Econometric Reviews, 31(1):84–106, 2012

ISSN: 0747-4938 print/1532-4168 online
DOI: 10.1080/07474938.2011.607089

Taylor & Francis
Taylor & Francis Group

ESTIMATION OF DYNAMIC DISCRETE CHOICE MODELS USING ARTIFICIAL NEURAL NETWORK APPROXIMATIONS

Andriy Norets

Department of Economics, Princeton University, Princeton, New Jersey, USA

□ *I propose a method for inference in dynamic discrete choice models (DDCM) that utilizes Markov chain Monte Carlo (MCMC) and artificial neural networks (ANNs). MCMC is intended to handle high-dimensional integration in the likelihood function of richly specified DDCMs. ANNs approximate the dynamic-program (DP) solution as a function of the parameters and state variables prior to estimation to avoid having to solve the DP on each iteration. Potential applications of the proposed methodology include inference in DDCMs with random coefficients, serially correlated unobservables, and dependence across individual observations. The article discusses MCMC estimation of DDCMs, provides relevant background on ANNs, and derives a theoretical justification for the method. Experiments suggest this to be a promising approach.*

Keywords Artificial neural network; Bayesian estimation; Dynamic discrete choice model; Markov chain Monte Carlo.

JEL Classification C11; C35; C45; C63.

1. INTRODUCTION

The dynamic discrete choice model (DDCM) is a dynamic program (DP) with discrete controls. Estimation of these models is a growing area in econometrics with a wide range of applications. Labor economists employed DDCMs in modeling job search and occupational choice (Keane and Wolpin, 1997; Miller, 1984; Wolpin, 1987), retirement decisions (French, 2005; Rust and Phelan, 1997; Stock and Wise, 1990), fertility (Hotz and Miller, 1993; Wolpin, 1984), and crime (Imai and Krishna, 2004). Health economists estimated DDCMs of medical care utilization (Gilleskie, 1998), health and financial decisions of elderly (Davis, 1998), and smoking addiction (Choo, 2000). In industrial organization DDCMs

Address correspondence to Andriy Norets, Department of Economics, Princeton University, Princeton, NJ 08544, USA; E-mail: anorets@princeton.edu

were used for studying optimal investment replacement (Cho, 2011; Das, 1992; Kennet, 1994; Rust, 1987). Pakes (1986) estimated a DDCM of patent renewals. There is a growing interest to DDCMs in marketing literature (Erdem and Keane, 1996; Osborne, 2011).

DDCMs are attractive for empirical research since they are grounded in economic theory. However, estimation of these models is very computationally expensive. The DP has to be solved at each iteration of an estimation procedure and the likelihood function of a richly specified DDCM contains high-dimensional integrals.

Norets (2009) shows that in the Bayesian framework Markov chain Monte Carlo (MCMC) methods can handle the high dimensional integration in the DDCMs' likelihood function. An MCMC estimation procedure for a DDCM may require a lot of iterations for convergence. Thus, the solution of the DP that has to be obtained at each iteration of the estimation procedure constitutes a considerable part of the algorithm's computational costs. Imai et al. (2009) and Norets (2009) proposed methods for solving the DP that use information from the previous MCMC iterations to speed up the DP solution on the current iteration. The approach based on artificial neural networks (ANN) proposed here further reduces the costs of solving the DP in the estimation procedure.

The expected value function can be seen as a function of the parameters and the state variables. Instead of obtaining the DP solution at each iteration of the estimation procedure one could beforehand approximate it by a function of the parameters and states and then use this approximating function in the estimation procedure. Under this approach, there is no need to solve the DP at each iteration of a long posterior simulator run and for each individual in the sample if random/individual specific coefficients are included in the model specification.

Approximating a function of several variables is a formidable task. In the previous literature, some authors, e.g., Ackerberg (2009), Bajari et al. (2010), and Brown and Flinn (2006), proposed to presolve a model at some state space and parameter grid points and use kernel smoothing or similar methods to evaluate the solution of the model at different parameter and state space points during estimation. However, kernel smoothing might not work well in high dimensional problems and it did not perform well in experiments on the model considered in this article, see Norets (2007, Section 1.5.1.4). ANNs seem to be a method of choice for high dimensional approximation problems. An intuitive explanation for excellent performance of ANNs in theory and practice might be that the basis functions in the ANN case can be tuned, which provides additional flexibility relative to other approximation methods, e.g., approximation by polynomials, in which the basis functions are fixed.

An issue that needs to be addressed is whether we can use ANN function approximation properties to show that the estimation results, e.g., posterior

expectations, that are obtained with approximated DP solutions converge to the exact ones as the approximation precision improves. Although there are a lot of different results available for the consistency and convergence rates for ANN function approximation, the result we could use to show the consistency of the estimated posterior expectations does not seem to be available in the ready-to-use form. In this article, I derive such a result (consistency of ANN approximations in the sup norm) from the contributions of White (1990), Hornik et al. (1989), and Chen (2007).

Section 2 of the article sets up a DDCM and outlines an MCMC estimation procedure. Section 3 introduces ANNs and derives necessary theoretical results. A DDCM used for experiments is described in Section 4.1. The corresponding MCMC algorithm is given in Section 4.2. The ANN approximation quality is evaluated in Section 4.3. Section 4.4 presents estimation results.

2. DDCM AND MCMC

A DDCM is a single agent model. Each time period t the agent chooses an alternative d_t from a finite set of available alternatives $D(s_t)$. The per-period utility $u(s_t, d_t; \theta)$ depends on the chosen alternative, current state variables $s_t \in S$, and a vector of parameters $\theta \in \Theta$. The state variables are assumed to evolve according to a controlled first order Markov process with a transition law denoted by $f(s_{t+1} \mid s_t, d_t; \theta)$ for $t \geq 1$; the distribution of the initial state is denoted by $f(s_1 \mid \theta)$. Time is discounted with a factor β. In the recursive formulation of the problem, the lifetime utility of the agent or the value function is given by the maximum of the alternative-specific value functions:

$$V(s_t; \theta) = \max_{d_t \in D(s_t)} \mathcal{V}(s_t, d_t; \theta) \tag{1}$$

$$\mathcal{V}(s_t, d_t; \theta) = u(s_t, d_t; \theta) + \beta E\{V(s_{t+1}; \theta) \mid s_t, d_t; \theta\}. \tag{2}$$

This formulation embraces a finite horizon case if time t is included in the vector of the state variables.

In estimable DDCMs, some extra assumptions are usually made. First of all, some of the state variables are assumed to be unobservable for econometricians (the agent observes s_t at time t.) Let's denote the unobserved state variables by y_t and the observed ones by x_t. Examples of unobservables include taste idiosyncrasy, ability, and health status. Using the unobserved state variables is a way to incorporate random errors in DDCMs structurally. Some of the state variables could be common to all individuals in a dataset. Let's denote these common states by z_t. We assume that z_t are unobserved (the case of observed z_t would be simpler.) To avoid modeling the interactions between agents, it is assumed

that the evolution of z_t is not affected by individual states and decisions. Introducing common states z_t is a way to model dependence across observations in the sample. Thus, the state variables are separated into three parts $s_t = (z_t, x_t, y_t)$, and they evolve according to $f(s_{t+1} \mid s_t, d; \theta) = p(z_{t+1} \mid z_t; \theta)p(x_{t+1}, y_{t+1} \mid x_t, y_t, z_t, d; \theta)$. The set of the available alternatives $D(s_t)$ is assumed to depend only on the observed state variables. Hereafter, it will be denoted by D without loss of generality.

There is a consensus in the literature that it is desirable to allow for individual heterogeneity in panel data models. Examples of individual heterogeneity in DDCMs include individual specific time discount rates and individual specific intercepts or coefficients in the per-period utility function that would represent taste idiosyncrasies. To allow for that, let's assume that the parameter vector θ contains individual specific components θ_1^i and common components θ_2 and the prior $p(\theta_1^i \mid \theta_2)p(\theta_2)$ is specified. The common parameters θ_2 may include components that define $p(\theta_1 \mid \theta_2)$ and do not affect the DP.

