Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Nonlinear Dynamic Boltzmann Machines for Time-Series Prediction

Sakyasingha Dasgupta and **Takayuki Osogami**
IBM Research - Tokyo
{sdasgup, osogami}@jp.ibm.com

Abstract

The dynamic Boltzmann machine (DyBM) has been proposed as a stochastic generative model of multi-dimensional time series, with an exact, learning rule that maximizes the log-likelihood of a given time series. The DyBM, however, is defined only for binary valued data, without any nonlinear hidden units. Here, in our first contribution, we extend the DyBM to deal with real valued data. We present a formulation called Gaussian DyBM, that can be seen as an extension of a vector autoregressive (VAR) model. This uses, in addition to standard (explanatory) variables, components that captures long term dependencies in the time series. In our second contribution, we extend the Gaussian DyBM model with a recurrent neural network (RNN) that controls the bias input to the DyBM units. We derive a stochastic gradient update rule such that, the output weights from the RNN can also be trained online along with other DyBM parameters. Furthermore, this acts as nonlinear hidden layer extending the capacity of DyBM and allows it to model nonlinear components in a given time-series. Numerical experiments with synthetic datasets show that the RNN-Gaussian DyBM improves predictive accuracy upon standard VAR by up to $\approx 35\%$. On real multi-dimensional time-series prediction, consisting of high nonlinearity and non-stationarity, we demonstrate that this nonlinear DyBM model achieves significant improvement upon state of the art baseline methods like VAR and long short-term memory (LSTM) networks at a reduced computational cost.

Introduction

The Boltzmann machine (BM) (Hinton and Sejnowski 1983; Ackley, Hinton, and Sejnowski 1985) is an artificial neural network that is motivated by the Hebbian learning rule (Hebb 1949) of biological neural networks, in order to learn a collection of static patterns. That is, the learning rule of the BM that maximizes the log likelihood of given patterns exhibits a key property of the Hebb's rule, i.e. co-activated neural units should be connected. Although the original BM is defined for binary values (1 for neuron firing and 0 representing silence), it has been extended to deal with real values in the form of Gaussian BM (Marks and Movellan 2001; Welling, Rosen-Zvi, and Hinton 2004) or Gaussian-Bernoulli restricted BM (Hinton and Salakhutdinov 2006) primarily for engineering purposes. Unlike BM, the recently proposed dynamic Boltzmann machine (DyBM) (Osogami and Otsuka 2015a; 2015b) can be used to learn a generative model of temporal pattern sequences, using an exact learning rule that maximises the log likelihood of given time-series. This learning rule exhibits key properties of spike-timing dependent plasticity (STDP), a variant of the Hebbian rule. In STDP, the amount of change in the synaptic strength between two neurons that fired together depends on precise timing when the two neurons fired. However, similar to a BM, in the DyBM, each neuron takes a binary value, 0 or 1, following a probability distribution that depends on the parameters of the DyBM. This has limited applicability to real-world time-series modeling problems, which are often real valued.

First Contribution: Here, we extend the DyBM to deal with real values and refer to the extended model as a Gaussian DyBM. Although extension is possible in the way that BM is extended to the Gaussian BM, we also relax some of the constraints that the DyBM has required in (Osogami and Otsuka 2015a; 2015b). The primary purpose of these constraints in (Osogami and Otsuka 2015a; 2015b) was to interpret its learning rule as biologically plausible STDP. We relax these constraints in a way that the Gaussian DyBM can be related to a vector autoregressive (VAR) model (Lütkepohl 2005; Bahadori, Liu, and Xing 2013), while keeping the key properties of the DyBM having an exact learning rule intact. This makes our new model specifically suited for time-series prediction problems. Specifically, we show that a special case of the Gaussian DyBM is a VAR model having additional components that capture long term dependency of time series. These additional components correspond to DyBM's eligibility traces, which represent how recently and frequently spikes arrived from one unit[1] to another. This forms the first primary contribution of this paper.

