\Sigma_\phi(w,y)\big)$. Using the re-parametrization trick, we write

$$x(\phi,z) = \mu_\phi(w,y) + [\Sigma_\phi(w,y)]^{1/2} z, \qquad \text{where} \quad z \sim \mathcal{N}(0,I_d). \tag{11}$$

The re-parametrized VAE objective is

$$Q_{\text{VAE}}(\theta,\gamma,\phi) = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}\left[\log\left(\frac{p_{\theta,\gamma}(w,y,x(\phi,z))}{q_\phi(x(\phi,z)|w,y)}\right)\right]. \tag{12}$$

4.2 Importance weighted auto-encoder

The disadvantage of VAE is that it relies on a strong assumption that the true conditional distribution of X can be approximated well by the simple proposal distribution (in our case, a multivariate normal distribution). Burda, Grosse and Salakhutdinov (2016) proposed an improvement of VAE, called the importance weighted auto-encoder (IWAE). It aims to resolve this issue by borrowing ideas from importance sampling.

Instead of having one latent x drawn as in (11), we consider K latent samples:

$$x(\phi,z_k) = \mu_\phi(w,y) + [\Sigma_\phi(w,y)]^{1/2} z_k, \qquad \text{where} \quad z_1,z_2,\ldots,z_K \overset{iid}{\sim} \mathcal{N}(0,I_d). \tag{13}$$

Here, $z_1,z_2,\ldots,z_K$ are called importance samples or particles. The IWAE objective is

$$Q_{\text{IWAE}}(\theta,\gamma,\phi) \equiv \mathbb{E}_{z_{1:K}\sim\mathcal{N}(0,I_d)}\left[\log\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p_{\theta,\gamma}\big(w,y,x(\phi,z_k)\big)}{q_\phi\big(x(\phi,z_k)|w,y\big)}\right)\right]. \tag{14}$$

By Jensen's inequality,

$$\begin{aligned} Q_{\text{IWAE}}(\theta,\gamma,\phi) &\le \log\left(\mathbb{E}_{z_{1:K}\sim\mathcal{N}(0,I_d)}\left[\frac{1}{K}\sum_{k=1}^{K}\frac{p_{\theta,\gamma}\big(w,y,x(\phi,z_k)\big)}{q_\phi\big(x(\phi,z_k)|w,y\big)}\right]\right) \\ &= \log\int p_{\theta,\gamma}(w,y,x)dx = L(\theta,\gamma). \end{aligned}$$

Hence, $Q_{\text{IWAE}}(\theta,\gamma,\phi)$ is also an ELBO of the marginal log-likelihood. IWAE trains the neural networks by maximizing this objective.

The advantage of IWAE over VAE is seen from the tightness of the ELBO. Burda, Grosse and Salakhutdinov (2016) showed that, for any fixed (θ,γ,ϕ), under mild conditions, $Q_{\text{IWAE}}(\theta,\gamma,\phi)$ increases monotonically with K and converges to $L(\theta,\gamma)$ as $K\to\infty$. In (12), Q_{VAE} corresponds to $K=1$. Hence, IWAE provides a tighter lower bound of the marginal log-likelihood. A tighter ELBO generally makes the training of the generative network (or the estimation of (θ,γ)) more efficient.

The advantage of IWAE also lies in the increase of expressiveness of the inference network (or the proposal distribution). As argued in Burda, Grosse and Salakhutdinov (2016), the VAE optimizes the ELBO in (8), which effectively favors parameter values that make the posterior distribution of X as similar to the proposal (i.e., the multivariate normal) as possible, which

reduces the expressive power of the model. An inference network that places only a small fraction of its samples in the region of high posterior probability may still be sufficient for performing accurate inference. In the gradient of $\nabla_\phi Q_{\text{IWAE}}$, the contribution of each particle z_k is re-weighted by the likelihoods, giving more flexibility to train a generative network so that the resulting posterior distribution of X is not forced to fit the restrictive proposal distribution used in VAE. This was further made rigorous by Cremer, Morris and Duvenaud (2017), where they showed that IWAE can be equivalently viewed as a standard VAE using a much broader class of distributions to approximate the posterior.

The IWAE objective has a similar decomposition as in (10). Le et al. (2018) showed that

$$Q_{\text{IWAE}}(\theta,\gamma,\phi) = L(\theta,\gamma) - \text{KL}\left[Q_{\text{IS}}(\cdot \mid w,y) \| P_{\text{IS}}(\cdot \mid w,y)\right],$$

where $Q_{\text{IS}}(x_{1:K}|w,y)$ is the joint density of $x_1,\ldots,x_K$ given (w,y) under the proposal distribution and $P_{\text{IS}}(x_{1:K} \mid w,y) = \frac{1}{K}\sum_{k=1}^K \left[p(x_k \mid w,y)\prod_{\ell\neq k} q_\phi(x_\ell \mid w,y)\right]$. They also showed that $Q_{\text{IS}} = P_{\text{IS}}$ if and only if $q_\phi(x \mid w,y) = p(x \mid w,y)$ for all x. Therefore, by optimizing over ϕ jointly with other parameters, IWAE also encourages the proposal distribution to be close to the true posterior.

4.3 Doubly reparametrized gradient estimators

From now on, we write $Q(\theta,\gamma,\phi) = Q_{\text{IWAE}}(\theta,\gamma,\phi)$ for short. The IWAE objective is often maximized via a gradient ascent algorithm. The gradient can be written as

$$\nabla Q(\theta,\gamma,\phi) = \mathbb{E}_{z_{1:K}\sim\mathcal{N}(0,I_d)}\left[\left(\frac{\beta_k}{\sum_{\ell=1}^K \beta_\ell}\right)\nabla \log\left(\frac{p_{\theta,\gamma}(w,y,x(\phi,z_k))}{q_\phi(x(\phi,z_k)|w,y)}\right)\right], \tag{15}$$

where $\beta_k = p_{\theta,\gamma}(w,y,x(\phi,z_k))/q_\phi(x(\phi,z_k)|w,y)$, for $1\le k\le K$. By (15), we write $\nabla Q = \mathbb{E}_{z_{1:K}}[h(z_1,z_2,\ldots,z_K)]$, for a function $h(z_{1:K})$. The L-sample Monte Carlo estimate is

$$\widehat{\nabla Q} = \frac{1}{L}\sum_{l=1}^L h\left(z_1^{(\ell)}, z_2^{(\ell)},\ldots,z_K^{(\ell)}\right), \tag{16}$$

where $\{z_{1:K}^{(\ell)}\}_{\ell=1}^L$ are L independent copies of $z_{1:K}$. Since the re-parametrization trick (13) has already been employed, the $\widehat{\nabla Q}$ based on (15) is the re-parametrized gradient estimator.

When K is large in IWAE, this gradient estimator is unsatisfactory for updating ϕ (Rainforth et al., 2018), as its variance is too large. Tucker et al. (2018) proposed an alternative form of the gradient by applying the reparametrization trick twice, called the doubly reparametrized gradient (DReG), which results in a smaller Monte Carlo variance.

To see the idea, we first re-write (15) as follows: Let $x_k = x(\phi,z_k)$ and β_k be the same as above. Note that $q_\phi(x_k \mid w,y)$ and β_k are functions of (ϕ,x_k). We use $\frac{d}{d\phi}$ to denote the full derivative and use $\frac{\partial}{\partial\phi}$ to represent the derivative without treating x_k as a function of ϕ. By the chain rule,

$$\nabla_\phi Q(\theta,\gamma,\phi) = \mathbb{E}_{z_{1:K}}\left[\sum_{k=1}^K\left(\frac{\beta_k}{\sum_{\ell=1}^K\beta_\ell}\right)\left(-\frac{\partial}{\partial\phi}\log\left(q_\phi(x_k)\right) + \frac{\partial\log(\beta_k)}{\partial x_k}\frac{dx(\phi,z_k)}{d\phi}\right)\right].$$

The term $\frac{\partial}{\partial\phi}\log(q_\phi(x_k))$ leads to a large variance in the Monte Carlo estimate (Roeder, Wu and Duvenaud, 2017). Noting that, for $k \in [1:K]$,

