Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Developing a deep learning estimator to learn nonlinear dynamic systems

Kai Wang*, Junghui Chen, Yalin Wang***
** School of Automation, Central South University, Changsha 410083, China*
(*e-mail: kaiwang@csu.edu.cn; ylwang@csu.edu.cn*)
***Department of Chemical Engineering, Chung-Yuan Christian University*
Chung-Li, Taoyuan, 32023 Taiwan, R.O.C. (*e-mail: jason@ wavenet.cycu.edu.tw*)

Abstract: Process complexities are characterized by strong nonlinearities, dynamics and uncertainties. Modeling such a complex process requires a flexible model with deep layers describing the corresponding strong nonlinear dynamic behavior. The proposed model is constructed by deep neural networks to represent the process of state transition and observation generation, both of which together constitute a stochastic nonlinear state space model. This model is evolved from the variational auto-encoder learned by the stochastic expectation-maximization algorithm. To solve the complexity of posteriors for dynamic processes, the posterior distributions with respect to state variables are constructed by a forward-backward recurrent neural network. One example is given to validate that the proposed method outperforms the comparative methods in modeling complex nonlinearities.

Keywords: learning nonlinear systems, deep neural networks, variational auto-encoders

1. INTRODUCTION

Nonlinear system identification has been developed for learning nonlinear mathematical models from process data. Consider the nonlinear state-space model

$$\begin{aligned} \boldsymbol{z}_k &= f\left(\boldsymbol{z}_{k-1}\right) + \boldsymbol{w}_k \\ \boldsymbol{x}_k &= g\left(\boldsymbol{z}_k\right) + \boldsymbol{v}_k \end{aligned} \tag{1}$$

where $\boldsymbol{z}_k \in \mathbb{R}^n$ are unobserved state variables/latent variables (LVs) and $\boldsymbol{x}_k \in \mathbb{R}^m$ are observations. $f\left(\boldsymbol{z}_{k-1}\right)$ is an unknown nonlinear state transition function. Given the corresponding states, $g\left(\boldsymbol{z}_k\right)$ is another unknown nonlinear function that outputs the true observations. $\boldsymbol{w}_k$ is unmeasured process noise while $\boldsymbol{v}_k$ represents the observation noise. For known or partly known model structures in $f\left(\boldsymbol{z}_{k-1}\right)$ and $g\left(\boldsymbol{z}_k\right)$, the model identification is commonly reduced to the issue of parameter estimation with some optimization strategies (Schön 2011, Abdalmoaty 2019). However, it is often intractable for most chemical processes to summarize a good model structure. Instead, the linear combination of basis functions is often used to parameterize process nonlinearities, constructing a nominal nonlinear model for unknown nonlinear structures (Gopaluni 2010, Svensson 2017). Commonly used basis functions include polynomials, wavelets, the Fourier basis, the radial basis and Gaussian kernels. In this case, the task of model identifications is evolved into the estimation of the coefficients of linear combination and some undetermined parameters in basis functions.

However, several intrinsic drawbacks in the basis function-based strategies have blocked the progress in learning more complex nonlinear dynamic systems. Firstly, a simple linear combination of basis functions defines a model representation with a shallow layer. When treating complex nonlinear processes, the deep model can do it well while the model with a shallow layer may suffer from underfitting (Ge 2018, Yuan 2018). On the other hand, the learning strategies often resorts to Markov Chain Monte Carlo (MCMC) sampling methods such as particle Metropolis-Hastings (Schön 2015) and particle Gibbs with ancestor sampling (Svensson and Schön 2017) to approximate the filtering/smoothing posterior distribution representing data likelihoods with respect to states or the expectation to be maximized. However, sequential sampling makes a fairly high complexity of the algorithm so that learning for (1) is time-consuming and hard to cope with long time series and high-dimensional variables. The drawbacks from both the model structure and the model learning limit the extensions of the existing methods to more complex nonlinear cases with more data. For other model structures like nonlinear autoregressive (AR) models (Martínez-Ramón 2006), they often face the same problems mentioned above.

Recently, deep learning models have become an effective tool with a flexible model structure to simultaneously deal with strong nonlinearities, large-scale data and high-dimensional variables (Goodfellow 2016). By stacking the neurons layer by layer, strong nonlinearities can be represented and high-dimensional observations are condensed into low-dimensional abstract features (Wang 2018). Deep learning communities have developed a number of special models to deal with different applications, e.g. auto-encoders for unsupervised learning, convolutional neural networks for image recognition and recurrent neural networks (RNNs) for time series. Intrinsically, deep learning also uses the combination of basis functions to represent nonlinearities. The difference is that deep learning connects the basis functions with a hierarchical structure layer by layer instead of a simple linear combination. Thus, more flexible and extensible structures are achieved to represent more complex nonlinearities. Moreover, the gradient

backpropagation makes the network easy to train with an acceptable learning efficiency.