Second Contribution: Similar to DyBM, its direct extension to the Gaussian case, can have two restrictions. Firstly, the maximum number of units has to be equal to the dimension of the time-series being learned, and secondly, the absence of nonlinear hidden units. In our second contribution, we extend DyBM by adding a RNN layer that computes a

[1] In the rest of the paper the term neuron and unit are used interchangeably.

high dimensional nonlinear feature map of past input sequences (from the time-series data) to DyBM. The output from the RNN layer is used to update the bias parameter in Gaussian DyBM at each time. As such, we call this extension as RNN-Gaussian DyBM. The RNN is modeled similar to an echo state network (Jaeger and Haass 2004). Such that, the weights in the RNN layer is fixed randomly, and we only update the weights from the recurrent layer to the bias layer for Gaussian DyBM. We derive a stochastic gradient descent (SGD) update rule for the weights, with the objective of maximizing the log likelihood of given time-series. RNN-Gaussian DyBM, thus allows hidden nonlinear units in the form of the recurrent layer, where by the size of RNN layer can be selected differently than the dimension of time-series being modeled. This can significantly improve the learning of high-dimensional time-series by enabling very long temporal memory in DyBM that can also deal with nonlinear dynamics of the data.

Evaluation: We demonstrate the effectiveness of this nonlinear DyBM model, namely, RNN-Gaussian DyBM through numerical experiments on two different synthetic datasets. Furthermore, we also evaluate its performance on more complex real time-series prediction using non-stationary multidimensional financial time-series and nonlinear sunspot time-series prediction problems . We train the RNN-Gaussian DyBM and let it predict the future values of the time-series in a purely online manner with SGD (Tieleman and Hinton 2012). Namely, at each moment, we update the parameters and variables by using only the latest values of the time-series, and let the DyBM predict the next values of time-series. The experimental results show that the RNN-Gaussian DyBM can significantly better the predictive performance against standard VAR methods, as well as match (and on occasion outperform) the performance of state of the art RNNs like long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) networks. Furthermore, RNN-Gaussian DyBM can be implemented in an online manner, at a significantly reduced cost.

Related work

In the remainder of this section, we review the prior work related to ours. Besides the DyBM, the BM has also been extended into temporal horizon to deal with time-series in various ways, and these extensions have been shown to perform effectively in practice (Sutskever, Hinton, and Taylor 2009; Hinton and Brown 1999; Sutskever and Hinton 2007; Taylor and Hinton 2009; Mittelman et al. 2014). In particular, Gaussian units have been applied in (Sutskever, Hinton, and Taylor 2009; Taylor and Hinton 2009; Mittelman et al. 2014). Unlike the DyBM, however, exact learning of the log-liklihood gradient without back-propagation and sampling, in these extended models is intractable and needs to be approximated. On the other hand, the RNN-Gaussian DyBM can be trained to maximize the log-likelihood of given time-series without approximation, and this learning rule has the characteristics of STDP, which is inherited from DyBM.

A large amount of the prior work has compared recurrent neural networks (RNN) against autoregressive models (Connor, Atlas, and Martin 1992; Zhang, Patuwo, and Hu 1998).

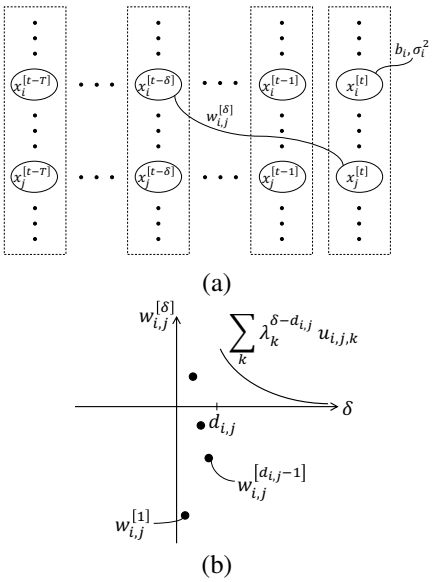


Figure 1: (a) A Gaussian Boltzmann machine (BM) that when unfolded in time, gives a Gaussian DyBM as $T \to \infty$. (b) The parametric form of the weight assumed in the Gaussian BMs and its extensions.