(17)
$$\mathbb{E}_{z_{1:K}}\left[\left(-\frac{\beta_k}{\sum_{\ell=1}^K \beta_\ell}\right)\frac{\partial}{\partial\phi}\log\big(q_\phi(x_k)\big)\right] = \mathbb{E}_{z_{-k}}\left\{\mathbb{E}_{z_k}\left[\left(-\frac{\beta_k}{\sum_{\ell=1}^K \beta_\ell}\right)\frac{\partial}{\partial\phi}\log\big(q_\phi(x_k)\big)\right]\right\}$$
$$= \mathbb{E}_{z_{-k}}\left\{\mathbb{E}_{x_k}\left[\left(-\frac{\beta_k}{\sum_{\ell=1}^K \beta_\ell}\right)\frac{\partial}{\partial\phi}\log\big(q_\phi(x_k)\big)\right]\right\},$$

Tucker et al. (2018) applied the re-parametrization trick again to rewrite (17) as

(18)
$$\mathbb{E}_{z_{-k}}\left\{\mathbb{E}_{z_k}\left[\frac{\partial}{\partial x}\left(-\frac{\beta_k}{\sum_{\ell=1}^K \beta_\ell}\right)\frac{dx(\phi,z)}{d\phi}\right]\right\},$$

because of the general re-parametrization formula that, for any differentiable function g,

$$\mathbb{E}_{x\sim q_\phi(x)}\left[g(x,\phi)\frac{\partial}{\partial\phi}\log q_\phi(x)\right] = \mathbb{E}_z\left[\frac{\partial g(x,\phi)}{\partial x}\frac{dx(\phi,z)}{d\phi}\right].$$

Inserting (18) back to $\nabla_\phi Q$, Tucker et al. (2018) obtained an alternative form of the gradient:

(19)
$$\nabla_\phi Q(\theta,\gamma,\phi) = \mathbb{E}_{z_{1:K}\sim\mathcal{N}(0,I_d)}\left[\sum_{k=1}^K\left(\frac{\beta_k}{\sum_{\ell=1}^K\beta_\ell}\right)^2\cdot\frac{\partial\log(\beta_k)}{\partial x_k}\frac{dx(\phi,z_k)}{d\phi}\right].$$

We now have $\nabla_\phi Q = \mathbb{E}_{z_{1:K}}[h^*(z_{1:K})]$, for a function h^* different from the h in (15). The two are equivalent in the population form, but they lead to different Monte Carlo estimates in (16), because the function h has changed.

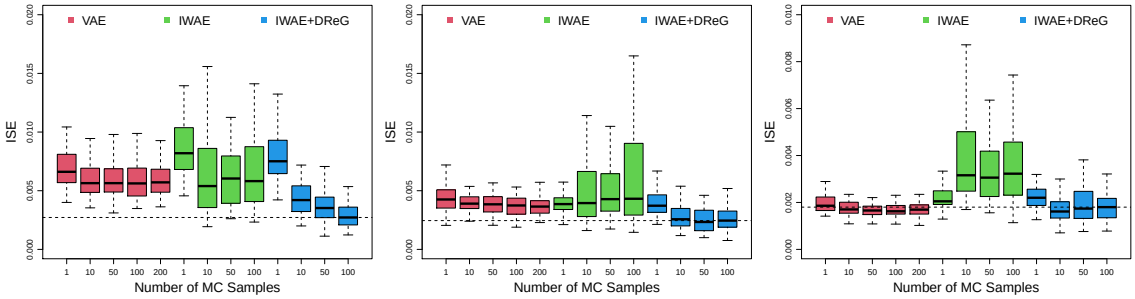
For IWAE, a larger K yields a tighter ELBO and should perform better. However, this is often not true in practice if we use (15) to approximate the gradient. Rainforth et al. (2018) shows that, when K is large, the magnitude of $\nabla_\phi Q$ becomes small and can be easily dominated by the noise in the Monte Carlo estimate. It is thus crucial to use a gradient estimator with a smaller variance. Using DReG is especially helpful in this case.

4.4 An algorithm for training NNME

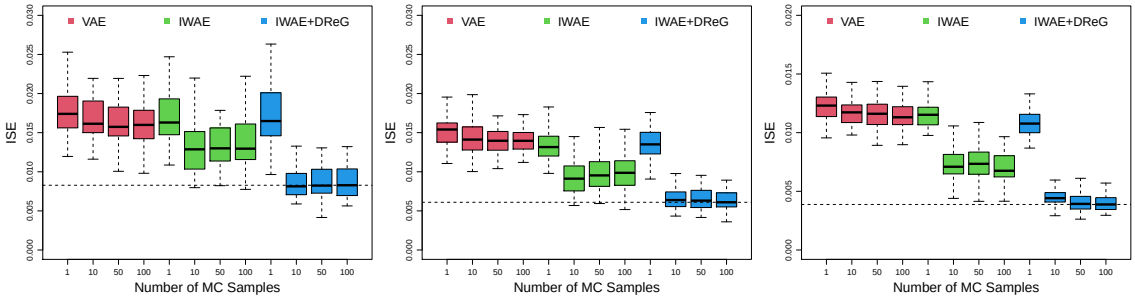
We combine the aforementioned ideas to design an algorithm to train NNME. Our objective function is Q_{IWAE} plus an $L2$-penalty term, $\lambda_0\|\theta\|^2 + \lambda_1\|\phi\|^2 + \lambda_2\|\gamma\|^2$, which helps to stabilize the performance. This objective is optimized by using *Adam* (Kingma and Ba, 2014), a variant of the stochastic gradient ascent. *Adam* has adaptive learning rates and only requires estimates of the gradients. Monte Carlo estimates (16) of the gradient are used, where h is from (15) for (θ,γ) and from (19) for ϕ. Variance σ^2 is estimated along with the training.

Training algorithm for NNME: For epoch = 1 to Max_Epoch, run the following steps to obtain $(\hat\theta,\hat\gamma,\hat\phi,\hat\sigma^2)$:

- Draw Monte Carlo samples $\{z_{ik}\}_{1\le i\le n, 1\le k\le K}$ IID from $\mathcal{N}(0,I_d)$.
- Estimate the gradients: Compute the Monte Carlo estimate $\widehat{\nabla_\theta Q}$ and $\widehat{\nabla_\gamma Q}$ for the re-parametrized gradient (15). Compute $\widehat{\nabla_\phi Q}$ for the doubly re-parametrized gradient (19). Let $\widehat{\nabla_\theta Q}_* = \widehat{\nabla_\theta Q} + 2\lambda_0\theta$, $\widehat{\nabla_\gamma Q}_* = \widehat{\nabla_\gamma Q} + 2\lambda_2\gamma$ and $\widehat{\nabla_\phi Q}_* = \widehat{\nabla_\phi Q} + 2\lambda_1\phi$ (to account for the L2 penalty).



(a) The univariate example, where $f(x) = \sin(\pi x)$. From left to right, $n = 1000, 2000, 5000$.



(b) The trivariate example, where $f(x) = (x_1 x_2 + x_3)^2$. From left to right, $n = 1000, 2000, 5000$.

Fig 4: Comparison of different training algorithms for NNME. X-axis: number of Monte Carlo samples at each iteration (i.e., L in VAE and K in IWAE and IWAE+DReG); Y-axis: box plots of ISE based on 50 repetitions. In each plot, the dash line is the median ISE for IWAE+DReG with $K = 100$.

- Plug the above gradient estimators into *Adam* to update $(\hat{\theta}, \hat{\gamma}, \hat{\phi})$.
- Let β_{ik} be the β_k in (15)-(19) evaluated using importance samples drawn for the ith observation, for $1 \le k \le K$ and $1 \le i \le n$. Update $\hat{\sigma}^2$ by

$$\hat{\sigma}^2 = \frac{1}{nK} \sum_{i=1}^{n} \sum_{k=1}^{K} (y_i - f_\theta(x_{ik}))^2 \frac{\beta_{ik}}{\sum_{\ell=1}^{K} \beta_{i\ell}}, \tag{20}$$

More details are given in Appendix A, where we describe how to initialize and how to choose the architecture of neural networks and the tuning parameters.

A numerical experiment. We compare three training algorithms: VAE, IWAE (using (15) for $\nabla_\phi Q$), and IWAE+DReG (using (19) for $\nabla_\phi Q$). We use L Monte Carlo samples for gradient estimation (see (16)). For VAE, we tried $L \in \{1, 10, 50, 100, 200\}$. For IWAE and IWAE+DReg, we fixed $L = 1$ and tried $K \in \{1, 10, 50, 100\}$. In Setting 1, $d = 1$, $f(x) = \sin(\pi x)$, $X \sim unif(-2, 2)$, and $\epsilon, U \sim N(0, 0.1^2)$. In Setting 2, $d = 3$, $f(x) = (x_1 x_2 + x_3)^2$, $X_1, X_2, X_3 \overset{iid}{\sim} unif(-1, 1)$ and $\epsilon \sim N(0, 0.1^2), U \sim \mathcal{N}(0, 0.1^2 I_3)$. We measure the performance by the integrated squared error. The results are shown in Figure 4.