A data set that is usually used for the estimation of a dynamic discrete choice model consists of a panel of I individuals. The observed state variables and the decisions are known for each individual $i \in \{1, \ldots, I\}$ for T periods: $(x, d) = \{x_{t,i}, d_{t,i}; t = 1, \ldots, T; i = 1, \ldots, I\}$. The number of time periods T is assumed to be the same for each individual only to simplify the notation. The likelihood function is given by the integral over the latent variables:

$$p(x, d \mid \theta_2) = \int p(x, d, y, \theta_1, z \mid \theta_2) d(y, \theta_1, z), \tag{3}$$

where $y = \{y_{t,i}; t = 1, \ldots, T; i = 1, \ldots, I\}$, $z = \{z_t; t = 1, \ldots, T\}$, and $\theta_1 = \{\theta_1^i; i = 1, \ldots, I\}$. Because of the high dimensionality of the integral computing the likelihood function is infeasible for richly specified DDCMs.

In a Bayesian framework, the high dimensional integration over the latent variables can be handled by employing MCMC for exploring the joint posterior distribution of the latent variables and parameters. As was shown in Norets (2009), it is convenient to use the variables

$$\begin{aligned} \Delta\mathcal{V} = \{\Delta\mathcal{V}_{t,d,i} = u(s_{t,i}, d; \theta) + \beta E[V(s_{t+1}; \theta) \mid s_{t,i}, d; \theta)] \\ - E[V(s_{t+1}; \theta) \mid s_{t,i}, \bar{d}; \theta)], \forall i, t, d\} \end{aligned} \tag{4}$$

as the latent variables in the MCMC algorithm instead of a part of $y_{t,i}$, where $\bar{d}$ is a chosen base alternative. Let's denote the part of $y_{t,i}$ substituted with $\Delta\mathcal{V}_{t,i}$ by $v_{t,i}$ and the remaining part by $\epsilon_{t,i}$; thus $y_{t,i} = (v_{t,i}, \epsilon_{t,i})$. To save space it is assumed below that $p(v_{t,i} \mid z_t, x_{t,i}, \epsilon_{t,i}, z_{t-1}, x_{t-1,i}, \epsilon_{t-1,i}, d_{t-1,i}; \theta^i) = p(v_{t,i} \mid z_t, x_{t,i}, \epsilon_{t,i}; \theta^i)$. However, this assumption is not necessary in what follows.

The joint posterior distribution of the parameters and latent variables will be proportional to the joint distribution of the data, the parameters, and the latent variables

$$p(\theta_1, \theta_2, \Delta\mathcal{V}, \epsilon, z \mid x, d) \propto p(d, \Delta\mathcal{V}, \theta_1, \theta_2, \epsilon, z, x),$$

which in turn can be decomposed into the product of marginals and conditionals:

$$\begin{aligned} p(d, \Delta\mathcal{V}, \theta_1, \theta_2, \epsilon, z, x) = \prod_{t=1}^{T} \Bigg[\prod_{i=1}^{I} \big(& p(d_{t,i} \mid \Delta\mathcal{V}_{t,i}) p(\Delta\mathcal{V}_{t,i} \mid x_{t,i}, \epsilon_{t,i}, z_t; \theta_1^i, \theta_2) \\ & \cdot p(x_{t,i}, \epsilon_{t,i} \mid x_{t-1,i}, \epsilon_{t-1,i}, z_{t-1}, d_{t-1,i}; \theta_1^i, \theta_2) \big) \\ & \cdot p(z_t \mid z_{t-1}, \theta_2) \Bigg] \cdot \Bigg[\prod_{i=1}^{I} p(\theta_1^i \mid \theta_2) \Bigg] \cdot p(\theta_2). \end{aligned} \tag{5}$$

The Gibbs sampler can be used to simulate a Markov chain which would have the stationary distribution equal to the posterior. The densities of the Gibbs sampler blocks: $p(\theta_1^i \mid \Delta\mathcal{V}_i, \theta_2, \epsilon_i, z, d_i, x_i)$, $p(\theta_2 \mid \Delta\mathcal{V}, \epsilon, z, d, x)$, $p(\Delta\mathcal{V}_{t,i} \mid \theta_1^i, \theta_2, \epsilon_{t,i}, z_{t,i}, d_{t,i}, x_{t,i})$, $p(\epsilon_{t,i} \mid \Delta\mathcal{V}_{t,i}, \theta_1^i, \theta_2, \epsilon_{t-1,i}, \epsilon_{t+1,i}, z_{t,i}, d_{t,i}, x_{t,i})$, and $p(z_t \mid \Delta\mathcal{V}_t, \theta_1, \theta_2, \epsilon_t, z_{t+1}, z_{t-1}, d_t, x_t)$ are proportional to (5). If $p(\Delta\mathcal{V}_{t,i} \mid \theta, x_{t,i}, \epsilon_{t,i}, z_t)$ can be quickly computed, then (5) (and, thus, the kernels of the densities of the Gibbs sampler blocks) can be quickly computed as well. Therefore, it is possible to use the Metropolis-within-Gibbs algorithm to simulate the chain. An example of the MCMC algorithm for a specific model is presented in Section 4.2.

As evident from (4), computing the value of the joint density (5) will require computing the differences in expected value functions $F(s, d, \theta) = E[V(s_{t+1}; \theta) \mid s, d; \theta)] - E[V(s_{t+1}; \theta) \mid s, d; \theta)]$. Let $F(s, \theta) = \{F(s, d, \theta), d \in D\}$ be a vector of the differences in expected value functions corresponding to all available alternatives, the same current state s, and the parameter vector θ. Solving the DP and computing $F(s, \theta_1^i, \theta_2)$ for each observation $i = 1, \ldots, I$ at each MCMC iteration would be infeasible. Instead, ANNs can be used to approximate $F(.)$ beforehand of the estimation procedure. The following sections explore this approach.

3. FEEDFORWARD ANN

3.1. Definition of Feedforward ANN

It is beyond the scope of this article to survey the literature on artificial neural networks and their (potential) applications in economics. For general information, history, and econometric perspective on ANN, the reader is referred to the work by Kuan and White (1994). Rust (1996)

discusses the application of neural networks to function approximation problems in the context of numerical dynamic programming. Cho and Sargent (1996) consider applications of neural networks in dynamic economics and game theory.

The purpose of this section is to provide information on artificial neural networks relevant to applications in DDCM estimation. The section describes a particular type of artificial neural networks, feedforward networks (FFANN), that are well suited for function approximation problems. Figure 1 shows the structure of a multilayer FFANN that transforms the input vector $x \in R^n$ into the output vector $y \in R^m$. The network consists from a number of nodes called neurons. The neurons are grouped into layers. The outputs of the neurons on the layer $i-1$ are used as the inputs for each neuron on the next layer i. The inputs for the first level are the network inputs x. The outputs of the last layer are the network outputs. The neuron j on the level i multiplies the inputs $y^{i-1} = (y_1^{i-1}, \ldots, y_{N_{i-1}}^{i-1})$ by the connection weights $w^{ij} = (w_l^{ij}, \ldots, w_{N_{i-1}}^{ij})^T$ and transforms the sum of the weighted inputs into a scalar output y_j^i:

$$y_j^i = f\left(\sum_{l=1}^{N_{i-1}} y_l^{i-1} w_l^{ij}\right) = f(y^{i-1} w^{ij}), \quad i = 1, \ldots, k, \;\; j = 1, \ldots, N_i, \tag{6}$$