The focus of such study, however, is on non-linearity of simple RNNs. The Gaussian DyBM formulation, first extends the linear VAR model but with the additional variables, related to DyBM's eligibility traces, which take into account the long term dependency in time-series. The addition of the RNN layer then allows to model the nonlinear components of the time-series.

Deriving a Gaussian DyBM

We will define a G-DyBM[2] as a limit of a sequence of Gaussian BMs. Each of the Gaussian BMs defines a probability distribution of the patterns that it generates, and an analogous probability distribution for the G-DyBM is defined as a limit of the sequence of those probability distributions.

Gaussian DyBM as unfolded Gaussian Boltzmann machines for $T \to \infty$

Consider a Gaussian BM having a structure illustrated in Figure 1 (a). This Gaussian BM consists of $T+1$ layers of units. Let, N be the number of units in each layer. This Gaussian BM can represent a series of N-dimensional patterns of length $T+1$. In the figure, this series of patterns is denoted as $\mathbf{x}^{[t-T,t]} \equiv (\mathbf{x}^s)_{s=t-T,\ldots,t}$ for some time t. That is, the δ-th layer represents the pattern, $\mathbf{x}^{[t-\delta]} \equiv (x_i^{[t-\delta]})_{i=1,\ldots,N}$, at time $t-\delta$ for $\delta = 0, 1, \ldots, T$.

The Gaussian BM in Figure 1(a) has three kinds of parameters, bias, variance, and weight, which determine the probability distribution of the patterns that the Gaussian BM

[2]In interest of space, we refer to Gaussian DyBM as G-DyBM and its RNN extension as RNN-G-DyBM on certain occasions.

generates. For $i = 1,\dots,N$, let b_i be the bias of the i-th unit of any layer and σ_i^2 be the variance of the i-th unit of any layer. Let $w_{i,j}^{[\delta]}$ be the weight between the i-th unit of the $(s+\delta)$-th layer and the j-th unit of the s-th layer for $(i,j) \in \{1,\dots,N\}^2$, $s = 0,\dots,T-\delta$, and $\delta = 1,\dots,T$. We assume that there are no connections within each layer: namely $w_{i,j}^{[0]} = 0$. With this assumption, the most recent values, $x_i^{[t]}$ for $i = 1,\dots,N$, are conditionally independent of each other given $\mathbf{x}^{[t-T,t-1]}$.

Hence, the conditional probability density of $\mathbf{x}^{[t]}$ given $\mathbf{x}^{[t-T,t-1]}$ can be represented as

$$p(\mathbf{x}^{[t]}|\mathbf{x}^{[t-T,t-1]}) = \prod_{j=1}^{N} p_j(x_j^{[t]}|\mathbf{x}^{[t-T,t-1]}), \quad (1)$$

where each factor of the right-hand side denotes the conditional probability density of $x_j^{[t]}$ given $\mathbf{x}^{[t-T,t-1]}$ for $j = 1,\dots,T$. More specifically, $x_j^{[t]}$ has a Gaussian distribution for each j:

$$p_j(x_j^{[t]}|\mathbf{x}^{[t-T,t-1]}) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\Big(-\frac{\big(x_j^{[t]} - \mu_j^{[t]}\big)^2}{2\sigma_j^2}\Big), \quad (2)$$

where $\mu_j^{[t]}$ takes the following form and can be interpreted as the expected value of the j-th unit at time t given the last T patterns:

$$\mu_j^{[t]} \equiv b_j + \sum_{\delta=1}^{T}\sum_{i=1}^{N} w_{i,j}^{[\delta]} x_i^{[t-\delta]}. \quad (3)$$