There are a few noteworthy observations. For Setting 1, VAE and IWAE+DReG have similar performances when $n = 5000$, but VAE is much worse for $n \in \{1000, 2000\}$. Increasing L in VAE yields no significant improvement. This suggests that IWAE is indeed a better ELBO to optimize. IWAE and IWAE+DReG maximizes the same ELBO but uses different gradient estimates. When $K = 1$, the two have similar performances, but when K is large,

IWAE+DReG performs much better. This shows that, without the doubly reparametrized technique in gradient estimation, the advantage gained by increasing K is counterbalanced by the large variance in gradients. For Setting 2, the disadvantage of VAE is much more apparent. In VAE, the proposal distribution is a multivariate normal with a diagonal covariance matrix (i.e., a product measure on the two entries of X), which is quite far from the true conditional distribution of X. It is thus crucial to use the IWAE objective for training to relax this restrictive proposal choice in VAE.

5. COMPARISON OF CLASSICAL METHODS AND NEURAL NETWORK METHODS

We compare NNME with a few classical methods in nonparametric and semi-parametric statistics for analyzing measurement error models (see Table 1). We recall that 'deconv' is the method in Fan and Truong (1993), 'lpoy' is from Delaigle, Fan and Carroll (2009), 'BSSP' is from Jiang, Ma and Carroll (2018), and 'SIMEX' is from Carroll, Maca and Ruppert (1999). 'Pspline' and 'NN' ignore measurement errors and represent f by splines and FNN, respectively. 'KILE' and 'KALE' (Cressie and Kornak, 2003) are kriging methods for Gaussian process regression (i.e., $f(x)$ is assumed to be randomly generated from a stationary Gaussian process; see Appendix E), without and with measurement errors, respectively. We include 'KILE' and 'KALE' partially because they are fast to compute for $d > 1$ while other classical methods in Table 1 are computationally expensive for $d > 1$.

TABLE 1
Methods included in numerical experiments.

Method	Description	Method	Description
deconv	deconvolution	SIMEX	simulation extrapolation
lpoly	local polynomial deconvolution	KILE	kriging with a radial basis kernel ($*, \dagger$)
KALE	kriging, accounting for measurement errors ($\dagger$)	Pspline	penalized regression splines ($*$)
NN	neural networks for regression ($*$)	BSSP	B-splines for measurement error model
NNME	neural networks for measurement error model		

$*$: ignoring measurement errors. $\dagger$: assuming *a priori* that f follows a Gaussian random field.

5.1 Example 1: Deterministic functions

We consider a smooth function, $f(x) = \frac{\sin((3x-1.5)\pi)}{1+4(6x-3)^2[\mathrm{sgn}(2x-1)+1]}$ for $x \in \mathbb{R}$, used in the simulation study of Berry, Carroll and Ruppert (2002), and generate data as in model (4), where $U \sim \mathcal{N}(0, \sigma_0^2)$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$, and either $X \sim \mathrm{Unif}[0,1]$ or $X \sim \mathrm{Beta}(2,2)$. Two scenarios are considered: (a) large response error and small measurement error, $(\sigma, \sigma_0) = (0.3, 0.1)$; (b) small response error and large measurement error, $(\sigma, \sigma_0) = (0.1, 0.2)$. For each scenario, we let n range in $\{500, 1000, 2000\}$. The performance is measured by the ISE, $\int_0^1 [\hat{f}(x) - f(x)]^2 dx$, computed as the Riemann sum over 1000 equally spaced points.

Several methods in Table 1 have tuning parameters. For classical nonparametric methods, how to select data-driven tuning parameters is largely unclear. We instead use the *ideal tuning parameters* that minimize the true loss function (e.g., the ISE), which biases in favor of these methods. In Pspline and BSSP, we use cubic B-splines with equally spaced knots, with numbers of knots ranging from 5 to 30. The one that minimizes the true ISE is finally chosen.

TABLE 2

Example 1 (smooth function), small measurement error ($\sigma_0 = 0.1$, $\sigma = 0.3$). The ISE and its standard deviation (in brackets) are based on 50 repetitions.

	$X \sim \text{Uniform}(0,1)$			$X \sim \text{Beta}(2,2)$		
	n=500	n=1000	n=2000	n=500	n=1000	n=2000
BSSP	.026 (.003)	.020 (.003)	.013 (.002)	.055 (.005)	.051 (.005)	.045(.004)
Pspline	.050 (.002)	.048 (.001)	.047 (.010)	.089 (.003)	.089 (.002)	.085 (.002)
SIMEX	.015 (.001)	.009 (.001)	.006 (.000)	.025 (.002)	.020 (.002)	.016 (.001)
deconv	.033 (.002)	.027 (.001)	.021 (.001)	.055 (.005)	.045 (.005)	.036 (.004)
lpoly	.039 (.002)	.034 (.001)	.029 (.001)	.077 (.003)	.077 (.003)	.071 (.002)
KALE	.052 (.012)	.098 (.017)	.250 (.012)	.076 (.006)	.078 (.006)	.102 (.009)
KILE	.050 (.002)	.048 (.001)	.046 (.001)	.086 (.002)	.085 (.002)	.082 (.002)
NN	.047 (.002)	.045 (.001)	.045 (.001)	.092 (.003)	.081 (.002)	.082 (.002)
NNME	**.013** (.001)	**.008** (.001)	**.005** (.000)	**.024** (.004)	**.015** (.003)	**.011** (.001)

TABLE 3

Example 1 (smooth function), large measurement error ($\sigma_0 = 0.2$, $\sigma = 0.1$). The ISE and its standard deviation (in brackets) are based on 50 repetitions.

	$X \sim \text{Uniform}(0,1)$			$X \sim \text{Beta}(2,2)$		
	n=500	n=1000	n=2000	n=500	n=1000	n=2000
BSSP	.164 (.007)	.170 (.005)	.168 (.005)	.279 (.007)	.270 (.005)	.259 (.006)
Pspline	.185 (.003)	.179 (.002)	.178 (.001)	.247 (.003)	.246 (.003)	.242 (.002)
SIMEX	.125 (.006)	.112 (.004)	.111 (.002)	.215 (.006)	.208 (.005)	.197 (.004)
deconv	.140 (.006)	.127 (.005)	.125 (.004)	.209 (.007)	.213 (.006)	.194 (.005)
lpoly	.155 (.005)	.150 (.004)	.151 (.003)	.229 (.005)	.237 (.003)	.228 (.003)
KALE	.236 (.007)	.229 (.009)	.227 (.009)	.273 (.005)	.283 (.005)	.282 (.005)
KILE	.185 (.003)	.179 (.002)	.180 (.001)	.241 (.003)	.243 (.002)	.243 (.002)
NN	.174 (.003)	.173 (.003)	.175 (.001)	.239 (.005)	.244 (.003)	.239 (.002)
NNME	**.021** (.003)	**.019** (.002)	**.014** (.001)	**.059** (.005)	**.058** (.004)	**.045** (.003)

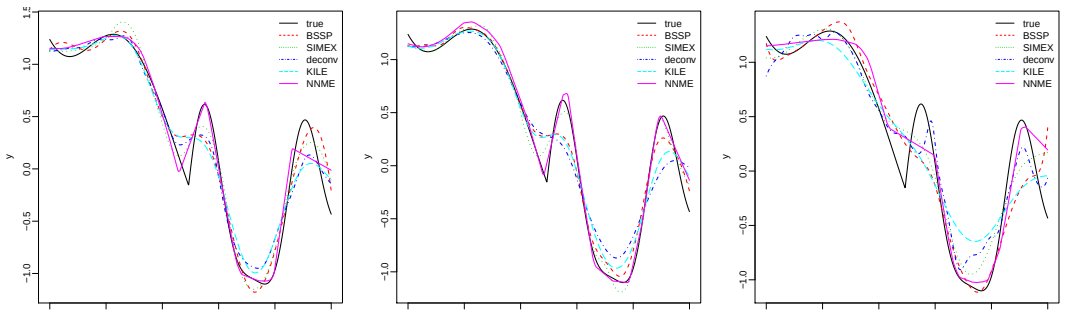
BSSP requires initial estimates of spline coefficients. We initialize by fitting B-splines on the observed data, ignoring measurement errors. In SIMEX, the nonparametric method to plug in is the regression spline on truncated power bases. The selection of knots and penalization parameter, as well as the initialization of the coefficients, are similar to that for BSSP. In 'deconv', we set the bandwidth as $\sigma_0[\log(n)]^{-1/2}$ (σ_0^2 is the known variance of measurement errors), the theoretically optimal one for density deconvolution under Gaussian errors (Stefanski and Carroll, 1990). In 'lpoly', we set the polynomial degree as 1. We run the methods for 5 bandwidth values near $1.06\sigma_0 n^{-1/5}$ and select the one that minimizes the ISE. For NNME, we use 6 layers in the decoder and 3 layers in the encoder, where each layer has 32 nodes. After centering and scaling data, we use $2 \cdot t_3$ to model the prior distribution of X. We set the L2-regularization parameters as $\lambda_0 = \lambda_1 = 10^{-5}$ and $\lambda_2 = 0$.