One can imagine that it will be a powerful tool if the advantage of deep learning can be taken to overcome the drawbacks of the shallow model and time-consuming conventional model identification methods in the face of complex nonlinearities, dynamics and uncertainties. The challenge is how to construct a deep learning structure corresponding to the state-space model (1). Recently, auto-encoding variational Bayes, also known as variational autoencoders (VAE) (Doersch 2016) was proposed to learn a generative model from LVs to observations with an encoding-decoding deep learning structure. The encoder maps the observations to LVs by learning the posterior distribution $p(\boldsymbol{z}_k \mid \boldsymbol{x}_k)$ while the decoder learns the generative distribution $p(\boldsymbol{x}_k \mid \boldsymbol{z}_k)$. Even though VAE provides a paradigm of neural network models for learning latent variable models with uncertainties, most of the variations of VAE limit the static models without any consideration of Markov chain properties for dynamics. Rahul et al. (Krishnan 2015) proposed an effective method for the learning model (1) by integrating RNN with VAE, known as deep Kalman filters. And bidirectional RNN is also used to produce the smoothing posteriors of LVs given the observations not only in the past but also in the future (Krishnan 2017). However, the bidirectional RNN lacks a theoretical interpretation for the forward recursion in the filter and the backward recursion in the smoother.

In this paper, a novel model structure is proposed based on deep learning to learn nonlinear state-space models. The deep smoother is designed by a forward-backward RNN structure for the state posteriors (encoders). And the time-consuming MCMC sampling is replaced by a simple sampling strategy in model learning. The parameter estimation of the deep learning model is based on the expectation-maximization (EM) algorithm, a learning approach to probabilistic LV models with numerical stability and theoretical convergence. The remaining part of this article is organized as follows. Section 2 briefly introduces the basic ideas about recurrent neural networks (RNNs), the EM algorithm and the variational lower bound. In Section 3, the smoothing posteriors are realized by a structured RNN. Then the whole training strategy is constructed based on the optimized objective in Section 4. Section 5 provides a numerical case for model identification and observation reconstruction, and the final section draws conclusions.

2. PRELIMINARIES

2.1 Recurrent neural networks and filtered distribution

The idea behind RNN makes use of sequential information and deals with complex process dynamics. Specifically, given observation sequences $\boldsymbol{X} = \{\boldsymbol{x}_k \in \mathbb{R}^m, k = 1,2,\cdots,N\}$, RNNs first nonlinearly map the past and current input information $\boldsymbol{x}_{1:k}$, which denotes the sequence from $\boldsymbol{x}_1$ to $\boldsymbol{x}_k$, into an RNN cell state $\boldsymbol{h}_k \in \mathbb{R}^l$ by

$$\boldsymbol{h}_k = \mathcal{A}(\boldsymbol{x}_{1:k}) \tag{2}$$

where $\mathcal{A}$ stands for a general designation for feature extraction with neurons. With a recurrent property, Eq.(2) can be further implemented by

$$\boldsymbol{h}_k = \mathcal{A}(\boldsymbol{h}_{k-1}, \boldsymbol{x}_k) \tag{3}$$

Note that the filtering distribution $q_f(\boldsymbol{z}_k) = p(\boldsymbol{z}_k \mid \boldsymbol{x}_{1:k}) = p(\boldsymbol{z}_k \mid \boldsymbol{z}_{k-1}, \boldsymbol{x}_k)$ has the same structure as (2) and (3), i.e., the current states are generated under the condition of the previous states and the current measurements. Hence, the distribution of $\boldsymbol{z}_k$ can be generated using $\boldsymbol{h}_k$. Specifically, the filters in nonlinear situations can be modeled by the RNN as shown in Fig. 1. Hence, the filters in nonlinear situations can be modeled with RNN (Fig. 1). Note that the colorful nodes in Fig. 1 can represent a deep neural network (DNN) with several hidden layers without any confusion. If local Gauss descriptors are chosen each time, the means and the covariance matrix will be the outputs of the RNN shown in Fig. 2, in which the covariance matrix is assumed to be diagonal. The mean is outputted through a linear function while the diagonal elements of the covariance matrix are outputted through the softplus function $\zeta(x) = ln(1+e^x)$, guaranteeing the values larger than zero.

$$q_f(\boldsymbol{z}_k) = p(\boldsymbol{z}_k \mid \boldsymbol{x}_{1:k}) = \mathcal{N}(\boldsymbol{\mu}_k(\boldsymbol{x}_{1:k}), \Lambda_k(\boldsymbol{x}_{1:k})) \tag{4}$$

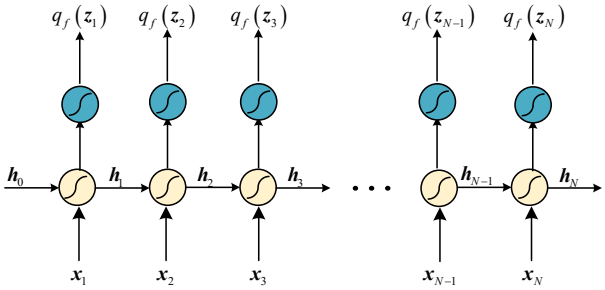

Fig. 1 The structure of RNNs

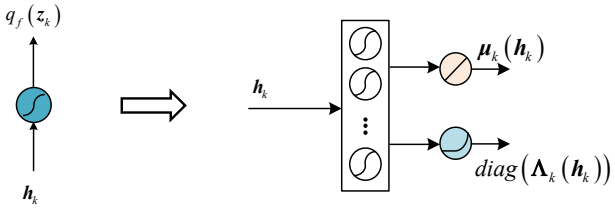

Fig. 2 The structure of the output of an RNN with a Gauss descriptor.