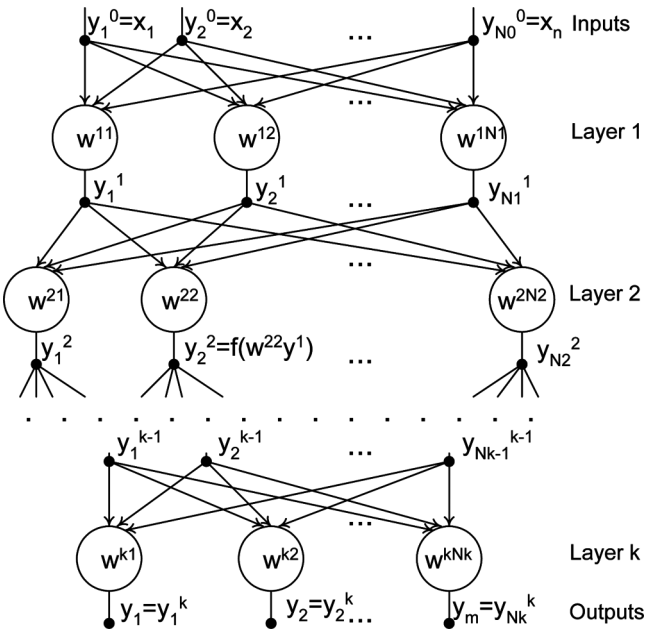

FIGURE 1 Multilayer feedforward neural network.

where k is the number of layers, N_i is the number of neurons in the layer i, and $f(.)$ is called activation function. The logistic sigmoid $f(z) = 1/(1 + \exp\{-z\})$ is a popular choice for the activation function. The activation functions do not have to be the same for all neurons. The identity function $f(z) = z$ is sometimes used for the neurons on the last (output) layer. It is standard practice to add an extra input equal to 1 to each neuron. This is a way to introduce intercepts (called *biases* in the ANN literature) in addition to the coefficients (weights) in (6).

An explicit formula for computing the output for a two-layer network with one-dimensional output might be helpful for understanding the general case:

$$y = y_1^2 = \widehat{F}(x; w) = f\left(\sum_{l=1}^{N_1} w_l^{21} f\left(\sum_{j=1}^{m} w_j^{1l} x_j\right)\right).$$

Let $F(x)$ denote the function we wish to approximate by a neural network. The connection weights w are adjusted so that the neural network $\widehat{F}(x; w)$ fits $F(x)$. The process of adjusting the weights is called learning or training. Training is performed on a dataset $\{x^j, y^j, j = 1, J\}$, where y^j is equal to $F(x^j)$ perhaps with some noise. The method of least squares is the most common way to adjust the weights:

$$\min_w S(w) = \min_w \sum_j [y^j - \widehat{F}(x^j; w)]^2 = \min_w \sum_j e^j(w)^2.$$

If the activation function is differentiable then gradient methods can be used to perform the minimization. In the ANN literature the gradient descent algorithm is referred to as the back error propagation. The derivatives are computed by the chain rule: $S'(w) = 2e'(w)^T e(w)$. More sophisticated optimization methods, such as conjugate gradient algorithms and quasi-Newton methods, can be used to increase the training speed. According to the Matlab manual, the Levenberg–Marquardt algorithm is the fastest method for training moderate-sized FFANN (up to several hundred weights). My experiments with a few other algorithms (not implemented in Matlab) confirm this claim. The Levenberg–Marquardt algorithm iteratively updates the weights according to

$$w^{q+1} = w^q - [e'(w^q)^T e'(w^q) + \mu I]^{-1} e'(w^q)^T e(w^q).$$

If the scalar μ is small then the method works as a quasi-Newton method with the Hessian approximated by $e'(w)^T e'(w)$ (computing actual Hessian would be very time consuming.) If μ is large then the method works as the gradient descent algorithm with a small step. The Newton method

performs considerably better than the gradient descent algorithm near the optimum. Thus, after successful iterations (the ones that decrease $S(w)$) μ is decreased; otherwise it is increased. For large FFANNs conjugate gradient methods might perform better than the Levenberg–Marquardt algorithm.

The training methods mentioned above are local optimization methods. There is no guarantee that they will find the global minimum. The theoretical results on the consistency of ANN approximations do require finding the global minimum. Therefore, running training algorithms for several initial values is advisable. In experiments on approximating the expected value function by a FFANN the Matlab implementation of the Levenberg–Marquardt algorithm performed very well. No cases of getting stuck in a very bad local minimum were detected.

3.2. Consistency of FFANN Approximations

The consistency and convergence rates for ANN function approximation were examined by a number of authors. However, the result we would need for approximation of the DDCM solutions does not seem to be available in the literature in the ready-to-use form. In this section, we deduce the necessary result building on the existing theory of ANN approximation.

The proof of Theorem 1.3 in Norets (2007) implies that the uniform (in sup norm) convergence in probability of the expected value function approximation would imply the consistency of the approximated posterior expectations. It was also shown in Norets (2007) that the expected value function is continuous in the state variables and parameters under suitable continuity and compactness assumptions on the primitives of DDCMs. At the same time, the differentiability of the expected value function with respect to the state variables and parameters does not seem to have been established for a general DDCM. Therefore, it would be desirable to have the consistency of the ANN approximations in sup norm for continuous functions on compact spaces. However, the consistency results in the ANN literature are available for convergence in L_p norm and/or for smooth functions (Barron, 1994; Chen and White, 1999; White, 1990). The required consistency result is derived below from the contributions of White (1990), Hornik et al. (1989), and Chen (2007).

Following White (1990) let's consider a FFANN sieve estimator. For a survey of sieve estimation see Chen (2007). Let $\mathcal{F}$ denote a set of continuous functions on a compact space $\mathcal{X}$ with sup norm $\|.\|$. Let (X, Y) be random variables defined on a complete probability space. Assume that X has a positive density on $\mathcal{X}$, $E(Y \mid X = x) = F(x)$, and $F \in \mathcal{F}$. Let $\{x^j, y^j\}_{j=1}^n$ denote an i.i.d. sample of (X, Y). The consistency results are

proven below for randomly generated $\{x^j\}_{j=1}^n$. Experiments show that using low discrepancy sequences on $\mathscr{X}$ instead of randomly generated ones does not improve the approximation quality.

A FFANN with one hidden layer consisting of q neurons and a linear activation function for the neuron on the output layer is described by

$$\widehat{F}(x; w, q) = w_0^{21} + \sum_{k=1}^{q} w_k^{21} f\left(w_0^{1k} + \sum_j w_j^{1k} x_j\right).$$

Let

$$T(q, \Delta) = \left\{\widehat{F}(.; w, q) : \sum_{k=0}^{q} |w_k^{21}| < \Delta \text{ and } \sum_{k=1}^{q} \sum_j |w_j^{1k}| < q\Delta\right\}$$

be a set of FFANNs with q neurons on the hidden layer and the weights satisfying a restriction on their sum norm. For specified sequences $\{q_n\}$ and $\{\Delta_n\}$, $T(q_n, \Delta_n)$ is called a sieve. The sieve estimator $\widehat{F}_n(.)$ is defined as the solution to the least squares problem:

$$\min_{\widehat{F} \in T(q_n, \Delta_n)} \frac{1}{n} \sum_{j=1}^{n} [y^j - \widehat{F}(x^j)]^2. \tag{7}$$

The parameters q_n and Δ_n determine the flexibility of approximating functions $\widehat{F} \in T(q_n, \Delta_n)$. As they increase to infinity the set $T(q_n, \Delta_n)$ will become dense in $\mathscr{F}$. The flexibility of approximating functions should depend on the number of observations n in such a way that overfitting and underfitting are avoided at the same time. Specific restrictions on q_n and Δ_n that achieve this are given below. Also, introducing a finite bound on the weights Δ_n makes $T(q_n, \Delta_n)$ compact. White (1990) proves a version of the following theorem for L_p norm. Here, I present a proof for sup norm.

Theorem 1. *Assume that the activation function f is Lipschitz continuous and it is a squashing function (f is non-decreasing, $\lim_{x\to-\infty} f(x) = 0$, $\lim_{x\to+\infty} f(x) = 1$). Also assume that $q_n, \Delta_n \nearrow \infty$, $\Delta_n = o(n^{1/4})$, and $\Delta_n^4 q_n \log(\Delta_n q_n) = o(n)$. Under these conditions there exists a measurable sieve estimator $\widehat{F}_n(.)$ defined by* (7) *and for any $\epsilon > 0$*

$$\lim_{n\to\infty} P(\|F - \widehat{F}_n\| > \epsilon) = 0.$$

Proof. Theorem 3.1 in Chen (2007) specifies five conditions under which an abstract extremum sieve estimator will be consistent. Let's show that these five conditions are satisfied.