The results are in Tables 2-3. When the measurement error is small (Table 2), NNME and SIMEX have comparable performances and attain the smallest ISE in all settings. BSSP, deconv, and lpoly perform similarly, among which BSSP is the best when $X \sim \text{Unif}[0,1]$ and deconv is the best when $X \sim \text{Beta}(2,2)$. The two methods that ignore measurement errors, Pspline and NN, are significantly worse than their counterparts that account for measurement errors. KILE and KALE have unsatisfactory performances, especially for large n. This is not surprising as these kriging methods assume a different model on $f(x)$. When the measurement error is large (Table 3), NNME is significantly better than all other methods. It is worth noting

that the performance of NN is comparable to other methods, suggesting that the advantage of NNME indeed comes from a proper account of measurement errors.

5.2 Example 2: Functions generated from a Gaussian process

We consider an example where $f(x)$ is generated from a Gaussian process (GP) on $[0,1]$ with two different radial basis function (RBF) kernels on intervals $[0,0.5]$ and $[0.5,1]$, respectively. Both kernels are squared exponential kernels of the form $K_\ell(x,y)=\exp(-\beta_\ell|x-y|^2)$, where $\beta_1=16$ and $\beta_2=64$. A GP with a squared exponential kernel has mean squared derivatives of all orders and is smooth with high probability (Rasmussen, 2003). Therefore, $f(x)$ is smooth within $[0,0.5]$ and $[0.5,1]$, but not at 0.5. In simulations, we let $n=2m$ be even and generate $f(x)$ as follows: (i) sample $\{x_i\}_{i=1}^n$ uniformly from $[0,1]$ and relabel them so that $x_1<\ldots<x_n$; (ii) sample $\{f(x_i)\}_{1\le i\le m}$ from the first GP and $\{\tilde f(x_i)\}_{m\le i\le n}$ from the second GP; (iii) define $f(x_i)=\tilde f(x_i)+f(x_m)-\tilde f(x_m)$ for $i\in[m:n]$ to make $f(x)$ continuous at x_m (with high probability, x_m is located near 0.5). Examples of $f(x)$ from simulations are shown in Figure 5 by solid black curves. Clearly, the first and second half of the curve have different smoothness, and the connection at the middle is erratic. Given $\{(x_i,f(x_i))\}_{i=1}^n$, we obtain $\{(w_i,y_i)\}_{i=1}^n$ by generating $u_i\overset{iid}{\sim}\mathcal{N}(0,\sigma_0^2)$. We fix $\sigma=0.2$ and let σ_0 range in $\{0.02,0.05,0.1,0.2\}$. For each σ_0, we consider $n\in\{500,1000,2000\}$.



(a) (σ,σ_0)=(0.2,0.05), n=500. (b) (σ,σ_0)=(0.2,0.05), n=2000. (c) (σ,σ_0)=(0.2,0.1), n=2000.

Fig 5: Functions generated from GP with a mixture of two kernels as described in Section 5.2.

The results are in Table 4. SIMEX is the best for $\sigma_0=0.02$, and NNME is the best for other values of σ_0. KALE is significantly worse than the other methods, because the model of KALE does not hold exactly (f is not from a single GP). NN, Pspline, and KILE are methods that ignore measurement errors; they perform well when $\sigma_0=0.02$ but unsatisfactorily for larger σ_0. We overlay the estimated curves in Figure 5. On panel (b) of Figure 5, the classical methods (deconv, BSSP, and SIMEX) fail to capture the local curvature around $x=x_{n/2}$ and oversmooth in the second half interval, but NNME fits the true curve very well. When the sample size reduces to 500 (see panel (a)), NNME fits the true curve well except at the boundary; in this case, there are fewer samples at the boundary, and NNME produces a nearly linear curve at the boundary as a result of using the ReLU activation function. When σ_0 increases to 0.1 (see panel (c)), NNME fits well each half of the curve but misses the curvature around $x=x_{n/2}$; in this case, the large measurement error mask the local curvature of the

TABLE 4
Example 2 (function from Gaussian process). The ISE and its standard deviation (in brackets) is shown.

	$\sigma_0 = 0.02$			$\sigma_0 = 0.05$		
	n=500	n=1000	n=2000	n=500	n=1000	n=2000
BSSP	.015 (.003)	.012 (.002)	.011 (.002)	.035 (.006)	.033 (.005)	.031 (.005)
Pspline	.004 (.001)	.003 (.000)	.002 (.000)	.032 (.005)	.031 (.005)	.027 (.004)
SIMEX	**.003** (.000)	**.002** (.000)	**.001** (.000)	**.016** (.003)	.012 (.002)	.008 (.001)
deconv	.018 (.002)	.011(.002)	.007 (.001)	.028 (.005)	.022 (.004)	.016 (.003)
lpoly	.014 (.002)	.009 (.001)	.006 (.001)	.044 (.006)	.024 (.004)	.018 (.003)
KALE	.055 (.034)	.035 (.011)	.070 (.030)	.155 (.039)	.181 (.055)	.168 (.046)
KILE	.004 (.000)	.003 (.000)	.002 (.000)	.032 (.005)	.030 (.004)	.027 (.004)
NN	.015 (.006)	.009 (.003)	.006 (.002)	.039 (.016)	.037 (.009)	.036 (.006)
NNME	.006 (.001)	.003 (.000)	.002 (.000)	**.016** (.003)	**.009** (.002)	**.006** (.001)
	$\sigma_0 = 0.1$			$\sigma_0 = 0.2$		
	n=500	n=1000	n=2000	n=500	n=1000	n=2000
BSSP	.103 (.017)	.092 (.017)	.092 (.016)	.328 (.036)	.328 (.039)	.319 (.042)
Pspline	.126 (.017)	.121 (.015)	.120 (.015)	.352 (.037)	.341 (.035)	.332 (.036)
SIMEX	.065 (.011)	.052 (.008)	.048 (.007)	.264 (.033)	.236 (.029)	.221 (.028)
deconv	.087 (.014)	.079 (.011)	.068 (.009)	.277 (.033)	.263 (.030)	.247 (.030)
lpoly	.094 (.015)	.089 (.013)	.081 (.011)	.301 (.034)	.295 (.031)	.285 (.031)
KALE	.193 (.028)	.297 (.068)	.341 (.060)	.359 (.039)	.397 (.056)	.402 (.056)
KILE	.136 (.018)	.124 (.015)	.122 (.015)	.353 (.039)	.341 (.036)	.334 (.036)
NN	.124 (.034)	.121 (.028)	.125 (.026)	.342 (.206)	.332 (.182)	.331 (.157)
NNME	**.034** (.006)	**.028** (.004)	**.026** (.006)	**.206** (.037)	**.182** (.032)	**.157** (.030)

true curve. The classical methods (e.g., deconv, BSSP, SIMEX) may be improved by using different bandwidths or knots in $[0, x_{n/2}]$ and $[x_{n/2}, 1]$, but this requires prior knowledge of $x_{n/2}$. The neural network approach can adaptively impose different levels of smoothing in two regions, without any prior knowledge.