To take the limit of $T \to \infty$ while keeping the number of parameters constant (*i.e.*, independent of T), we assume that the weight has the parametric form illustrated in Figure 1(b). Here, $d_{i,j} \geq 1$ is an integer and will represent the conduction delay from i to j in the G-DyBM. Specifically, for $\delta \geq d_{i,j}$, we use the parametric form of the DyBM (Osogami and Otsuka 2015a; 2015b):

$$w_{i,j}^{[\delta]} = \sum_{k=1}^{K} \lambda_k^{\delta - d_{i,j}} u_{i,j,k}, \quad (4)$$

where, for each k, $u_{i,j,k}$ is a learnable parameter, and the decay rate, λ_k, is fixed in range $[0,1)$. Unlike the DyBM, we assume no constraint on $w_{i,j}^{[\delta]}$ for $0 < \delta < d_{i,j}$. Although these weight could have shared weight in the G-DyBM as well, the unconstrained weight will allow us to interpret the G-DyBM as an extension of the VAR model in the next section.

Plugging (4) into (3) and letting $T \to \infty$, we obtain

$$\mu_j^{[t]} \equiv b_j + \sum_{i=1}^{N}\sum_{\delta=1}^{d_{i,j}-1} w_{i,j}^{[\delta]} x_i^{[t-\delta]} + \sum_{i=1}^{N}\sum_{k=1}^{K} u_{i,j,k}\, \alpha_{i,j,k}^{[t-1]}, \quad (5)$$

where $\alpha_{i,j,k}^{[t-1]}$ will be referred to as an eligibility trace and is defined as follows:

$$\alpha_{i,j,k}^{[t-1]} \equiv \sum_{\delta=d_{i,j}}^{\infty} \lambda_k^{\delta - d_{i,j}} x_i^{[t-\delta]}. \quad (6)$$

Notice that the eligibility trace can be computed recursively:

$$\alpha_{i,j,k}^{[t]} = \lambda_k\, \alpha_{i,j,k}^{[t-1]} + x_i^{[t-d_{i,j}+1]}. \quad (7)$$

Gaussian DyBM as extended vector autoregression

When the expression (5) is seen as a regressor (or predictor) for $x_j^{[t]}$, it can be understood as a model of vector autoregression (VAR) with two modifications to the standard model. First, the last term in the right-hand side of (5) involves eligibility traces, which can be understood as features of historical values, $\mathbf{x}^{[-\infty,t-d_{i,j}]}$, and are added as new variables of the VAR model. Second, the expression (5) allows the number of terms (*i.e.,* lags) to depend on i and j through $d_{i,j}$, while this number is common among all pairs of i and j in the standard VAR model.

When the conduction delay has a common value ($d_{i,j} = d, \forall i,j$), we can represent (5) simply with vectors and matrices as follows:

$$\mu^{[t]} = \mathbf{b} + \sum_{\delta=1}^{d-1} \mathbf{W}^{[\delta]}\, \mathbf{x}^{[t-\delta]} + \sum_{k=1}^{K} \mathbf{U}_k\, \alpha_k^{[t-1]}, \quad (8)$$

where $\mu \equiv (\mu_j)_{j=1,\dots,N}$, $\mathbf{b} \equiv (b_j)_{j=1,\dots,N}$, $\mathbf{x}^{[t]} \equiv (x_j^{[t]})_{j=1,\dots,N}$, and $\alpha_k^{[t-1]} \equiv (\alpha_{j,k}^{[t-1]})_{j=1,\dots,N}$ for $k = 1,\dots,K$ are column vectors; $\mathbf{W}^{[\delta]} \equiv (\tilde{w}_{i,j})_{(i,j)\in\{1,\dots,N\}^2}$ for $0 < \delta < d_{i,j}$ and $\mathbf{U}_k \equiv (u_{i,j,k})_{(i,j)\in\{1,\dots,N\}^2}$ for $k = 1,\dots,K$ are $N \times N$ matrices.

RNN-Gaussian DyBM as a nonlinear model

Having derived the G-DyBM, we now formulate the RNN-G-DyBM, as a nonlinear extension of the G-DyBM model by updating the bias parameter vector $\mathbf{b}$, at each time using a RNN layer. This RNN layer computes a nonlinear feature map of the past time series input to the G-DyBM. Where in, the output weights from the RNN to the bias layer along with DyBM parameters, can be updated online using a stochastic gradient method.