Condition 3.1: $E[Y - g(X)]^2$ is uniquely minimized over $g \in \mathcal{F}$ at F and $E[Y - F(X)]^2 < \infty$. This identification condition is satisfied in our case because $F(x) \equiv E(Y \mid X = x)$ is a minimizer and it is unique since functions in $\mathcal{F}$ are continuous and the density of X is positive on $\mathcal{X}$.

Condition 3.2: The sequence of sieves is increasing ($T(q_n, \Delta_n) \subset T(q_{n+1}, \Delta_{n+1})$) and $\bigcup_{n=1}^{\infty} T(q_n, \Delta_n)$ is dense in $\mathcal{F}$. The denseness of the set of one hidden layer FFANNs with unconstrained weights $\bigcup_{n=1}^{\infty} T(n, \infty)$ in the set of continuous functions on compacts is proven in Hornik et al. (1989), Theorem 2.4. The condition is satisfied since $\cup_{n=1}^{\infty} T(q_n, \Delta_n) = \bigcup_{n=1}^{\infty} T(q_n, \infty)$ for $q_n, \Delta_n \to \infty$.

Condition 3.3: $-E[Y - g(X)]^2$ is upper semicontinuous in g w.r.t $\|.\|$. The condition is trivially satisfied since $E[Y - g(X)]^2$ is continuous.

Condition 3.4: $T(q_n, \Delta_n)$ is compact under $\|.\|$. Since any element in $T(q_n, \Delta_n)$ is defined by a vector of weights belonging to a compact set and the activation function f is continuous, any sequence in $T(q_n, \Delta_n)$ will have a convergent subsequence with the limit in $T(q_n, \Delta_n)$, thus $T(q_n, \Delta_n)$ is compact.

Condition 3.5: (Uniform convergence over sieves) $\operatorname{plim}_{n\to\infty} \sup_{g \in T(q_n, \Delta_n)} | \frac{1}{n} \sum_{j=1}^{n} [y^j - g(x^j)]^2 - E[Y - g(X)]^2 | = 0$. This condition is proven in White (1990, pp. 543–544). That is where the Lipschitz continuity of f and the specific conditions on q_n and Δ_n are used. □

4. EXPERIMENTS

Experiments in this section demonstrate how well FFANNs can approximate expected value functions and what the performance gains of using FFANNs in MCMC estimation of DDCMs can be. The Rust (1987) model of optimal bus engine replacement with added serially correlated unobservables is used for experiments.

4.1. Rust's (1987) Model

In Rust's model, a maintenance superintendent of a bus transportation company decides every time period whether to replace a bus engine. The observed state variable is the bus mileage x_t since the last engine replacement. The control variable d_t takes on two values: 2 if the engine is replaced at t, and 1 otherwise. The per-period utility function of the superintendent is the negative of per-period costs

$$u(x_t, \epsilon_t, v_t, d_t; \alpha) = \begin{cases} \alpha_1 x_t + \epsilon_t & \text{if } d_t = 1 \\ \alpha_2 + v_t & \text{if } d_t = 2, \end{cases} \tag{8}$$

where ϵ_t and v_t are the unobserved state variables, α_1 is the negative of per-period maintenance costs per unit of mileage, α_2 is the negative of the costs of engine replacement. Rust assumes that ϵ_t and v_t are extreme value independently identically distributed (IID). I assume v_t is IID $N(0, h_v^{-1})$ truncated to $[-\bar{v}, \bar{v}]$, ϵ_t is $N(\rho\epsilon_{t-1}, h_\epsilon^{-1})$ truncated to $E = [-\bar{\epsilon}, \bar{\epsilon}]$, and $\epsilon_0 = 0$. The bus mileage since the last replacement is discretized into $M = 90$ intervals $X = \{1, \ldots, M\}$. The observed state x_t evolves according to

$$P(x_{t+1} \mid x_t, d_t; \eta) = \begin{cases} \pi(x_{t+1} - x_t; \eta) & \text{if } d_t = 1 \\ \pi(x_{t+1} - 1; \eta) & \text{if } d_t = 2 \end{cases} \tag{9}$$

and

$$\pi(\Delta x; \eta) = \begin{cases} \eta_1 & \text{if } \Delta x = 0 \\ \eta_2 & \text{if } \Delta x = 1 \\ \eta_3 & \text{if } \Delta x = 2 \\ 0 & \text{if } \Delta x \geq 3. \end{cases} \tag{10}$$

Rust assumes that if the mileage reaches the state M it stays in this state with probability 1. I instead assume that the engine is replaced at t if x_t exceeds $M - 1$, which slightly simplifies the DP solution. In the recursive formulation, the life-time utility for $x_t < M$ is given by

$$V(x_t, \epsilon_t, v_t; \theta) = \max \Bigg\{ \alpha_1 x_t + \epsilon_t + \beta \sum_{k=1}^{3} \eta_k E[V(x_t + k - 1, \epsilon', v'; \theta) \mid \epsilon_t; \theta], \\ \alpha_2 + v_t + \beta EV_2(\theta) \Bigg\}, \tag{11}$$

where

$$EV_2(\theta) = \sum_{k=1}^{3} \eta_k E[V(k, \epsilon', v'; \theta) \mid 0; \theta] \tag{12}$$

$$E[V(x_{t+1}, \epsilon', v'; \theta) \mid \epsilon_t; \theta] = \int V(x_{t+1}, \epsilon', v'; \theta) dP(\epsilon', v' \mid \epsilon_t; \theta). \tag{13}$$

For $x_t \geq M$,

$$V(x_t, \epsilon_t, v_t; \theta) = \alpha_2 + v_t + \beta EV_2(\theta). \tag{14}$$

4.2. Gibbs Sampler

Each bus i is observed over T_i time periods: $\{x_{t,i}, d_{t,i}\}_{t=1}^{T_i}$ for $i = 1, \ldots, I$. The parameters are $\theta = (\alpha, \eta, \rho)$; h_ϵ and h_v are fixed for normalization.

The latent variables are $\{\Delta\mathscr{V}_{t,i}, \epsilon_{t,i}\}_{t=1}^{T_i}$ $i = 1, \ldots, I$

$$\Delta\mathscr{V}_{t,i} = x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} - v_{t,i} + F_{t,i}(\theta, \epsilon_{t,i}),$$

where

$$\begin{aligned} F(x_{t,i}, \epsilon, \theta) &= F_{t,i}(\theta, \epsilon) \\ &= \beta \sum_{j=1}^{3} \eta_j (E[V(x_{t,i} + j - 1, \epsilon', v'; \theta) \mid \epsilon; \theta] - EV_2(\theta)). \end{aligned} \tag{15}$$

The compact space for parameters Θ is defined as follows: $\alpha_i \in [-\bar{\alpha}, \bar{\alpha}]$, $\rho \in [-\bar{\rho}, \bar{\rho}]$, $h_\epsilon \in [\underline{h_\epsilon^l}, \overline{h_\epsilon^r}]$, and η belongs to a three dimensional simplex. The joint distribution of the data, the parameters, and the latent variables is

$$\begin{aligned} &p(\theta; \{x_{t,i}, d_{t,i}; \Delta\mathscr{V}_{t,i}, \epsilon_{t,i}\}_{t=1}^{T_i}; i = 1, \ldots, I) \\ &= p(\theta) \prod_{i=1}^{I} \prod_{t=1}^{T_i} [p(d_{t,i} \mid \Delta\mathscr{V}_{t,i}) p(\Delta\mathscr{V}_{t,i} \mid x_{t,i}, \epsilon_{t,i}; \theta) \\ &\quad \cdot p(x_{t,i} \mid x_{t-1,i}; d_{t-1,i}; \eta) p(\epsilon_{t,i} \mid \epsilon_{t-1,i}, \rho, h_\epsilon)], \end{aligned}$$

where $p(\theta)$ is a prior density for the parameters; $p(x_{t,i} \mid x_{t-1,i}; d_{t-1,i}; \eta)$ is given in (9) and