5.3 Example 3: Two-dimensional functions

We consider f generated from a 2-dimensional Gaussian processes. Given $\beta > 0$, we first draw $\{x_i\}_{1 \le i \le n}$ from the distribution of X, then construct $\Sigma \in \mathbb{R}^{n \times n}$ by $\Sigma_{ij} = \exp(-\beta \|x_i - x_j\|^2)$, and finally draw $(f(x_1), f(x_2), \ldots, f(x_n))$ from $\mathcal{N}(0, \Sigma)$. We consider two settings. In the first one, $\beta = 16$. In the second one, we generate $f_1(x)$ and $f_2(x)$ with β equal to 16 and 4, respectively, and let $f(x) = \max\{f_1(x), f_2(x)\}$. An example of the realized $f(x)$ from the second setting is shown in Figure A6. For each setting, we set the distribution of X to be a 2-component mixture of multivariate normal distributions, $0.7\mathcal{N}(\mu_1, \Sigma_1) + 0.3\mathcal{N}(\mu_2, \Sigma_2)$, where (μ_1, Σ_1) and (μ_2, Σ_2) are the same as in (9). Given the x_i and $f(x_i)$, we generate w_i and y_i by adding Gaussian errors $\mathcal{N}(0, \sigma_0^2 I_2)$ and $\mathcal{N}(0, \sigma^2)$ respectively. We fix $\sigma = 0.2$, and consider $\sigma_0 \in \{0.05, 0.1, 0.2\}$ and $n \in \{500, 1000, 2000, 4000, 8000\}$. The performance metric ISE is computed via a uniform grid on the union of two rectangular regions, $[-1, 0.2] \times [-1, 0.5] \cup [-0.5, 1] \times [-0.2, 1]$ (we restrict to this region because there are too few training x_i's outside this region). In NNME, we use 9 hidden layers in the decoder and 5 hidden layers in the encoder, with 32 nodes per layer. More results about the sensitivity to tuning is contained in Appendix D. How to choose the model for X was investigated in Section 3.2. We explore two options: NNME_NICE uses the NICE model, and NNME_GM4 models X by a 4-component Gaussian mixture distribution with unknown parameters.

TABLE 5

Example 3 (two-dimensional function from Gaussian processes). The ISE evaluated at a uniform grid, as well as its standard deviation (in brackets), is shown. Top sub-table: f is from a Gaussian process with $\beta = 16$. Bottom sub-table: f is the maximum of two Gaussian processes with $\beta = 16$ and $\beta = 4$, respectively.

		$n = 500$	1000	2000	4000	8000
$\sigma_0 = 0.05$	KILE	**.219** (.019)	.138 (.011)	.103 (.008)	-	-
	KALE	.222 (.019)	**.136** (.011)	**.098** (.008)	-	-
	NNME_GM4	.362 (.033)	.248 (.016)	.163 (.012)	.110 (.009)	**.071** (.008)
	NNME_NICE	.346 (.036)	.232 (.014)	.177 (.017)	**.100** (.010)	.079 (.006)
$\sigma_0 = 0.1$	KILE	.415 (.021)	.349 (.019)	.280 (.015)	-	-
	KALE	**.414** (.021)	**.335** (.019)	.259 (.015)	-	-
	NNME_GM4	.455 (.027)	.364 (.024)	.258 (.018)	**.167** (.019)	**.108** (.009)
	NNME_NICE	.483 (.049)	.398 (.039)	**.250** (.020)	.202 (.033)	.124 (.014)
$\sigma_0 = 0.2$	KILE	**.773** (.040)	.711 (.038)	.682 (.042)	-	-
	KALE	.809 (.040)	.748 (.038)	.714 (.042)	-	-
	NNME_GM4	.820 (.081)	.696 (.066)	.548 (.037)	**.389** (.024)	**.313** (.027)
	NNME_NICE	.815 (.065)	**.670** (.058)	**.459** (.050)	.429 (.045)	.350 (.035)
$\sigma_0 = 0.05$	KILE	**.126** (.011)	**.092** (.011)	**.063** (.005)	-	-
	KALE	.143 (.011)	.104 (.011)	.071 (.005)	-	-
	NNME_GM4	.167 (.018)	.112 (.011)	.073 (.005)	**.047** (.004)	**.037** (.004)
	NNME_NICE	.205 (.024)	.121 (.013)	.080 (.007)	.063 (.009)	.040 (.005)
$\sigma_0 = 0.1$	KILE	**.226** (.020)	.187 (.019)	.150 (.009)	-	-
	KALE	.232 (.020)	.187 (.019)	.147 (.009)	-	-
	NNME_GM4	.251 (.027)	**.169** (.019)	**.116** (.009)	**.079** (.005)	.076 (.009)
	NNME_NICE	.303 (.030)	.175 (.022)	.140 (.013)	.103 (.013)	**.072** (.010)
$\sigma_0 = 0.2$	KILE	**.376** (.037)	.362 (.037)	.351 (.034)	-	-
	KALE	.382 (.037)	.371 (.037)	.363 (.034)	-	-
	NNME_GM4	.400 (.042)	**.321** (.028)	**.268** (.035)	**.192** (.027)	**.159** (.018)
	NNME_NICE	.442 (.042)	.367 (.052)	.300 (.055)	.241 (.039)	.181 (.021)

The results are in Table 5. The data generating process in the first setting is exactly the same as the KALE model, and one would expect KALE to perform the best. This is true when the measurement error is small ($\sigma_0 \in \{0.05, 0.1\}$) or the sample size is small to moderate ($n \in \{500, 1000\}$). However, when the measurement error is large ($\sigma_0 = 0.2$) or the sample size is $n = 2000$, NNME outperforms KALE. Between the two versions of NNME, NNME_NICE is better. We note that the two kriging methods require to invert an $n \times n$ matrix. Therefore, they do not scale to large n. In fact, we are not able to run these methods for $n \in \{4000, 8000\}$. In the second setting, f is no longer from the KALE model. NNME uniformly outperforms KILE and KALE when $n > 500$ and $\sigma_0 > 0.05$; in other cases, KILE performs the best. Between the two versions of NNME, NNME_GM4 is better.

5.4 Summary

We compared NNME with classical nonparametric methods in settings where f is a smooth or non-smooth function, and we also compared NNME with kriging methods when f is generated from a 2-dimensional Gaussian process. In classical nonparametric settings (e.g., Example 1), the nonparametric methods indeed perform well (for example, SIMEX has the best performance when measurement error is small). However, NNME attains comparable performance, sometimes even better. One possible reason is that NNME maximizes the marginal log-likelihood and can be more efficient than methods that do not use the likelihood. Another reason is that the tuning parameter selection in classical methods is sometimes challenging (e.g., in Example 2, classical methods will improve if multiple bandwidths or unequally

spaced knots are used). Our numerical results do not contradict the minimax optimality of classical methods, because the optimality there is about the "worst-case" performance. In the settings where f is generated from a Gaussian process (Example 3), the kriging methods indeed perform well, but they work unsatisfactorily for other measurement error model settings (e.g., Examples 1 and 2). NNME has reasonably good performance in all settings without requiring any prior knowledge. Computationally it is also much more efficient than classical methods when $d > 1$.

Due to space limit, we relegate some numerical results into the Appendix. In Appendix D, we study choosing the numbers of layers in NNME by cross validation and provide sensitivity analysis. In Appendix F, we present simulations with f generated by a neural network.

6. REAL DATA APPLICATIONS

6.1 Historical sea level estimation

The data set in Kemp et al. (2011) consists of measurements of relative sea level (RSL) in North Carolina for the past 2000 years. Following Cahill et al. (2015), we use a measurement error framework: let y_i be the observed RSL, x_i the true calendar year, and w_i the estimated calendar year. Let $g(x_i)$ be the ocean level at year x_i. The true RSL is the ocean level minus the land level, where by glacio-isostatic adjustment, the land level decreases at an annual rate of $0.001r$ with known r (for this dataset, $r = 0.9$ or 1 depending on the measuring sites). The model is

$$w_i = x_i + u_i, \qquad y_i = g(x_i) - [c_0 + r \cdot (2.010 - x_i)] + \epsilon_i, \tag{21}$$

where the calendar years x_i and w_i have been divided by 1000 (e.g., year 1996 is written as 1.996) and c_0 is the land level in 2010 AD. Both u_i and ϵ_i are assumed to be normal:

$$u_i \sim \mathcal{N}\left(0, \sigma_{u_i}^2\right), \qquad \epsilon_i \sim \mathcal{N}\left(0, \tau^2 + \sigma_{\epsilon_i}^2\right), \tag{22}$$

where $\sigma_{u_i}^2$ and $\sigma_{\epsilon_i}^2$ are known for each observation. The data are $\{(w_i, y_i, \sigma_{u_i}^2, \sigma_{\epsilon_i}^2)\}_{1 \le i \le n}$. The goal is estimating the function $g(x)$. Without loss of generality, we let $c_0 = 0$, so that the ocean level in 2010 AD is viewed as the baseline.