2.2 Variational lower bound

Given the estimated model parameters $\boldsymbol{\theta}$, the marginal log-likelihood function of observations is

$$ln(p(\boldsymbol{X} \mid \boldsymbol{\theta})) = \int q(\boldsymbol{Z}) ln(p(\boldsymbol{X} \mid \boldsymbol{\theta})) d\boldsymbol{Z} \tag{5}$$

where $q(\boldsymbol{Z})$ is any distribution with respect to LVs. Further, (5) can be rewritten as

$$ln(p(\boldsymbol{X} \mid \boldsymbol{\theta})) = kl(q(\boldsymbol{Z}) \| p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta})) + \mathcal{L}(q(\boldsymbol{Z}), \boldsymbol{\theta}) \tag{6}$$

where the first term at the right side is a Kullback–Leibler (KL) divergence measuring the dissimilarity between $q(\boldsymbol{Z})$ and the posterior distribution $p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta})$. The KL divergence is

non-negative and will be zero when $q(\boldsymbol{Z})$ is equal to $p(\boldsymbol{Z}\mid\boldsymbol{X},\boldsymbol{\theta})$. Actually, $\mathcal{L}(q(\boldsymbol{Z}),\boldsymbol{\theta})$ is considered as a variational lower bound of the marginal log-likelihood because there is $\mathcal{L}(q(\boldsymbol{Z}),\boldsymbol{\theta})\le ln(p(\boldsymbol{X}\mid\boldsymbol{\theta}))$. And there is

$$\mathcal{L}(q(\boldsymbol{Z}),\boldsymbol{\theta})=\int q(\boldsymbol{Z})ln\frac{p(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\theta})}{q(\boldsymbol{Z})}d\boldsymbol{Z} \tag{7}$$

It is clear that the lower bound will reach the marginal log-likelihood when $q(\boldsymbol{Z})$ is chosen to be the true posterior $p(\boldsymbol{Z}\mid\boldsymbol{X},\boldsymbol{\theta})$. To maximize $ln(p(\boldsymbol{X}\mid\boldsymbol{\theta}))$, an iterative optimization procedure, known as the EM algorithm, is widely applied. Briefly, EM algorithm estimates parameters by letting the $q(\boldsymbol{Z})$ be equal to $p(\boldsymbol{Z}\mid\boldsymbol{X},\boldsymbol{\theta})$ with the $\boldsymbol{\theta}$ estimated in the last iteration and then maximizing the lower bound in (7) to update $\boldsymbol{\theta}$ in an iterative fashion until convergence.

3. DEEP SMOOTHER

Considering model (1), assume the noise distributions are respectively

$$p(\boldsymbol{w}_k)=\mathcal{N}(0,I) \tag{8}$$

$$p(\boldsymbol{v}_k)=\mathcal{N}(0,\Gamma) \tag{9}$$

where the covariance matrix of $\boldsymbol{w}_k$ can be assumed to be an identity matrix, which will not influence the model representation. Correspondingly, there are

$$p(\boldsymbol{z}_k\mid\boldsymbol{z}_{k-1})=\mathcal{N}(f(\boldsymbol{z}_{k-1}),I) \tag{10}$$

$$p(\boldsymbol{x}_k\mid\boldsymbol{z}_k)=\mathcal{N}(g(\boldsymbol{z}_k),\Gamma) \tag{11}$$

(10) is known as the transition distribution and (11) is the emission distribution, respectively. Simultaneously, the initial LV distribution is assumed to be

$$p(\boldsymbol{z}_0)=\mathcal{N}(\boldsymbol{\mu}_0,\Lambda_0) \tag{12}$$

Thus, these terms $\{\boldsymbol{\mu}_0,\Lambda_0,f(\boldsymbol{z}_{k-1}),g(\boldsymbol{z}_k),\Gamma\}$ are needed to be estimated for process modeling. To learn the model, the lower bound is firstly rewritten as

$$\mathcal{L}(\boldsymbol{\theta})=\int q(\boldsymbol{Z})ln\,p(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\theta})d\boldsymbol{Z}-\int q(\boldsymbol{Z})ln\,q(\boldsymbol{Z})d\boldsymbol{Z} \tag{13}$$

where $q(\boldsymbol{Z})$ is set to be $p(\boldsymbol{Z}\mid\boldsymbol{X},\bar{\boldsymbol{\theta}})$ obtained and taken as a known condition in the last iteration so that the final objective is

$$\begin{aligned}&\arg\max_{\boldsymbol{\theta}}\ \mathcal{L}(\boldsymbol{\theta})=\int q(\boldsymbol{Z})ln\,p(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\theta})d\boldsymbol{Z}+const\\&=\mathbb{E}\{ln\,p(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\theta})\}_{\langle q(\boldsymbol{Z})\rangle}+const\end{aligned} \tag{14}$$

where *const* stands for a constant term to be ignored and $\mathbb{E}\{\cdot\}$ denotes the mathematical expectation operator under some distributions. Moreover, the joint log-likelihood $ln\,p(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\theta})$ can be factorized as

$$\begin{aligned}&ln\,p(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\theta})=\\&ln\,p(\boldsymbol{z}_0\mid\boldsymbol{\theta})+\sum_{k=1}^{N}ln\,p(\boldsymbol{z}_k\mid\boldsymbol{z}_{k-1},\boldsymbol{\theta})+\sum_{k=1}^{N}ln\,p(\boldsymbol{x}_k\mid\boldsymbol{z}_k,\boldsymbol{\theta})\end{aligned} \tag{15}$$