We consider a G-DyBM connected with a M-dimensional RNN, whose state vector changes dependent on a nonlinear feature mapping of its own history and the N-dimensional time-series input data vector at time $t-1$. Where in, for most settings $M > N$. Specifically, for RNN-G-DyBM we consider the bias vector to be time-dependent. Where in, it is updated at each time as:

$$\mathbf{b}^{[t]} = \mathbf{b}^{[t-1]} + \mathbf{A}^\top \Psi^{[t]} \quad (9)$$

Here, $\Psi^{[t]}$ is the $M \times 1$ dimensional state vector at time t of a M dimensional RNN. $\mathbf{A}$ is the $M \times N$ dimensional learned output weight matrix that connects the RNN state to the bias vector. Where, the RNN state is updated based on the input time-series vector $\mathbf{x}^{[t]}$ as follows:

$$\Psi^{[t]} = (1-\rho)\Psi^{[t-1]} + \rho\, \mathscr{F}(\mathbf{W}_{rnn}\Psi^{[t-1]} + \mathbf{W}_{in}\mathbf{x}^{[t]}), \quad (10)$$

Where, $\mathscr{F}(x) = \texttt{tanh}(x)$. This can however be replaced by any other suitable nonlinear function, e.g. rectified linear units, sigmoid etc. Here, $0 < \rho \leq 1$ is a leak rate hyper-parameter of the RNN, which controls the amount of memory in each unit of the RNN layer. $\mathbf{W}_{rnn}$ and $\mathbf{W}_{in}$ are the

$M \times M$ dimensional RNN weight matrix and $N \times M$ dimensional projection of the time series input to the RNN layer, respectively. Here, we design the RNN similar to an echo state network (Jaeger and Haass 2004). Such that, the weight matrices $\mathbf{W}_{rnn}$ and $\mathbf{W}_{in}$ are initialized randomly. $\mathbf{W}_{rnn}$ is initialized from a Gaussian distribution $\mathscr{N}(0,1)$ and $\mathbf{W}_{in}$ is initialized from $\mathscr{N}(0,0.1)$. The sparsity of the RNN weight matrix can be controlled by the parameter ϕ and it is scaled to have a spectral radius less than one, for stability (Jaeger and Haass 2004). For all results presented here, the RNN weight matrix was 90% sparse and had a spectral radius of 0.95.

Online training of a RNN-Gaussian DyBM

We now derive an online learning rule for the RNN-G-DyBM in a way that the log-likelihood of given time-series data, $\mathscr{D}$, is maximized. The log-likelihood of $\mathscr{D}$ is given by

$$\mathrm{LL}(\mathscr{D}) = \sum_{\mathbf{x}\in\mathscr{D}} \sum_{t} \log p(\mathbf{x}^{[t]}|\mathbf{x}^{[-\infty,t-1]}). \tag{11}$$

Here, we show the case where $\mathscr{D}$ consists of a single time-series, but extension to multidimensional cases is straightforward. The approach of stochastic gradient is to update the parameters of the RNN-G-DyBM at each step, t, according to the gradient of the conditional probability density of $\mathbf{x}^{[t]}$,

$$\nabla \log p(\mathbf{x}^{[t]}|\mathbf{x}^{[-\infty,t-1]}) = \sum_{i=1}^{N} \nabla \log p_k(x_i^{[t]}|\mathbf{x}^{[-\infty,t-1]}) \tag{12}$$

$$= -\sum_{i=1}^{N} \Big(\frac{1}{2} \nabla \log \sigma_i^2 + \nabla \frac{(x_i^{[t]} - \mu_i^{[t]})^2}{2\sigma_i^2} \Big), \tag{13}$$

where, the first equality follows from the conditional independence (1), and the second equality follow from (2). From (13) and (5), we can now derive the derivative with respect to each parameter. These parameters can then be updated, for example, as follows:

$$b_j \leftarrow b_j + \eta \frac{(x_j^{[t]} - \mu_j^{[t]})}{\sigma_j^2}, \tag{14}$$

$$\sigma_j \leftarrow \sigma_j + \eta \left(\frac{(x_j^{[t]} - \mu_j^{[t]})^2}{\sigma_j^2} - 1 \right) \frac{1}{\sigma_j}, \tag{15}$$

$$w_{i,j}^{[\delta]} \leftarrow w_{i,j}^{[\delta]} + \eta \frac{(x_j^{[t]} - \mu_j^{[t]})}{\sigma_j^2} x_i^{[t-\delta]}, \tag{16}$$

$$u_{i,j,k} \leftarrow u_{i,j,k} + \eta \frac{(x_j^{[t]} - \mu_j^{[t]})}{\sigma_j^2} \alpha_{i,j,k}^{[t-1]} \tag{17}$$

$$A_{l,j} \leftarrow A_{l,j} + \eta' \frac{(x_j^{[t]} - \mu_j^{[t]})}{\sigma_j^2} \psi_l^{[t]}, \tag{18}$$

for $k = 1,\ldots,K$, $\delta = 1,\ldots,d_{i,j}-1$, and $(i,j) \in \{1,\ldots,N\}$, where η and η' are learning rates. We set, $\eta' < \eta$ such that A_{lj} is stationary while other parameters are updated. The learning rates can be adjusted at each step according to stochastic optimization techniques like Adam (Kingma and Ba 2015) and RMSProp (Tieleman and Hinton 2012). In (14)-(18), $\mu_j^{[t]}$ is given by (5), where $\alpha_{i,j,k}^{[t-1]}$ and $x_i^{[t-\delta]}$ for $\delta \in [1, d_{i,j}-1]$, respectively, are stored and updated in a synapse and a FIFO queue that connects from neuron i to neuron j. It should be noted that, in the absence of the formulations in (9) and (10), this same learning procedure updates the parameters of an equivalent G-DyBM model, without the need for (18). See algorithmic description in *supplementary* (Dasgupta and Osogami 2016).

Numerical experiments

We now demonstrate the advantages of the RNN-G-DyBM through numerical experiments with two synthetic and two real time series data. In these experiments, we use the RNN-G-DyBM with a common conduction delay ($d_{i,j} = d$ for any i,j; see (8)). All the experiments were carried out with a Python 2.7 implementation (with numpy and theano backend) on a Macbook Air with Intel Core i5 and 8 GB of memory.

In all cases we train the RNN-G-DyBM in an online manner. Namely, for each step t, we give a pattern, $\mathbf{x}^{[t]}$, to the network to update its parameters and variables such as eligibility traces (see (7)), and then let theRNN-G-DyBM predict the next pattern, $\tilde{\mathbf{x}}^{[t+1]}$, based on $\mu^{[t+1]}$ using (8)-(10). This process is repeated sequentially for all time or observation points $t = 1,2,\ldots$. Here, the parameters are updated according to (14)-(18). The learning rates, η and η', in (14)-(18) is adjusted for each parameter according to RMSProp (Tieleman and Hinton 2012), where the initial learning rate was set to 0.001. Throughout, the initial values of the parameters and variables, including eligibility traces and the spikes in the FIFO queues, are set to 0. However, we initialize $\sigma_j = 1$ for each j to avoid division by 0 error. All RNN weight matrices were initialized randomly as described in the RNN-G DyBM model section. The RNN layer leak rate ρ was set to 0.9 in all experiments.