Defining $f(x) = g(x) - r(2.010 - x)$, we first apply NNME to estimate $f(x)$ and then convert it to an estimate of $g(x)$. A minor difference from the previous measurement error models is that the variances of response errors and measurement errors are heterogeneous across observations. Since both $\sigma_{u_i}^2$ and $\sigma_{\epsilon_i}^2$ are known, we extend our algorithm by incorporating $(\sigma_{u_i}^2, \sigma_{\epsilon_i}^2, \tau^2)$ into the marginal likelihood and modifying the gradient ascent steps accordingly. We also modify (20) to an estimate of τ^2 as $\hat{\tau}^2 = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{1}{K}\sum_{k=1}^{K}(y_i - f(x_{ik}))^2 \frac{\beta_{ik}}{\sum_{\ell=1}^{K}\beta_{i\ell}} - \sigma_{\epsilon_i}^2\right\}$, where $x_{i1}, x_{i2}, \ldots, x_{iK}$ are importance samples for the ith observation and β_{ik} is ratio between the complete likelihood and the density of proposal distribution, similar to that in (20). We assume that X follows a 2-component mixture of Gammas *a priori*. We let the decoder and encoder have 5 and 3 hidden layers, respectively, with 32 nodes per layer. The activation function is chosen as tanh in the decoder (to make $\hat{f}$ smooth) and ReLU in the encoder. Since the measurement error variances are small (from 2.5×10^{-7} to 0.009) compared to the date variances (about 0.33), we also add a direct link from W to μ_ϕ in

Figure 2 to accelerate the learning process. It means that the output of the encoder becomes $\mu(w, y) = \mu_\phi(w, y) + w$, where μ_ϕ is from a 3-layer FNN. Besides an estimate of f, we also compute a 95% confidence band via a parametric bootstrap procedure. In the standard model-based bootstrap, we should draw samples $\{(x_i^*, w_i^*, y_i^*)\}_{i=1}^n$ from the estimated model for X and model (21)-(22) for (W, Y). However, the noise variances in covariates and responses are only known at the observed sites. Hence, we cannot use the standard model-based bootstrap. We instead fix $x_i^* = \mu(w_i, y_i)$, the estimated posterior mean of x_i from the encoder, and draw (w_i^*, y_i^*) from model (21)-(22) using the known variances for the ith observation. The resulting bootstrap confidence band can be viewed as conditioning on (estimates of) x_i's.

The left panel of Figure 6 shows the estimated ocean level, which is $\hat{g}(x) = \hat{f}(x) + r(2.010 - x)$. The right panel shows the estimated sea level change, which is the derivative of $\hat{g}$. It is computed by a back propagation algorithm using estimated parameters. The confidence band of $\hat{g}$ is from a similar bootstrap procedure. We compare NNME with the approach in Cahill et al. (2015), denoted as "GP", which models the sea level change (i.e., derivative of g) by a Gaussian process with measurement errors and uses Markov chain Monte Carlo for estimation and inference. While it is based on Gaussian process, this method is different from KALE. The estimated curves of sea level (left panel of Figure 6) by NNME and GP are similar, except in the period between 1600 AD and 1800 AD. Both methods estimate the sea level to decrease first and increase later in this period, but NNME estimates this "fluctuation" to be less prominent. For GP, we plot the 95% Bayesian credible interval (Cahill et al., 2015). Although the credible interval is not directly comparable with the confidence band, we may still draw the conclusion from the plots that NNME gives less "confidence" on the fluctuation of sea level between 1600 AD and 1800 AD. The estimated curves of sea level change (right panel of Figure 6) are also similar. The curve by GP is smoother. A possible explanation is that GP directly models the sea level change while NNME models the sea level.

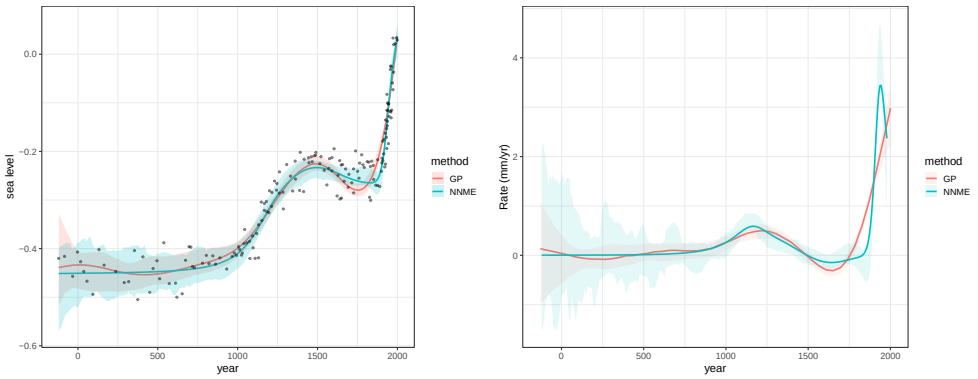


Fig 6: Estimated sea level (left panel) and estimated rate of sea level change (right panel). The 95% confidence band for NNME and 95% credible interval for GP are shown.

Since there is no ground truth, we evaluate the performance by the prediction mean squared errors (MSEs). Given $\hat{f}$ and a testing w, we predict y by $\mathbb{E}[\hat{f}(x) \mid w]$, where the expectation is with respect to the posterior distribution of X given W. We use two ways to approximate the posterior distribution of X. The first is $\mathcal{N}(w, \sigma_0^2)$, where σ_0^2 is the variance of measurement

TABLE 6
Prediction mean squared errors for each method in consideration on the sea level data set.

NNME_posterior1	NNME_posterior2	GP_posterior1	GP_posterior2	NN
1.387 (0.068)	1.402 (0.069)	1.499 (0.065)	1.503 (0.065)	2.478 (0.020)

errors in this observation. This ad-hoc approach can be viewed as imposing a flat prior on X. The second is using the estimated prior distribution of X from NNME to derive the posterior distribution of X. In the actual implementation, to obtain $\mathbb{E}[\hat{f}(x) \mid w]$, we first draw samples $\{x_i^*\}$ from the conditional distribution of X given w, and then take a weighted average of $f(x_i^*)$, with weights proportional the prior density of x_i^*. For $\hat{f}$ resulting from both NNME and GP, we construct predictors using both ways, denoted as "_posterior1" and "_posterior2" respectively. The prediction MSE of each method is estimated by a 5-fold cross-validation procedure: we randomly partition data into 5 folds and then successively leave one fold out for testing (i.e., computing the MSE between predictions and observations) with the other 4 folds used to train the model; we then average the 5 MSEs to arrive at the overall prediction MSE. We repeat this procedure 10 times and report the mean and standard error of the prediction MSEs. Table 6 shows the prediction MSEs for both NNME and GP, as well as NN. Since NN ignores measurement errors in estimating f, it simply predicts $f(x)$ by $\hat{f}(w)$. The performance of NN is much worse than NNME, suggesting that the advantage of NNME comes from not only using neural networks but also accounting for measurement errors. NNME performs the best in terms of prediction MSE, and the one resulting from using a flat prior for X appears to be slightly better.

6.2 Framingham Heart Study

The Framingham Heart Study is an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The dataset includes over 4,000 records and 15 attributes of patients. We downloaded the data set from Kaggle (https://www.kaggle.com/dileep070/heart-disease-prediction-using-logistic-regression). The covariate of special interest is the systolic blood pressure (SBP). The long-term SBP is estimated from averaging the reads in several clinic visits of the same patient and has measurement errors. Other covariates were assumed error-free, except the total cholesterol level (Chol). We followed Carroll et al. (2006) to model the measurement errors on $\log(\text{SBP}) - 50)$ and $\log(\text{Chol})$ as bivariate Gaussian with a given covariance matrix. We fit a nonparametric logistic regression for predicting the 10-year risk of coronary heart disease (CHD): $\mathbb{P}(Y = 1|X) = L\big(\beta_0 + \sum_j \beta_j X_j + f(X_{SBP}, X_{Chol})\big)$, where $L(x)$ is the logistic sigmoid function, $Y \in \{0, 1\}$ indicates whether this patient has CHD in 10 years, X_j's are error-free covariates, and for (X_{SBP}, X_{Chol}) only error-prone observations, (W_{SBP}, W_{Chol}), are available. Previous studies used a linear logistic regression model with measurement errors, but our approach can estimate the nonlinear, interaction effects.

We pre-processed data by dividing the variable *age* by 100 and then centering all variables to have mean zero. We adapted NNME to the current setting by changing the form of the log-likelihood. NICE is used to model the (prior) joint distribution of SBP and Chol. The training is similar as that in Section 4.4 (IWAE+DReG). We used 3 hidden layers for both the decoder

and encoder, with 32 nodes per layer and ReLU as the activation function. To accommodate the logistic model, we used the sigmoid function in the last layer of the decoder. The estimated 10-year risk of CHD, as a 2-dimensional function of SBP and Chol, is shown in Figure 7.