Further, $\mathcal{L}(\boldsymbol{\theta})$ is given by

$$\begin{aligned}&\arg\max_{\boldsymbol{\theta}}\ \mathcal{L}(\boldsymbol{\theta})=-kl(q(\boldsymbol{z}_0)\,||\,p(\boldsymbol{z}_0\mid\boldsymbol{\theta}))\\&-\sum_{k=1}^{N}\mathbb{E}\{kl(q(\boldsymbol{z}_k\mid\boldsymbol{z}_{k-1})\,||\,p(\boldsymbol{z}_k\mid\boldsymbol{z}_{k-1},\boldsymbol{\theta}))\}_{\langle q(\boldsymbol{z}_{k-1})\rangle}\\&+\sum_{k=1}^{N}\mathbb{E}\{ln\,p(\boldsymbol{x}_k\mid\boldsymbol{z}_k,\boldsymbol{\theta})\}_{\langle q(\boldsymbol{z}_k)\rangle}+const\end{aligned} \tag{16}$$

Note that (16) is a result of factorizing the integral of KL divergence $kl(q(\boldsymbol{Z})\,||\,p(\boldsymbol{Z}))$,given in (Krishnan, Shalit et al. 2015).

To optimize objective (16), the first task is to figure out the smoothing posterior distributions, denoted as $q(\boldsymbol{z}_k)=p(\boldsymbol{z}_k\mid\boldsymbol{X})$. Since the process is fairly complex and there is no any prior, these distributions are often intractable. To implement such a complex nonlinear structure, DNNs can approximate the smoothing posterior distribution, the transition equation and the emission equation.

Let $q_s(\boldsymbol{z}_k)=p(\boldsymbol{z}_k\mid\boldsymbol{X})$ where subscript '*s*' specifies the smoothing posteriors and it follows a backward recursion, given by

$$q_s(\boldsymbol{z}_k)=\int p(\boldsymbol{z}_k\mid\boldsymbol{z}_{k+1},\boldsymbol{x}_{1:k})q_s(\boldsymbol{z}_{k+1})\,d\boldsymbol{z}_{k+1} \tag{17}$$

which is easy to derive. It can also refer to the Kalman smoother (Nasrabadi 2007). The backward recursion indicates $q_s(\boldsymbol{z}_k)$ is determined by $q_s(\boldsymbol{z}_{k+1})$ and $\boldsymbol{x}_{1:k}$. That means a backward RNN can be trained to model the backward recursion of smoothed posteriors, given by

$$\tilde{\boldsymbol{h}}_k=\mathcal{A}(\tilde{\boldsymbol{h}}_{k+1},\boldsymbol{x}_{1:k}) \tag{18}$$

where $\tilde{\boldsymbol{h}}_k$ denotes the cell state in the backward RNN. Considering $\boldsymbol{h}_k$ is an information summary of $\boldsymbol{x}_{1:k}$. (18) can be changed into

$$\tilde{\boldsymbol{h}}_k=\mathcal{A}(\tilde{\boldsymbol{h}}_{k+1},\boldsymbol{h}_k) \tag{19}$$

Observing (19), each cell state $\boldsymbol{h}_k$ in forward RNN needs to be connected to the corresponding $\tilde{\boldsymbol{h}}_k$ in backward RNN, which entangles the forward RNN and the backward RNN, making the network structure fairly complicated and potentially increasing extra routes of gradient backpropagation. Note that there is $\boldsymbol{h}_N=\mathcal{A}(\boldsymbol{h}_k,\boldsymbol{x}_{k+1:N})$ for any time point *k* in the forward RNN. That means it is likely to use $\boldsymbol{h}_N$ and $\boldsymbol{x}_{k+1:N}$ to reconstruct the information related to $\boldsymbol{h}_k$, which is needed for constructing the smoothed distribution as implied by (19). From this illustration, one can train a backward RNN, given by

$$\tilde{\boldsymbol{h}}_k=\mathcal{A}(\tilde{\boldsymbol{h}}_{k+1},\boldsymbol{h}_N,\boldsymbol{x}_{k+1:N}) \tag{20}$$

which avoids the direct connection from $\boldsymbol{h}_k$ to $\tilde{\boldsymbol{h}}_k$. Based on those understandings, smoothers are designed and shown in Fig. 3. Firstly, a forward RNN is used to produce filtered distributions $q_f(\boldsymbol{z}_k)$, which are mapped from the forward RNN cell states $\boldsymbol{h}_k$. Then the cell states $\tilde{\boldsymbol{h}}_k$ in the backward RNN can be utilized to produce smoothed distributions

$q_s(z_k)$ according to (20). Specifically, the smoothed distribution is described by

$$q_s(z_k) = p(z_k \mid \boldsymbol{X}) = \mathcal{N}\left(\tilde{\boldsymbol{\mu}}_k(\tilde{\boldsymbol{h}}_k), \tilde{\Lambda}_k(\tilde{\boldsymbol{h}}_k)\right) \quad (21)$$

Note that the endpoint cell state $\boldsymbol{h}_N$ of the forward RNN is connected to the initial cell state $\tilde{\boldsymbol{h}}_N$ of the backward RNN, as $q_s(z_N) = q_f(z_N)$. Also, the ending cell state $\tilde{\boldsymbol{h}}_0$ of the backward RNN is fed into the initial cell state $\boldsymbol{h}_0$ of the forward RNN in the next iteration.