$p(x_{1,i} \mid x_{0,i}; d_{0,i}; \eta) = 1_{\{1\}}(x_{1,i})$—all the buses start with a new engine;

$$p(d_{t,i} \mid \Delta\mathscr{V}_{t,i}) = \begin{cases} 1, & \text{if } d_{t,i} = 1, \Delta\mathscr{V}_{t,i} \geq 0 \\ 0, & \text{if } d_{t,i} = 1, \Delta\mathscr{V}_{t,i} < 0 \\ 1, & \text{if } d_{t,i} = 2, \Delta\mathscr{V}_{t,i} \leq 0 \\ 0, & \text{if } d_{t,i} = 2, \Delta\mathscr{V}_{t,i} > 0 \end{cases} \tag{16}$$

$$p(\Delta\mathscr{V}_{t,i} \mid x_{t,i}, \epsilon_{t,i}; \theta) = \exp\{-0.5 h_v (\Delta\mathscr{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})])^2\} \tag{17}$$

$$\cdot 1_{[-\bar{v}, \bar{v}]}(\Delta\mathscr{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})]) \tag{18}$$

$$\cdot \frac{h_v^{0.5}}{\sqrt{2\pi}[\Phi(\bar{v} h_v^{0.5}) - \Phi(-\bar{v} h_v^{0.5})]},$$

$$p(\epsilon_{t,i} \mid \epsilon_{t-1,i}, \theta) = \frac{h_\epsilon^{1/2} \exp\{-0.5 h_\epsilon (\epsilon_{t,i} - \rho\epsilon_{t-1,i})^2\}}{\sqrt{2\pi}[\Phi([\bar{\epsilon} - \rho\epsilon_{t-1,i}] h_\epsilon^{0.5}) - \Phi([-\bar{\epsilon} - \rho\epsilon_{t-1,i}] h_\epsilon^{0.5})]} 1_E(\epsilon_{t,i}). \tag{19}$$

Gibbs Sampler Blocks

The Gibbs sampler blocks for $\Delta\mathcal{V}_{t,i} \mid \ldots$ will have a normal truncated distribution proportional to (17) and (18), and also truncated to R^+ if $d_{t,i} = 1$ or to R^- otherwise. An algorithm from Geweke (1991) is used to simulate efficiently from the normal distribution truncated to R^+ (or R^-.) Acceptance sampling handles the truncation in (18).

The density for $\epsilon_{t,i} \mid \ldots$ is proportional to

$$
\begin{aligned}
p(\epsilon_{t,i} \mid \ldots) \propto & \frac{\exp\{-0.5h_v(\Delta\mathcal{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})])^2\}}{\Phi([\bar{\epsilon} - \rho\epsilon_{t-1,i}]h_\epsilon^{0.5}) - \Phi([-\bar{\epsilon} - \rho\epsilon_{t-1,i}]h_\epsilon^{0.5})} \\
& \cdot 1_{[-\bar{v},\bar{v}]}(\Delta\mathcal{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})]) \\
& \cdot \exp\{-0.5h_\epsilon(\epsilon_{t+1,i} - \rho\epsilon_{t,i})^2 - 0.5h_\epsilon(\epsilon_{t,i} - \rho\epsilon_{t-1,i})^2\} \cdot 1_E(\epsilon_{t,i}).
\end{aligned} \tag{20}
$$

Draws from this density are obtained from a Metropolis step with a normal truncated transition density proportional to (20). The blocks for $\epsilon_{t,i}$ with $t = 0$ and $t = T_i$ will be similar.

Assuming a normal prior $N(\underline{\rho}, \underline{h}_\rho)$ truncated to $[-\bar{\rho}, \bar{\rho}]$,

$$
\begin{aligned}
p(\rho \mid \ldots) \propto & \frac{\exp\{-0.5h_v \sum_{i,t}(\Delta\mathcal{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})])^2\}}{\prod_{i,t}\Phi([\bar{\epsilon} - \rho\epsilon_{t-1,i}]h_\epsilon^{0.5}) - \Phi([-\bar{\epsilon} - \rho\epsilon_{t-1,i}]h_\epsilon^{0.5})} \\
& \cdot \prod_{i,t} 1_{[-\bar{v},\bar{v}]}(\Delta\mathcal{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})]) \\
& \cdot \exp\{-0.5\bar{h}_\rho(\rho - \bar{\rho})^2\} \cdot 1_{[-\bar{\rho},\bar{\rho}]}(\rho),
\end{aligned} \tag{21}
$$

where $\bar{h}_\rho = \underline{h}_\rho + h_\epsilon \sum_i \sum_{t=2}^{T_i} \epsilon_{t-1,i}^2$ and $\bar{\rho} = \bar{h}_\rho^{-1}(\underline{h}_\rho\underline{\rho} + h_\epsilon \sum_i \sum_{t=2}^{T_i} \epsilon_{t,i}\epsilon_{t-1,i})$. To draw from this density, I use a Metropolis step with a normal truncated transition density proportional to (21).

Assuming a Dirichlet prior with parameters (a_1, a_2, a_3),

$$
\begin{aligned}
p(\eta \mid \ldots) \propto & \exp\left\{-0.5h_v \sum_{i,t}(\Delta\mathcal{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})])^2\right\} \\
& \cdot \prod_{i,t} 1_{[-\bar{v},\bar{v}]}(\Delta\mathcal{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})]) \\
& \cdot \prod_{j=1}^{3} \eta_j^{n_j + a_j - 1},
\end{aligned} \tag{22}
$$

where $n_j = \sum_i \sum_{t=2}^{T_i} 1_{\{j-1\}}(x_{t,i} - x_{t-1,i})$. A Metropolis step with a Dirichlet transition density proportional to (22) is used in this block.

$$p(\alpha \mid \dots) \propto p(\alpha) \exp\left\{ -0.5 h_v \sum_{i,t} (\Delta \mathcal{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})])^2 \right\}$$
$$\cdot 1_{[-\bar{\alpha},\bar{\alpha}]\times[-\bar{\alpha},\bar{\alpha}]}(\alpha) \cdot \prod_{i,t} 1_{[-\bar{v},\bar{v}]}(\Delta \mathcal{V}_{t,i} - [x_{t,i}\alpha_1 - \alpha_2 + \epsilon_{t,i} + F_{t,i}(\theta, \epsilon_{t,i})]).$$

To draw from this density, I use the Metropolis–Hastings random walk algorithm. The proposal density is normal truncated to $[-\bar{\alpha}, \bar{\alpha}] \times [-\bar{\alpha}, \bar{\alpha}]$ with a mean equal to the current parameter draw and a fixed variance. The variance matrix is chosen so that the acceptance probability would be between 0.2–0.3.

4.3. Evaluating ANN Approximation Quality

The posterior simulator outlined above requires computing the differences in expected value functions $F(x_{t,i}, \epsilon_{t,i}^m, \theta^m)$ defined in (15) for each parameter draw θ^m and each observation (i, t) in the sample. This section shows how FFANNs can be used for approximating $F(.)$.

A FFANN is trained and validated beforehand of the estimation procedure on a sample of inputs and outputs. The inputs include parameters and states:

$\{x^{ji}, \epsilon^j, \alpha_1^j, \alpha_2^j, \rho^j, \eta_1^j, \eta_2^j, \eta_3^j; i = 1, \dots, 90; j = 1, \dots, 2200\}$. The sample of inputs is generated randomly from a distribution on the state and parameter space. The support of this distribution should include all the relevant parts of the state and parameters space. At the same time, it should not be unnecessarily large. This can be achieved with the following procedure. First, estimate the posterior distribution for the model (or its simplified version, e.g., model without heterogeneity or serial correlation) using a very crude but fast DP solution method, e.g., DP solution on a small random grid.[1] Generate the inputs from a distribution with the support that includes the support of the estimated posterior. Train a FFANN on this inputs and corresponding outputs (outputs computation is discussed below). For a prior distribution with the same support as the distribution used for generating the inputs, estimate the posterior using the FFANN. If the posterior puts considerable weight on the regions near the boundary

[1]By solution of the DP on a random grid I mean solution of the DP by the value function iteration method obtained on a discretized state space, where the discretization points are chosen randomly rather than deterministically. Rust (1997) shows that this type of algorithms breaks the curse of dimensionality for DDCMs. Norets (2008) discusses this solution method and possible improvements in detail in the context of the bus engine replacement problem with serially correlated unobservables.

of the prior support, then the procedure should be repeated with a larger support for the distribution of inputs and the prior. For the model used in this article repeating the procedure was not necessary.