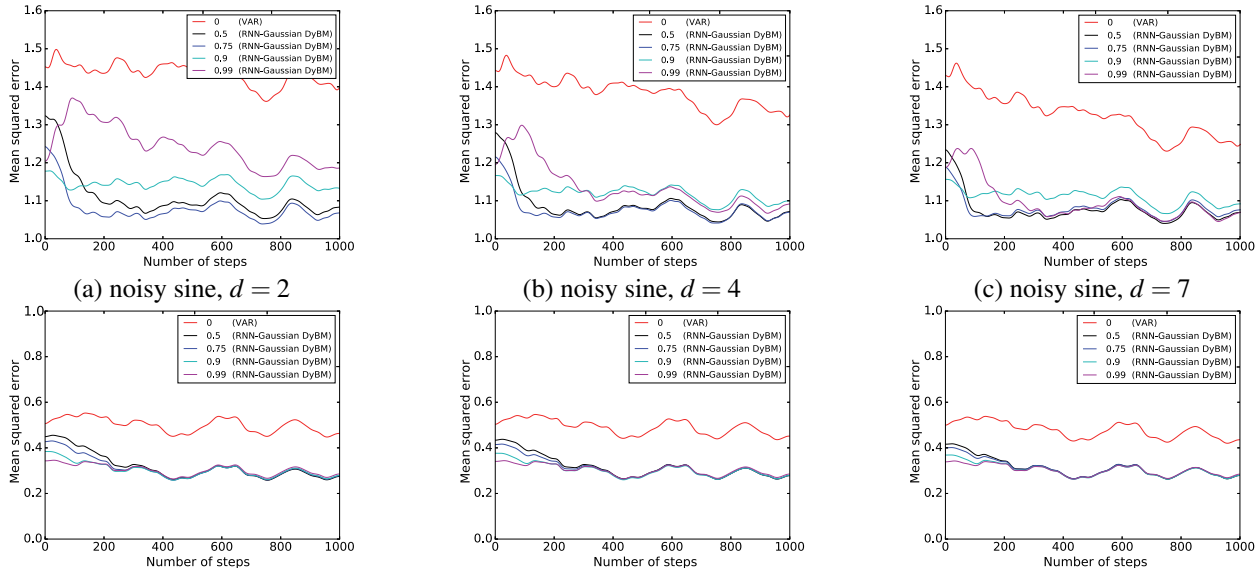
Synthetic time series prediction

The purpose of the experiments with the synthetic datasets is to clearly evaluate the performance of RNN-G-DyBM in a controlled setting. Specifically, we consider a RNN-G-DyBM with N DyBM units and M RNN units. The DyBM units are connected with FIFO queues of length d and eligibility traces of decay rate λ, where d and λ are varied in the experiments. For $\lambda = 0$, and in the absence of the RNN layer, we have $\alpha^{[t]} = x^{[t-d]}$, and this Gaussian DyBM reduces to a vector autoregressive model with d lags. We use this VAR model as the *baseline* for performance evaluation.

Multidimensional noisy sine wave: In the first synthetic task, we train the RNN-G-DyBM with a five dimensional noisy sine-wave. Where each dimension x_D is generated as:

$$x_D^{[t]} = \sin(D\pi t/100) + \varepsilon^{[t]}, D = (1,2,3,4,5), \tag{19}$$

for each t, where $\varepsilon^{[t]}$ is independent and identically distributed (i.i.d) with a Gaussian distribution $\mathscr{N}(0,1)$. The number of DyBM units $N = 5$ with $M = 10$ hidden RNN units.



(a) noisy sine, $d = 2$ (b) noisy sine, $d = 4$ (c) noisy sine, $d = 7$

(d) narma-30, $d = 2$ (e) narma-30, $d = 4$ (f) narma-30, $d = 7$

Figure 2: Mean squared error of prediction for the synthetic time series. For each step t, the mean squared error is averaged over 100 independent runs. Decay rate λ is varied as in the legend, and the red curve ($\lambda = 0$) corresponds to the baseline VAR model. Conduction delay d is varied across panels. (a)-(c) Prediction performance on the noisy sine task. (d)-(f) Prediction performance on the 30th order NARMA task.