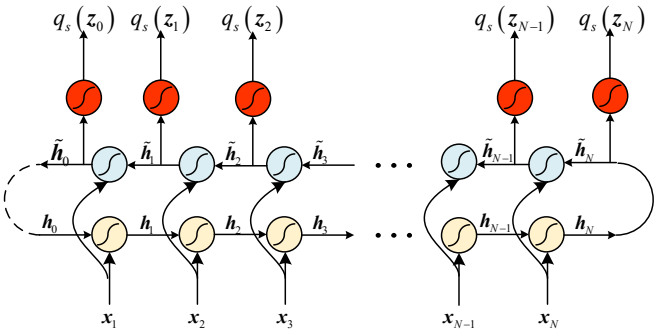

Fig. 3 The smoother generated from a forward-backward RNN

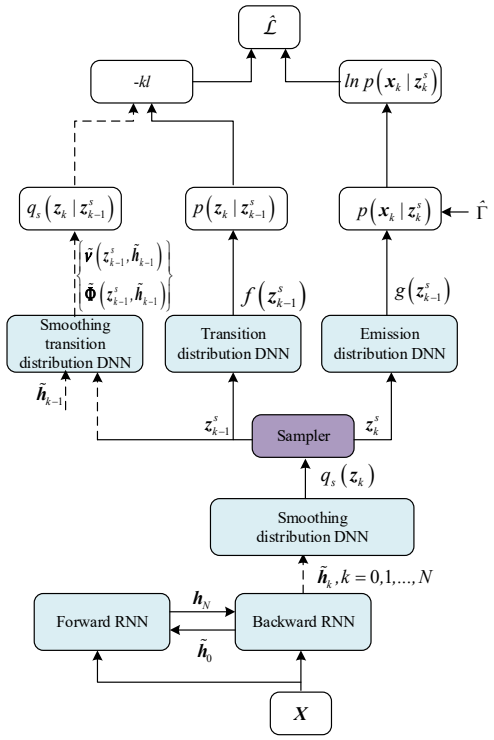

Fig. 4 The network structure for learning nonlinear state-space models

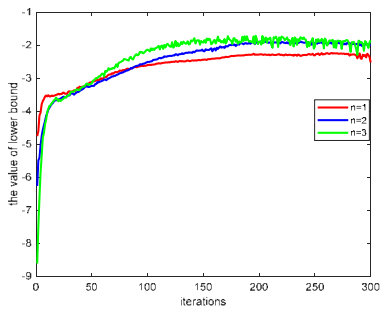

Fig.5 The curves of iterations for different n

4. MODEL TRAINING

After the posterior distribution is implemented by the forward-backward RNN, it is feasible to learn the nonlinear dynamic systems with the collected observation sequences. Hence, offline learning is performed with the smoothing distribution $q_s(z_k)$. In the offline learning stage, the first term in (16) will be $-kl\left(q_s(z_0) \| p(z_0 \mid \boldsymbol{\theta})\right)$, implying the learned initial LV distribution $p(z_0)$ should be close to the smoothing posterior $q_s(z_0)$. This has been implemented by the forward-backward RNN (Fig. 3) where $\tilde{\boldsymbol{h}}_0$ is the final output of the backward RNN. Also, $\tilde{\boldsymbol{h}}_0$ (the dashed line in Fig. 3) is assigned as the initial cell state of the forward RNN in each new iteration. However, the second and the third terms in the right side of (16) require the calculation of expectations associated with the smoothers. Because of complex distribution structures, it is hard to perform the integral for deriving the expectations. Therefore, like VAE, the empirical average is used in place of the true expectation, so the final optimization objective is

$$\begin{aligned} \underset{\boldsymbol{\theta}}{arg\,max}\ \hat{\mathcal{L}}(\boldsymbol{\theta}) &= -kl\left(q(z_0) \| p(z_0 \mid \boldsymbol{\theta})\right) \\ &+ \frac{1}{S}\sum_{k=1}^{N}\sum_{s=1}^{S} ln\, p\left(\boldsymbol{x}_k \mid z_k^s, \boldsymbol{\theta}\right) \\ &- \frac{1}{S}\sum_{k=1}^{N}\sum_{s=1}^{S} kl\left(q_s(z_k \mid z_{k-1}^s) \| p(z_k \mid z_{k-1}^s, \boldsymbol{\theta})\right) \end{aligned} \quad (22)$$

where z_k^s is s-th sampling point drawn from $q_s(z_k)$. And there are a total of S sampling points in each time point k. Moreover, the experience has been proved in VAE that S can be 1 in each iteration. The intuition is the analogy to stochastic gradient descent in which just one sample is used to update the network parameters in each iteration. After sufficient iterations, the parameters will be finally converged. Note that the sampling here is conducted in each individual Gaussian distribution without an expensive sampling cost in sequential Monte Carlo.