In the experiments described below, the inputs are generated from the following distributions: $\alpha_1^j \sim U[-0.006, 0]$, $\alpha_2^j \sim U[-25, -5]$, $\rho^j \sim U[0, 0.99]$, $\epsilon^j \sim U[-3.8, 3.8]$, $\eta \sim \text{Dirichlet}(34, 64, 2)$, and $x^{ji} = i$. Using low discrepancy sequences instead of random draws did not affect the results. For large and complicated models, more involved inputs could be used, e.g., some functions of states and parameters, and the value functions for the DP in which shocks are replaced by zeros. The latter could also be subtracted from the difference in expected value function to obtain a better behaved output.

Since the exact DP solution is not available for the model with serially correlated unobservables the following approximations are used as etalon outputs. For each θ^j the DP is solved on $\widetilde{N} = 100$ different random grids. Each grid consists of $\widehat{N} = 100$ randomly generated points on the space for v and ϵ. The differences in the expected value functions are computed for each random grid. The average over the grids, denoted by $F^{ji}_{\widetilde{N},\widehat{N}}$, is used as the etalon output. This procedure efficiently produces good approximations of $F(.)$. Let's illustrate this for the model with extreme value IID unobservables ϵ_t and v_t.

Under the extreme value IID assumption, the integration over the unobservables can be performed analytically (see Rust, 1994), the exact DP solution can be quickly computed, and solutions on the random grids can be compared with the exact solution. Figure 2 shows densities of the scaled difference between the exact solution and the solution on random

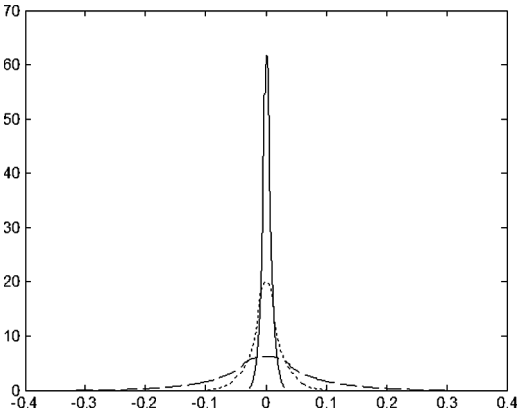

FIGURE 2 Densities of the difference between the exact solution and the solution on random grids. The model with extreme value IID unobservables. The dashed line is the density of $[F(x^{ji}, \theta^j) - F^{ji}_{1,100}]h_v^{0.5}$, the dotted line is the density of $[F(x^{ji}, \theta^j) - F^{ji}_{10,100}]h_v^{0.5}$, and the solid line is the density of $[F(x^{ji}, \theta^j) - F^{ji}_{100,100}]h_v^{0.5}$.

grids $[F(x^{ji}, \theta^j) - F^{ji}_{\tilde{N},\hat{N}}]h_v^{0.5}$ (for IID unobservables $F(.)$ does not depend on ϵ). Scaling is performed to facilitate the comparison of the approximation error and the magnitude of the random shocks in the model. The densities were estimated by kernel smoothing.

The precision of the DP solution obtained by averaging the results over 100 random grids with 100 points in each grid is about the same as for the solution obtained on one random grid with 10000 points. However, the former algorithm works about 100 times faster since the number of operations performed for one Bellman equation iteration is roughly proportional to the square of the number of points in the grid. See Section 1.5.1.4 of Norets (2007) for further discussion of this issue. The maximal approximation error for $F^{ji}_{100,100}$ does not exceed 3% of the standard deviation of the IID shocks in the model.

Having illustrated the high quality of data used for training ANNs let's look at the quality of ANN approximations. The experiments below were conducted on a three layer FFANN that was trained in Matlab by the Levenberg–Marquardt algorithm. The network layers contained 8, 10, and 1 neurons correspondingly (it will be referred below as the 8-10-1 FFANN). First 1500 points in the data were used for training, the remaining data were used for validation.

For the first experiment, the data were generated from the model with Gaussian serially correlated unobservables. Figure 3 shows the distributions of the residuals scaled by the standard deviation of the IID shocks in the model $e^{ji} = (F^{ij}_{100,100} - \widehat{F}(x^{ji}, \epsilon^j, \theta^j; w))h_v^{0.5}$ for the training and validation parts of the data.

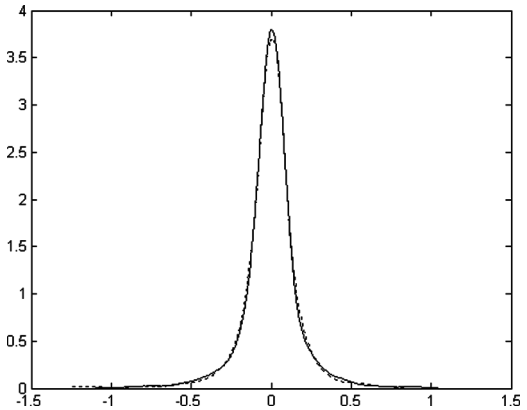

FIGURE 3 Densities of residuals e^{ji} (the dotted line is for the validation part of the sample). The model with serially correlated unobservables.

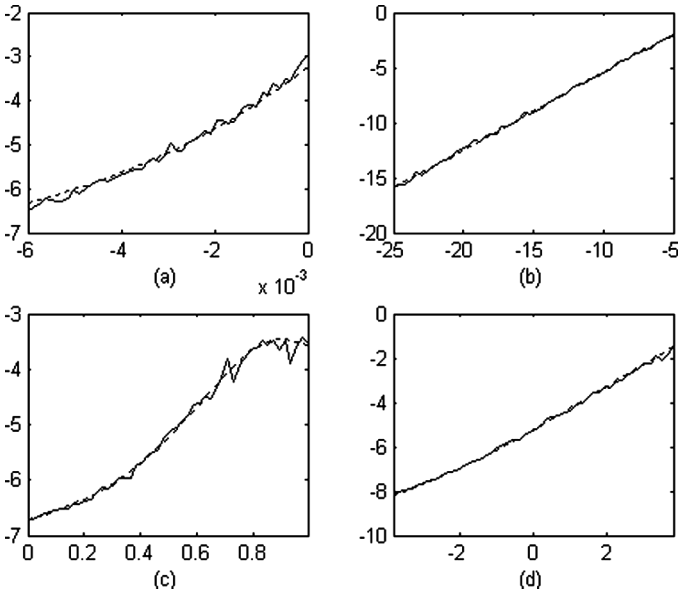

FIGURE 4 Fitted values $\widehat{F}(x^{ji}, \epsilon^j, \theta^j; w)$ (the dotted lines) and $F^{ji}_{100,100}$ (the solid lines) as functions of one input component. The model with serially correlated unobservables. The horizontal axes are: (a) α_1, (b) α_2, (c) ρ, and (d) ϵ.

As can be seen Fig. 3, the approximation quality for the validation part of the data is the same as for the training part. This suggest that no overfitting occurred.

In addition to the randomly generated validation part of the sample, $F^{ji}_{100,100}$ were computed for inputs with one component changing over a relevant range and the other components fixed at $(x, \epsilon, \alpha_1, \alpha_2, \rho, \eta_1, \eta_2, \eta_3) = (55, 0, -0.003, -10, 0.5, 0.34, 0.64, 0.02)$. Figure 4 shows these $F^{ji}_{100,100}$ and the corresponding fitted values.