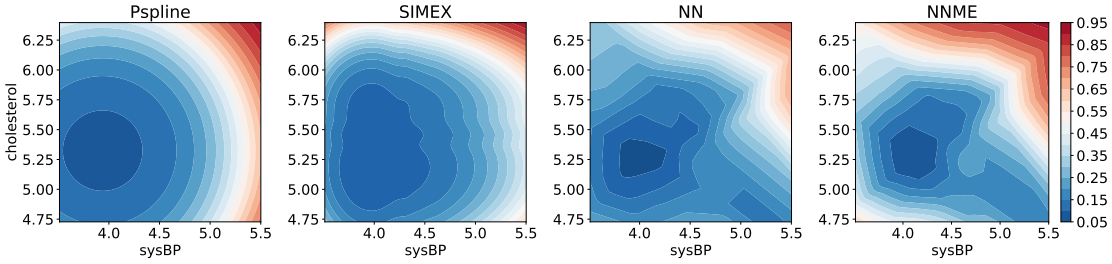


Fig 7: Estimated 10-year risk of CHD as a function of SBP and Chol, where other covariates take the mean values. The contours are for a 2-dimensional function on (x_{SBP}, x_{Chol}), defined by $L\big(\hat{\beta}_0 + \sum_j \hat{\beta}_j \bar{x}_j + \hat{f}(x_{SBP}, x_{Chol})\big)$.

For comparison, we also analyze the data with SIMEX, which requires as the plug-in a nonparametric logistic regression method that ignores measurement errors, for which we employ PSpline (degree = 2, number of knots = 10) and use linear extrapolation. The code to implement SIMEX basically allows for only 1 error-prone covariate; to apply it to 2 error-prone covariates, we have to assume an additive model. Consequently, this approach does not model interaction effects between SBP and Chol. In contrast, NNME includes a bivariate function of SBP and Chol in the risk and is able to capture interaction effects. Previously in Section 5, we also considered KALE for 2 error-prone covariates; unfortunately, KALE has no direct extension to the logistic regression model. The estimated 10-year risks of CHD by NNME and SIMEX are shown in Figure 7. For a better comparison, we also considered methods that ignore measurement errors, NN and Pspline, as counterparts of NNME and SIMEX. The comparison of risk contours of Pspline versus NN suggests that neural networks can capture nonlinear, beyond-quadratic effects of both SBP and Chol, as well as the interaction effect between two covariates. SIMEX generates more sophisticated risk contours than Pspline, but SIMEX still does not model interaction effects. NNME is the only one among 4 methods that accommodates measurement errors, nonlinear effects, and interaction effects.

To check whether NNME overfits, we evaluated the classification performance. Given an estimated model, we classify a sample by thresholding $\mathbb{E}[Y \mid \{X_j\}, W_{SBP}, W_{Chol}]$, where the posterior distribution of (X_{SBP}, X_{Chol}) is obtained from the measurement error distribution and the prior distribution from NICE. To avoid discussion of thresholds, we measure the performance by the area under ROC curve (AUC). We randomly selected 20% of CHD samples and 20% of non-CHD samples for testing and used the remaining samples for training. The mean and standard deviation of AUC, over 20 random splits of training and testing, are: The AUCs of different methods are similar. At least, it suggests that the more sophisticated models from neural networks are not due to overfitting. Additionally, accounting for measurement errors barely improves the classification performance.

TABLE 7
Classification accuracy for each method in consideration on the Framingham data set.

Pspline	SIMEX	NN	NNME
0.727 (0.005)	0.728 (0.005)	0.727 (0.005)	0.728 (0.005)

7. DISCUSSION

The use of neural networks in nonparametric statistics attracted a lot of recent attention, with encouraging progress on density estimation and nonparametric regression. This paper is an attempt to introduce neural networks to estimation of measurement error models, one of the classical topics in nonparametric statistics (Carroll and Hall, 1988; Fan and Truong, 1993; Carroll, Maca and Ruppert, 1999). We propose a neural network design, where the regression function $f(x)$, the prior distribution of X, and a "proposal distribution" that approximates the posterior distribution of X under our model, which is used to derive the ELBO and direct the training, are represented by three different neural networks. We estimate parameters of these neural networks by maximizing ELBO, a lower bound of the marginal log-likelihood of (W, Y), and solve the optimization problem by a stochastic gradient ascent algorithm, with a doubly reparametrized gradient estimator. Our algorithm combines recent advancements in neural network, including Burda, Grosse and Salakhutdinov (2016) on variational auto-encoder, Tucker et al. (2018) on stochastic gradient descent, and Dinh, Krueger and Bengio (2015) on normalizing flow.

Through extensive simulations and real data analysis, we demonstrate that the neural network approach is a promising alternative to classical nonparametric methods for measurement error models. The neural network approach is flexible in accommodating various classes of functions (even non-smooth functions); its performance is relatively insensitive to tuning parameters; it is convenient to implement for dimension $d > 1$; and it has good scalability to a large sample size. Additionally, our method can be easily extended to more general settings. For example, if the noise in X or the noise in Y is non-additive, our method can be implemented similarly, where we simply change the expression of the joint density of (X, W, Y). In contrast, classical nonparametric methods are more restrictive on model assumptions. For example, the deconvolution method (Fan and Truong, 1993) relies on the assumption that the measurement error is additive.

Theoretical understanding of neural network methods is a trending topic. Many theoretical frameworks have been proposed, such as size-independent complexity (Bartlett, 1998; Golowich, Rakhlin and Shamir, 2018), implicit regularization (Soudry et al., 2018), seive approximation (Chen and White, 1999), mean-field approximation (Mei, Montanari and Nguyen, 2018), and so on. Whether or not these theoretical frameworks can be used to understand the behavior of our method for measurement error models is an open problem. We leave it for future work.

We consider a fully nonparametric setting, where neither f nor the density of X is from a parametric model. The semi-parametric setting where the density of X is nonparametric but f is parametric has also been considered in the literature (Taupin, 2001; Butucea and Taupin, 2008). NNME could extend to this setting by replacing FNN by the given parametric model,

with other components (NICE and the inference network) unchanged. We also leave this to future investigation.

ACKNOWLEDGEMENTS

The authors thank Editor Sonia Petrone and the anonymous AE and referees for their great comments, which help improve the paper. The authors gratefully acknowledge the support of NSF grants, DMS-1943902 to Z Ke, and DMS-1903139 and DMS-2015411 to JS Liu.

SUPPLEMENTARY MATERIAL

Supplementary document. Appendix A to F. Appendix A provides the full algorithm of training NNME. Appendix B and C show the comparisons of different neural network structures for measurement error models and different prior models for X in NNME respectively. Appendix D shows the sensitivity of NNME to the depth of neural networks. Appendix E introduces kriging methods for Gaussian process regression. Appendix F shows an example of two-dimensional functions generated from a neural network.

REFERENCES

BARTLETT, P. L. (1998). The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory* **44** 525-536.

BAUER, B. and KOHLER, M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *The Annals of Statistics* **47** 2261–2285.

BERRY, S. M., CARROLL, R. J. and RUPPERT, D. (2002). Bayesian smoothing and regression splines for measurement error problems. *Journal of the American Statistical Association* **97** 160–169.

BLEI, D. M., KUCUKELBIR, A. and MCAULIFFE, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association* **112** 859–877.

BURDA, Y., GROSSE, R. and SALAKHUTDINOV, R. (2016). Importance weighted autoencoders. In *International Conference on Learning Representations*.

BUTUCEA, C. and TAUPIN, M.-L. (2008). New M-estimators in semi-parametric regression with errors in variables. In *Annales de l'IHP Probabilités et statistiques* **44** 393–421.

BUZ AS, J. S. (1997). Instrumental variable estimation in nonlinear measurement error models. *Communications in Statistics-Theory and Methods* **26** 2861–2877.

CAHILL, N., KEMP, A. C., HORTON, B. P. and PARNELL, A. C. (2015). Modeling sea-level change using errors-in-variables integrated gaussian processes. *The Annals of Applied Statistics* **9** 547–571.

CARROLL, R. J. and HALL, P. (1988). Optimal rates of convergence for deconvolving a density. *Journal of the American Statistical Association* **83** 1184–1186.

CARROLL, R., MACA, J. and RUPPERT, D. (1999). Nonparametric regression in the presence of measurement error. *Biometrika* **86** 541-554.