In the offline stage, $q_s(z_k \mid z_{k-1}^s)$ is given by the smoothed transition distribution $p(z_k \mid z_{k-1}^s, \boldsymbol{X})$, equal to $p(z_k \mid z_{k-1}^s, \boldsymbol{x}_{k:N})$. With the local Gaussian descriptors, $q_s(z_k \mid z_{k-1}^s)$ is constructed by

$$p(z_k \mid z_{k-1}^s, \boldsymbol{x}_{k:N}) = \mathcal{N}\left(\tilde{\nu}(z_{k-1}^s, \tilde{\boldsymbol{h}}_{k-1}), \tilde{\Phi}(z_{k-1}^s, \tilde{\boldsymbol{h}}_{k-1})\right) \quad (23)$$

Here $\tilde{\boldsymbol{h}}_{k-1}$ is used in the backward RNN to represent the information of $\boldsymbol{x}_{k:N}$. Similarly, a smoothed transition DNN similar to Fig. 3 is used to construct the distribution $q_s(z_k \mid z_{k-1}^s)$. Regarding $p(z_k \mid z_{k-1}^s)$ in (22), it has been represented in (10) where the means vary with z_{k-1}^s but the covariance matrix is identity. Given z_{k-1}^s, a transition DNN for $p(z_k \mid z_{k-1}^s)$ just outputs the mean $f(z_{k-1}^s)$. Similarly, an

emission DNN for $p\left(\boldsymbol{x}_k \mid \boldsymbol{z}_k^s\right)$ in (22) will output $g\left(\boldsymbol{z}_k^s\right)$ and the corresponding covariance matrix is estimated by

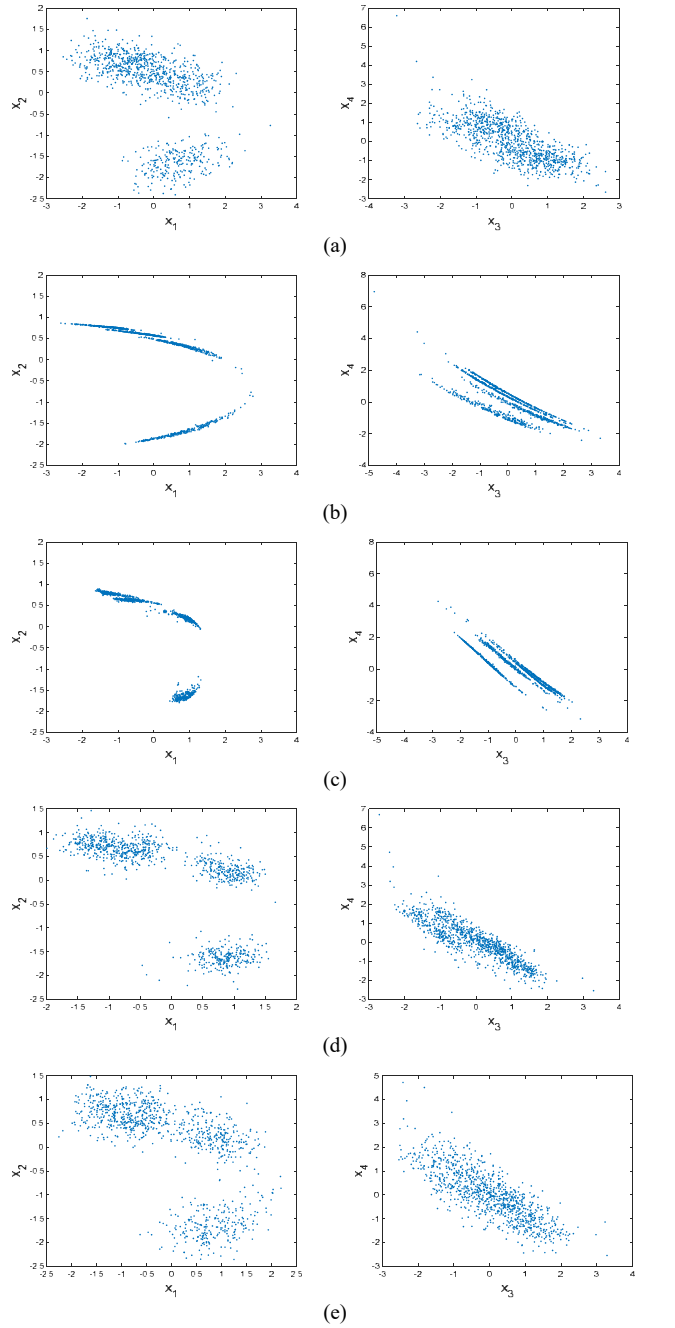

Fig. 6 The scatter plots of variables: From the left to the right, they are the scatter plots of $x^{(1)}-x^{(2)}$ and $x^{(3)}-x^{(4)}$.(a): the scatter plots of observations contaminated by measurement noises; (b): the true observations without measurement noises; (c): the reconstructed observations by the proposed method; (d): the reconstructed observations by the nonlinear AR model. (e): the reconstructed observations by the basis function method.