As Figs. 4 and 2 demonstrate, the values $F^{ji}_{100,100}$ used for neural network training are noisy approximations to the exact differences in expected value functions. It is not surprising since they were obtained by solving the dynamic program on random grids. The exact difference in the expected value functions should be a smooth function. Since the fitted function $\widehat{F}(.; w)$ tends to smooth out the noise, the actual error of the neural network approximation might be smaller on average than the residuals described by Fig. 3.

The quality of FFANN approximations can be further explored for the model with extreme value IID unobservables since the exact approximation error can be computed in this case. Figure 5 compares the densities of the exact approximation error for FFANNs and DP solutions on random grids.

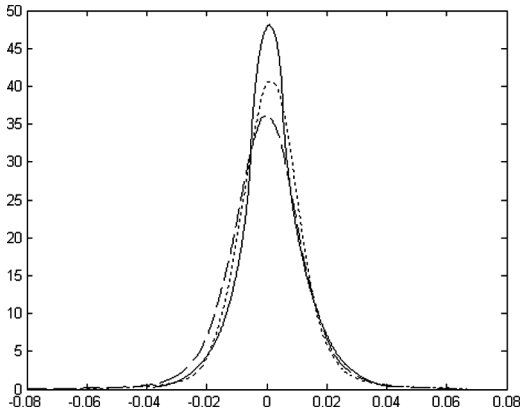

FIGURE 5 Densities of the approximation error for FFANNs and DP solutions on random grids. The model with extreme value IID unobservables. The dashed line is for a FFANN trained on exact data, the dotted line is for a FFANN trained on $F^{ji}_{100,100}$, and the solid line is for $F^{ji}_{100,100}$.

In this particular example, the noise in the training data does not affect the FFANN approximation quality as evidenced by similar results for FFANNs trained on exact and noisy data.

Figure 6 presents a similar comparison of the FFANN and random grid DP solution approximations for the model with serially correlated unobservables. Unfortunately, the exact DP solution and, thus, the exact approximation errors are not available for this model. Therefore, the figure shows the densities of the scaled residuals $e^{ji} = (F^{ji}_{100,100} - \widehat{F}(x^{ji}, \epsilon^j, \theta^j; w))h_v^{0.5}$ for an 8-10-1 FFANN and the scaled differences $[F^{ji}_{100,100} - F^{ji}_{10,100}]h_v^{0.5}$.

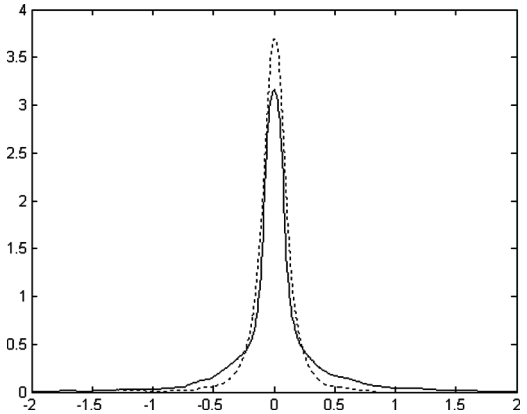

FIGURE 6 Densities of e from the neural network (the dotted line) and from the DP solution on a random grid (the solid line). The model with serially correlated unobservables.

As can be seen from Fig, 6, $\widehat{F}(x^{ji}, \epsilon^j, \theta^j)$ and $F^{ji}_{10,100}$ provide comparable precision in approximating $F^{ji}_{100,100}$. Since the variance of $F(x^{ji}, \epsilon^j, \theta^j) - F^{ji}_{10,100}$ is considerably larger than the variance of $F(\theta^j, x^i) - F^{ji}_{100,100}$ (see Fig. 2), we argue that $\widehat{F}(x^{ji}, \epsilon^j, \theta^j)$ and $F^{ji}_{10,100}$ provide comparable precision in approximating $F(x^{ji}, \epsilon^j, \theta^j)$. Looking back at Fig. 5 we see that for the model with extreme value IID unobservables, the approximation precision of an 8-10-1 FFANN is comparable to the precision of $F^{ji}_{100,100}$, which is better than that of $F^{ji}_{10,100}$ we obtain for the model with serially correlated unobservables. This can be explained by the fact that the dimension of the input vector is smaller for the model with extreme value IID unobservables. Increasing the number of neurons and/or layers in a FFANN improves the precision, e.g., adding another layer with 10 neurons decreased the approximation error by two times on average.

Let's now compare execution times for estimation procedures using FFANNs approximations and DP solutions on random grids. The posterior simulator for the model with serially correlated unobservables that uses the 8-10-1 FFANN works 4–5 times faster than the posterior simulator that uses DP solutions on one random grid with 100 points. Averaging DP solutions over 10 random grids (which would provide precision comparable to the 8-10-1 FFANN as we argued above) will increase the execution time by 10 times. Thus, for this particular example, the performance gains from using FFANNs in the posterior simulator could amount to about 40–50 times. In experiments, the MCMC estimation algorithm required at least 1 million draws to converge. The time required for preparing FFANN training data is less than 2% of the time required for solving the DP on 10 random grids for 1 million parameter draws. The time required for training an 8-10-1 FFANN is also of similar magnitude. Thus the overall time saving from using FFANN approximations in DDCM estimation seems considerable. This claim is confirmed by the performance comparison for the model with extreme value IID unobservables, which is presented with the estimation results in the next section. A reader interested in estimation results for the model with serially correlated unobservables is referred to Norets (2008) (DP solutions on random grids are used in that article).

4.4. Estimation Results

This section presents estimation results for the model with extreme value IID unobservables. The advantage of using this model relative to the model with serially correlated unobservables is that we can characterize the posterior distributions of parameters with a very high precision. The integration over the unobservables in solving the DP and in the likelihood function can be performed analytically. Thus it would be easier to evaluate the quality of the posterior simulator that uses FFANNs. The posterior

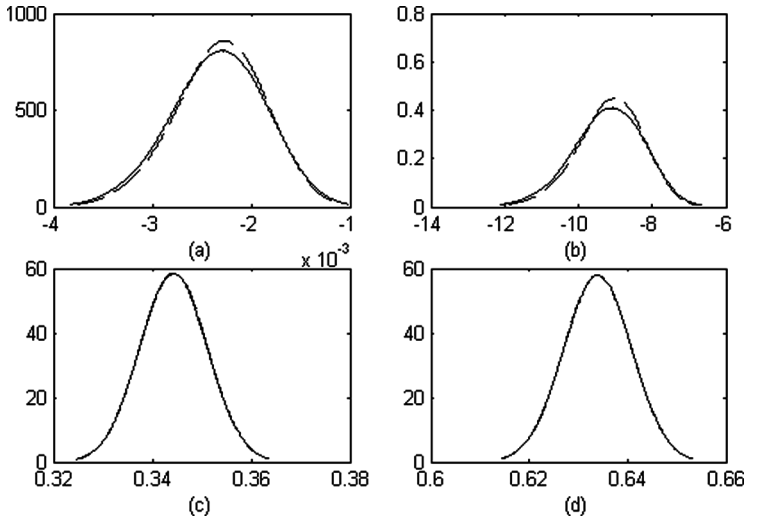

FIGURE 7 Estimated posterior densities: (a) α_1, (b) α_2, (c) η_1, (d) η_2. The solid lines for the algorithm use the exact DP solutions, and the dashed for the algorithm use the FFANN.

simulator for this model also uses the Metropolis–Hastings algorithm since the logit-like choice probabilities comprising the likelihood function contain the expected value functions that do not have an analytical representation.

Figure 7 shows the posterior densities estimated by the posterior simulators that use the exact DP solutions and the 8-10-1 FFANN approximations. The experiments use an artificial dataset consisting of observations on $I = 70$ buses (about 4000 mileage/decision points). The posterior densities were estimated by kernel smoothing over several simulator runs. The length of the runs was 3 million draws. The simulator using the 8-10-1 FFANN takes about 1.2 second to produce 1000 draws from the posterior on a 2002 vintage PC. The simulator that uses the exact DP solutions works 10 times slower. The estimated densities from both simulators are very similar.