CARROLL, R. J., ROEDER, K. and WASSERMAN, L. (1999). Flexible parametric measurement error models. *Biometrics* **55** 44-54.

CARROLL, R. J. and STEFANSKI, L. A. (1994). Measurement error, instrumental variables and corrections for attenuation with applications to meta-analyses. *Statistics in Medicine* **13** 1265-1282.

CARROLL, R. J., RUPPERT, D., CRAINICEANU, C. M., TOSTESON, T. D. and KARAGAS, M. R. (2004). Nonlinear and nonparametric regression and instrumental variables. *Journal of the American Statistical Association* **99** 736–750.

CARROLL, R. J., RUPPERT, D., STEFANSKI, L. A. and CRAINICEANU, C. M. (2006). *Measurement error in nonlinear models: a modern perspective*. Chapman and Hall/CRC.

CHEN, X. and WHITE, H. (1999). Improved rates and asymptotic normality for nonparametric neural network estimators. *IEEE Transactions on Information Theory* **45** 682–691.

COOK, J. R. and STEFANSKI, L. A. (1994). Simulation-extrapolation estimation in parametric measurement error models. *Journal of the American Statistical Association* **89** 1314–1328.

CRAINICEANU, C. M., RUPPERT, D. and WAND, M. P. (2005). Bayesian analysis for penalized spline regression using WinBUGS. *Journal of Statistical Software, Articles* **14** 1–24.

CREMER, C., MORRIS, Q. and DUVENAUD, D. (2017). Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*.

CRESSIE, N. and KORNAK, J. (2003). Spatial statistics in the presence of location error with an application to remote sensing of the environment. *Statistical Science* **18** 436 – 456.

DELAIGLE, A., FAN, J. and CARROLL, R. J. (2009). A design-adaptive local polynomial estimator for the errors-in-variables problem. *Journal of the American Statistical Association* **104** 348–359.

DELAIGLE, A. and HALL, P. (2008). Using SIMEX for smoothing-parameter choice in errors-in-variables problems. *Journal of the American Statistical Association* **103** 280–287.

DELLAPORTAS, P. and STEPHENS, D. A. (1995). Bayesian analysis of errors-in-variables regression models. *Biometrics* **51** 1085–1095.

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

DINH, L., KRUEGER, D. and BENGIO, Y. (2015). NICE: Non-linear independent components estimation. In *Workshop in International Conference on Learning Representations*.

EILERS, P. H. C. and MARX, B. D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science* **11** 89 – 121.

FAN, J., MA, C. and ZHONG, Y. (2021). A selective overview of deep learning. *Statistical Science* **36** 264–290.

FAN, J. and TRUONG, Y. K. (1993). Nonparametric regression with errors in variables. *The Annals of Statistics* **21** 1900–1925.

GANGULI, B., STAUDENMAYER, J. and WAND, M. P. (2005). Additive models with predictors subject to measurement error. *Australian & New Zealand Journal of Statistics* **47** 193-202.

GILKS, W. R. and RICHARDSON, S. (1992). Analysis of disease risks using ancillary risk factors, with application to job–exposure matrices. *Statistics in Medicine* **11** 1443-1463.

GOLOWICH, N., RAKHLIN, A. and SHAMIR, O. (2018). Size-independent sample complexity of neural networks. In *Conference On Learning Theory* 297–299.

GUNASEKAR, S., LEE, J. D., SREBRO, N. and SOUDRY, D. (2018). Implicit bias of gradient descent on linear convolutional networks. *Advances in Neural Information Processing Systems* **2018** 9461–9471.

GUSTAFSON, P. (2003). *Measurement Error and Misclassification in Statistics and Epidemiology*. Chapman and Hall/CRC.

GUSTAFSON, P., LE, N. D. and VALLÉE, M. (2002). A Bayesian approach to case–control studies with errors in covariables. *Biostatistics* **3** 229-243.

HALL, P. and QIU, P. (2005). Discrete-transform approach to deconvolution problems. *Biometrika* **92** 135–148.

HARDT, M., RECHT, B. and SINGER, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning* 1225–1234. PMLR.

JIANG, F., MA, Y. and CARROLL, R. J. (2018). A spline-assisted semiparametric approach to non-parametric measurement error models. *arXiv preprint arXiv:1804.00793*.

JORDAN, M. I., GHAHRAMANI, Z., JAAKKOLA, T. S. and SAUL, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning* **37** 183–233.

KEMP, A. C., HORTON, B. P., DONNELLY, J. P., MANN, M. E., VERMEER, M. and RAHMSTORF, S. (2011). Climate related sea-level variations over the past two millennia. *Proceedings of the National Academy of Sciences* **108** 11017–11022.

KINGMA, D. P. and BA, J. (2014). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

KINGMA, D. P. and WELLING, M. (2014). Auto-encoding variational bayes. In *International Conference on Learning Representations*.

LE, T. A., IGL, M., RAINFORTH, T., JIN, T. and WOOD, F. (2018). Auto-Encoding Sequential Monte Carlo. In *International Conference on Learning Representations*.

LIN, H. W., TEGMARK, M. and ROLNICK, D. (2017). Why does deep and cheap learning work so well? *Journal of Statistical Physics* **168** 1223–1247.

MAIOROV, V. and MEIR, R. (2000). On the near optimality of the stochastic approximation of smooth functions by neural networks. *Advances in Computational Mathematics* **13** 79–103.

MALLICK, B., HOFFMAN, F. O. and CARROLL, R. J. (2002). Semiparametric Regression Modeling with Mixtures of Berkson and Classical Error, with Application to Fallout from the Nevada Test Site. *Biometrics* **58** 13–20.

MEI, S., MONTANARI, A. and NGUYEN, P.-M. (2018). A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences* **115** E7665–E7671.

MHASKAR, H. N. (1996). Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation* **8** 164–177.

MÜLLER, P. and ROEDER, K. (1997). A Bayesian semiparametric model for case-control studies with errors in variables. *Biometrika* **84** 523–537.

PAPAMAKARIOS, G., PAVLAKOU, T. and MURRAY, I. (2017). Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems* 2338–2347.

RAINFORTH, T., KOSIOREK, A., LE, T. A., MADDISON, C., IGL, M., WOOD, F. and TEH, Y. W. (2018). Tighter Variational Bounds are Not Necessarily Better. In *International Conference on Machine Learning* 4277–4285.

RASMUSSEN, C. E. (2003). Gaussian processes in machine learning. In *Summer School on Machine Learning* 63–71. Springer.

REZENDE, D. J., MOHAMED, S. and WIERSTRA, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning* 1278–1286.

REZENDE, D. and MOHAMED, S. (2015). Variational Inference with Normalizing Flows. In *International Conference on Machine Learning* 1530–1538.

RICHARDSON, S. and GILKS, W. R. (1993). Conditional independence models for epidemiological studies with covariate measurement error. *Statistics in Medicine* **12** 1703-1722.

ROEDER, G., WU, Y. and DUVENAUD, D. K. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems* 6925–6934.

ROLNICK, D. and TEGMARK, M. (2018). The power of deeper networks for expressing natural functions. In *International Conference on Learning Representations*.

RUPPERT, D., WAND, M. P. and CARROLL, R. J. (2003). *Semiparametric Regression*. *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press.

SCHENNACH, S. M. (2007). Instrumental Variable Estimation of Nonlinear Errors-in-Variables Models. *Econometrica* **75** 201–239.

SCHMIDT-HIEBER, J. et al. (2020). Nonparametric regression using deep neural networks with ReLU activation function. *Annals of Statistics* **48** 1875–1897.

SOUDRY, D., HOFFER, E., NACSON, M. S., GUNASEKAR, S. and SREBRO, N. (2018). The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research* **19** 2822–2878.

STAUDENMAYER, J. and RUPPERT, D. (2004). Local polynomial regression and simulation-extrapolation. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **66** 17–30.

STEFANSKI, L. A. and CARROLL, R. J. (1990). Deconvolving kernel density estimators. *Statistics* **21** 169–184.

TABAK, E. G. and TURNER, C. V. (2013). A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics* **66** 145–164.

TAUPIN, M.-L. (2001). Semi-parametric estimation in the nonlinear structural errors-in-variables model. *The Annals of Statistics* **29** 66–93.

TUCKER, G., LAWSON, D., GU, S. and MADDISON, C. J. (2018). Doubly reparameterized gradient estimators for Monte Carlo objectives. In *International Conference on Learning Representations*.

WILLIAMS, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* **8** 229–256.