$$\hat{\Gamma}=\frac{1}{N-1}\sum_{k=1}^{N}\left(\boldsymbol{x}_k-g\left(\boldsymbol{z}_k^s\right)\right)\left(\boldsymbol{x}_k-g\left(\boldsymbol{z}_k^s\right)\right)^T \tag{24}$$

The whole network structure is given in Fig. 4. Firstly, sampling is performed based on the smoothed posterior distribution. Then, the sampling values are inputted into the smoothed transition distribution, transition distribution and emission distribution, all of which are implemented by DNNs. Since the Gaussian distributions are chosen, the KL divergence and the conditional log-likelihood in (22) have closed forms. However, a general sampler prevents the gradient from backpropagation because the sampling operation is non-differentiable. To solve this issue, reparameterization trick of Gaussian distribution (Doersch 2016) makes the network learnable without extra cost or compromise. The idea behind reparameterization is that the distribution in (21) can be regarded as an affine transformation of unit distribution $p(\varepsilon)=\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$ as follows:

$$\boldsymbol{z}_k=\tilde{\Lambda}_k\left(\tilde{\boldsymbol{h}}_k\right)^{\frac{1}{2}}\boldsymbol{\varepsilon}+\tilde{\boldsymbol{\mu}}_k\left(\tilde{\boldsymbol{h}}_k\right) \tag{25}$$

which is differentiable. Each required point $\boldsymbol{z}_k^s$ is given by $\boldsymbol{z}_k^s=\tilde{\Lambda}_k\left(\tilde{\boldsymbol{h}}_k\right)^{\frac{1}{2}}\boldsymbol{\varepsilon}^s+\tilde{\boldsymbol{\mu}}_k\left(\tilde{\boldsymbol{h}}_k\right)$ where ε^s is sampled from the unit Gaussian distribution. By reparameterizaiton, the gradient back-propagation is able to learn the network weights and biases. The iterative optimization is terminated when the lower bound tends to be stationary without increasing significantly.

Before the network is trained, the number of LVs (n) is an important hyperparameter to be predefined. A poor choice of n that is different from the true case can cause a severe model bias, which means the learned model cannot represent the training data well. In this work, different values of n are chosen and the one is selected as the optimal choice that makes the lower bound as large as possible when the iteration terminates.

5. ILLUSTRATIVE EXAMPLES

The mathematical model of the nonlinear dynamic system in the numerical example is given as

$$\begin{aligned}
x_k^{(1)}&=0.1z_k^{(1)}+z_k^{(1)}/\sqrt{\left(z_k^{(1)}\right)^2+\left(z_k^{(2)}\right)^2}+v_k^{(1)}\\
x_k^{(2)}&=0.1z_k^{(1)}z_k^{(2)}+z_k^{(2)}/\sqrt{\left(z_k^{(1)}\right)^2+\left(z_k^{(2)}\right)^2}+v_k^{(2)}\\
x_k^{(3)}&=cos^3\left(z_k^{(1)}\right)+0.1e^{sin\left(z_k^{(2)}\right)}+v_k^{(3)}\\
x_k^{(4)}&=sin^3\left(z_k^{(1)}\right)+ln\left(2+cos\left(z_k^{(2)}\right)\right)+v_k^{(4)}
\end{aligned} \tag{26}$$

where the four measurement noises ($v_k^{(i)}, i=1,2,3,4$) are zero-mean Gaussian with the standard deviations 0.05, 0.16, 0.02 and 0.05, respectively. And there are two LVs $z_k^{(1)}$ and $z_k^{(2)}$ following the nonlinear dynamics as follows:

$$\begin{aligned}
z_k^{(1)}&=cos^3\left(z_{k-1}^{(1)}\right)+0.1e^{sin\left(z_{k-1}^{(1)}\right)}+w_k^{(1)}\\
z_k^{(2)}&=sin\left(e^{z_{k-1}^{(2)}}+\left(z_{k-1}^{(2)}\right)^2\right)+w_k^{(2)}
\end{aligned} \tag{27}$$

where process noises ($w_k^{(1)}$ and $w_k^{(2)}$) are zero-mean Gaussian with the standard deviations 0.01. The networks in Fig. 4 are configured in this way: all DNNs are three hidden layers and each hidden layer has 30 neurons. And a single-layer RNN with 50 neurons is applied. Adam optimizer with the learning rate of 0.001 in TensorFlow is used. A total of 1,000 normal samples is generated for training the model. By incrementing the number of LVs from $n=1$, the optimal n is determined by maximizing the lower bound. Fig. 5 shows the trend of the variational lower bound as the iteration promotes forward. The objectives in the three cases tend to be stationary at about the

200th iteration and the values are maximized when $n = 2$ and $n = 3$. Since $n = 2$ is a simpler structure and the corresponding curve in Fig. 5 is smoother than that with $n = 3$, $n = 2$ is an optimal choice for this example, living up to the true settings. The parameters at the 200th iterations are adopted for learning the nonlinear dynamic models. Note that the overall trend of the lower bound is gradually growing, but it is not necessarily smooth just like the stochastic gradient descent because of sampling. After the process models are trained, the means of learned emission distribution can be considered as the reconstructed values for the true values of observations. Fig. 6(a), (b) and (c) present the scatter plots of the observations contaminated by the measurement noises, the true observations without measurement noises, and the reconstructed observations by the proposed method. One can see the severe nonlinearity between variables and the noises can make the contour profile fairly unclear. The reconstructions in Fig. 6 show the proposed method is effective at modeling the nonlinear dynamics with some uncertainty.