The model with extreme value IID unobservables could also be estimated by the algorithm that performs integration numerically as in the case of the model with serially correlated unobservables. The Gibbs sampler for this algorithm is the same as the one for the Gaussian unobservables described in Section 4.2; except here the Gaussian probability densities are replaced by the densities for the extreme value distribution.

Figure 8 compares the estimation results for the exact posterior simulator and the simulator that integrates unobservables numerically and solves the DP on a random grid with 100 points. The posteriors for η are not shown in the figure since they are identical for all simulators.

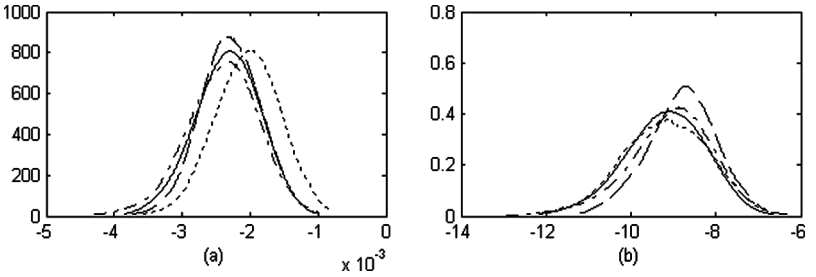

FIGURE 8 Estimated posterior densities: (a) α_1, (b) α_2. The solid lines are the simulator using the exact DP solutions, the other lines are the simulators using DP solutions on different random grids.

For some random grids the estimated density can be far off as the figure demonstrates. The simulator that integrates unobservables numerically and solves the DP on a random grid with 100 points produce 1000 parameter draws in 102 seconds. The same task takes 14 seconds for the same simulator if it uses an 8-10-1 FFANN instead of DP solutions on a random grid.

As Fig. 5 from the previous section demonstrates, the approximation precision of an 8-10-1 FFANN is comparable to the average over 100 DP solutions on random grids with 100 points. If the simulator uses averages of the DP solutions over several random grids the computing time will increase proportionally to the number of the random grids used. Thus, the performance gains from using FFANNs in this example can reach $729 = (102 \cdot 100/14)$ times. Estimation experiments in Section 1.5.2 of Norets (2007) suggest that averaging the posterior distributions estimated with the DP solved on different random grids improves estimation precision. Nevertheless, this posterior averaging strategy is still considerably outperformed by a simulator using FFANNs. A comparison of FFANNs with other approximation methods, e.g., Smolyak polynomials, in the context of DDCM estimation could be a subject of future work.

In summary, the experiments suggest that application of ANNs in the MCMC estimation of DDCMs is indeed a promising approach. It is fast and precise. It can also provide a feasible way to estimate rich DDCMs with different forms of individual heterogeneity such as serially correlated unobserved state variables and individual specific parameters.

ACKNOWLEDGMENTS

I thank John Geweke, anonymous referee, participants of the Greater NY Metropolitan Area Econometrics Colloquium and CEF2008 for useful comments. All remaining errors are mine.

REFERENCES

Ackerberg, D. (2009). A new use of importance sampling to reduce computational burden in simulation estimation. *Quantitative Marketing and Economics* 7:343–376.

Bajari, P., Hong, H., Ryan, S. (2010). Identification and estimation of a discrete game of complete information. *Econometrica* 78:1529–1568.

Barron, A. (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning* 14:115–133.

Brown, M., Flinn, C. (2006). *Investment in Child Quality Over Marital States*. Mimeo, University of Wisconsin–Madison and New York University.

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

Chen, X. (2007). Large sample sieve estimation of semi-nonparametric models. In Heckman, J., Leamer, E., eds. *Handbook of Econometrics*. Amsterdam: Elsevier.

Choo, E. (2000). Rational addiction and rational cessation: a dynamic structural model of cigarette consumption. Unpublished manuscript, University of Calgary.

Cho, I.-K., Sargent, T. (1996). Neural networks for encoding and adapting in dynamic economies. In: Amman, D. K. H., Rust, J. *Handbook of Computational Economics*. Amsterdam: North Holland.

Cho, S.-J. (2011). An empirical model of mainframe computer replacement. *Journal of Applied Econometrics* 26:122–150.

Das, M. (1992). A micro-econometric model of capital utilization and retirement: the case of the cement industry. *Review of Economic Studies* 59:277–297.

Davis, M. A. (1998). The Health and Financial Decisions of the Elderly. Ph.D. thesis, University of Pennsylvania.

Erdem, T., Keane, M. (1996). Decision-making under uncertainty: capturing dynamic brand choice processes in turbulent consumer goods markets. *Marketing Science* 15:1–20.

French, E. (2005). The effects of health, wealth, and wages on labour supply and retirement behaviour. *Review of Economic Studies* 72:395–427.

Geweke, J. (1991). Efficient simulation from the multivariate normal and student-t distributions subject to linear constraints and the evaluation of constraint probabilities. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, E. M. Keramidas, ed., Fairfax, Virginia: Interface Foundation of North America, Inc.

Gilleskie, D. (1998). A dynamic stochastic model of medical care use and work absence. *Econometrica* 6:1–45.

Hornik, K., Stinchcombe, M., White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks* 359–366.

Hotz, J., Miller, R. (1993). Conditional choice probabilities and the estimation of dynamic models. *Review of Economic Studies* 60:497–530.

Imai, S., Jain, N., Ching, A. (2009). Bayesian estimation of dynamic discrete choice models. *Econometrica* 77:1865–1899.

Imai, S., Krishna, K. (2004). Employment, deterrence, and crime in a dynamic model. *International Economic Review* 45:845–872.

Keane, M., Wolpin, K. (1997). The career decisions of young men. *Journal of Political Economy* 105:473–522.

Kennet, M. (1994). A Structural model of aircraft engine maintenance. *Journal of Applied Econometrics* 9:351–368.

Kuan, C.-M., White, H. (1994). Artificial neural networks: an econometric perspective. *Econometric Reviews* 1–92.

Miller, R. (1984). Job matching and occupational choice. *Journal of Political Economy* 1086–1120.

Norets, A. (2007). Bayesian inference in dynamic discrete choice models. PhD Dissertation, The University of Iowa.

Norets, A. (2008). Implementation of bayesian inference in dynamic discrete choice models. Unpublished manuscript, Princeton University.

Norets, A. (2009). Inference in dynamic discrete choice models with serially correlated unobserved state variables. *Econometrica* 77:1665–1682.

Osborne, M. (2011). Consumer learning, habit formation, and heterogeneity: a structural examination. *Quantitative Marketing and Economics* 9:25–70.

Pakes, A. (1986). Patents as options: some estimates of the value of holding european patent stocks. *Econometrica* 54:755–784.

Rust, J., Phelan, C. (1997). How social security and medicare affect retirement behavior in a world of incomplete markets. *Econometrica* 65:781–831.

Rust, J. (1987). Optimal replacement of GMC bus engines: an empirical model of Harold Zurcher. *Econometrica* 55:999–1033.

Rust, J. (1994). Structural estimation of markov decision processes. In Engle, R., McFadden, D., eds. *Handbook of Econometrics*. Amsterdam: North Holland.

Rust, J. (1996). Numerical dynamic programming in economics. In: Amman, D. K. H., Rust, J., eds. *Handbook of Computational Economics*. Amsterdam: North Holland, Available online at http://gemini.econ.umd.edu/jrust/sdp/ndp.pdf.

Rust, J. (1997). Using randomization to break the curse of dimensionality. *Econometrica* 65:487–516.

Stock, J., Wise, D. (1990). The pension inducement to retire: an option value analysis. *NBER Working Papers*, No. 2660.

White, H. (1990). Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings. *Neural Networks* 535–549.

Wolpin, K. I. (1984). An estimable dynamic stochastic model of fertility and child mortality. *The Journal of Political Economy* 92:852–874.

Wolpin, K. (1987). Estimating a structural search model: the transition from school to work. *Econometrica* 55:801–817.