Two typical nonlinear models, the nonlinear AR model (Zhu 2002, Ding 2019) and the basis function-based state-space representation (Gopaluni 2010, Svensson and Schön 2017), are chosen as comparative models to test the identification performance in terms of the observation reconstruction errors. In the training stage, the validation set with 300 samples is used to find suitable hyperparameters, such as the iterative number and the number of basis functions. For the nonlinear AR model, the 40 wavelet bases are chosen and the reconstructed values are plotted in Fig. 6(d). The nonlinear AR model is an input-output model. When the nonlinear state-space model without any structure priors is considered, the state equations and output equations should be simultaneously structured by linear combinations of the basis functions. In this case, the radial basis function suggested in (Gopaluni 2010) is chosen. And the number of bases is also 40. The corresponding scatter plots of the reconstructed observations are also shown in Fig. 6(e). From the reconstruction performance in this numerical example, one can see that the two comparative methods have a mediocre ability to identify such complex nonlinearities. Several elementary nonlinear functions like exponential, sinusoidal and logarithm functions are composite in a complex fashion. That means the structured deep model presents a powerful ability in complex nonlinear situations.

6. CONCLUSIONS

In this paper, a novel structured deep learning representation for complex nonlinear state-space models is proposed. Especially the deep smoother is designed by connecting the forward RNN with the backward RNN in the ending state, realizing the forward recursion and backward recursion in the dynamic systems. Moreover, the training strategy takes advantage of the simple sampling from each individual Gaussian distribution, avoiding severe computation burden and sometimes instability in MCMC processes. The simulation results with a complex nonlinear example sufficiently show the efficacy of the proposed identification methods.

ACKNOWLEDGEMENTS

This paper is supported by the National Natural Science Foundation of China (Grant No. U1911401) and Ministry of Science and Technology, Taiwan, R.O.C. (MOST 106-2221-E-033-060-MY3).

REFERENCES

WAbdalmoaty, M. R.-H. and H. Hjalmarsson (2019). Linear prediction error methods for stochastic nonlinear models. *Automatica* 105: 49-63.

Ding, J., Z. Cao, J. Chen and G. Jiang (2019). Weighted Parameter Estimation for Hammerstein Nonlinear ARX Systems. *Circuits, Systems, and Signal Processing*: 1-15.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Ge, Z. (2018). Process data analytics via probabilistic latent variable models: A tutorial review. *Industrial & Engineering Chemistry Research*.

Goodfellow, I., Y. Bengio and A. Courville (2016). Deep learning, *MIT press*.

Gopaluni, R. B. (2010). Nonlinear system identification under missing observations: The case of unknown model structure. *Journal of Process Control* 20(3): 314-324.

Krishnan, R. G., U. Shalit and D. Sontag (2015). Deep kalman filters. *arXiv preprint arXiv:1511.05121*.

Krishnan, R. G., U. Shalit and D. Sontag (2017). Structured inference networks for nonlinear state space models. Thirty-First AAAI Conference on Artificial Intelligence.

Martínez-Ramón, M., J. L. Rojo-Alvarez, G. Camps-Valls, J. Muñoz-Marí, E. Soria-Olivas and A. R. Figueiras-Vidal (2006). Support vector machines for nonlinear kernel ARMA system identification. *IEEE Transactions on Neural Networks* 17(6): 1617-1622.

Nasrabadi, N. M. (2007). Pattern recognition and machine learning. *Journal of electronic imaging* 16(4): 049901.

Schön, T. B., F. Lindsten, J. Dahlin, J. Wågberg, C. A. Naesseth, A. Svensson and L. Dai (2015). Sequential Monte Carlo Methods for System Identification. *IFAC-PapersOnLine* 48(28): 775-786.

Schön, T. B., A. Wills and B. Ninness (2011). System identification of nonlinear state-space models. *Automatica* 47(1): 39-49.

Svensson, A. and T. B. Schön (2017). A flexible state–space model for learning nonlinear dynamical systems. *Automatica* 80: 189-199.

Wang, K., B. Gopaluni, J. Chen and Z. Song (2018). Deep Learning of Complex Batch Process Data and Its Application on Quality Prediction. *IEEE Transactions on Industrial Informatics*.

Yuan, X., B. Huang, Y. Wang, C. Yang and W. Gui (2018). Deep Learning-Based Feature Representation and Its Application for Soft Sensor Modeling With Variable-Wise Weighted SAE. *IEEE Transactions on Industrial Informatics* 14(7): 3235-3243.

Zhu, Y. (2002). Estimation of nonlinear ARX models. Proceedings of the 41st IEEE Conference on Decision and Control, 2002., IEEE.