30th order nonlinear autoregressive moving average (NARMA): In this task, we train the RNN-G-DyBM for one step prediction with a one dimensional nonlinear time-series. Namely, the 30th order NARMA (Connor, Atlas, and Martin 1992; Jaeger and Haass 2004) which is generated as:

$$x^{[t]} = 0.2x^{[t-1]} + 0.004x^{[t-1]}\left[\sum_{i=0}^{29} x^{[t-1-i]}\right] + 1.5u^{[t-30]}u^{[t-1]} + 0.01 \quad (20)$$

Where, $u^{[t]}$ is i.i.d with a Gaussian distribution $\mathcal{N}(0, 0.5)$. As such, for future prediction, this task requires the modeling of the inherent nonlinearity and up to 30 time-steps of memory. The number of DyBM units are $N = 1$ with $M = 5$ hidden RNN units.

Figure 2 shows the predictive accuracy of the RNN-G-DyBM. Here, the prediction, $\tilde{\mathbf{x}}^{[t]}$, for the pattern at time t is evaluated with mean squared error, $\text{MSE}^{[t]} \equiv \frac{1}{100}\sum_{s=t-50}^{50}(\tilde{\mathbf{x}}^{[t]} - \mathbf{x}^{[t]})^2$, and $\text{MSE}^{[t]}$ is further averaged over 100 independent runs of the experiment. Due to the time-dependent noise $\varepsilon^{[t]}$, the best possible squared error is 1.0 in expectation. We vary decay rates λ as indicated in the legend and d as indicated below each panel. We observe that in the noisy sine task, although the prediction accuracy depends on the choice of λ, Figure 2 (a)-(c) shows that the RNN-G-DyBM (with $\lambda > 0$) significantly outperforms VAR model ($\lambda = 0$; red curves) and reduces the error by more than 30 %. A similar performance benefit is also observed in Figure 2 (d)-(f) for the 30th order NARMA prediction task. Where in, the RNN-G-DyBM performs robustly across varying decay rates and consistently outperforms the VAR model even when the lag or delays increases. As this task requires both long memory and nonlinearity, the gain RNN-G-DyBM has over the VAR stems from the use of the eligibility traces, $\alpha^{[t]}$ and the nonlinear RNN hidden units, instead of the lag-d variable, $x^{[t-d]}$. As such, even with increasing delays the VAR model does not match the performance of RNN-G-DyBM. The results for even longer delays can be found in the *supplementary material* (Dasgupta and Osogami 2016).

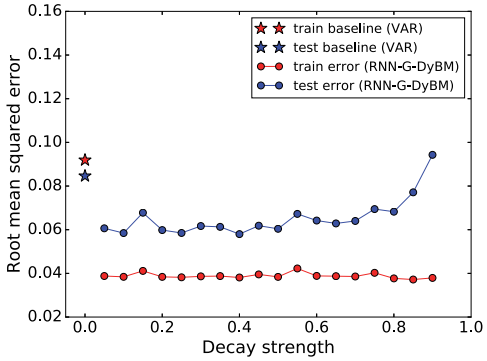


Figure 3: Average root mean squared error after 20 epochs for one-week ahead prediction plotted for varying decay rates (λ), on the real dataset of weekly retail prices for gasoline and diesel in U.S.A. The results shown are using fixed delay strength $d = 2$. The star marks show the performance of the baseline VAR model ($\lambda = 0.0$). Training error is plotted in red and test error is plotted in blue. *See supplementary results* (Dasgupta and Osogami 2016), for plots with $d = 3$ and $d = 4$.

MODEL	Retail-price prediction Test RMSE	Sunspots prediction Test RMSE
LSTM	0.0674	**0.07342**
VAR (d=2)	0.0846	0.2202
VAR (d=3)	0.0886	0.1859
VAR (d=4)	0.0992	0.2050
RNN-G-DyBM (d=2)	0.0580	0.08252
RNN-G-DyBM (d=3)	**0.0564**	**0.0770**
RNN-G-DyBM (d=4)	0.0605	0.0774

Table 1: The average test RMSE after 20 epochs for various models for online time series prediction tasks with the retail-price dataset and sunspot number datasets, respectively. The delay (d) for each model is in brackets. For LSTM model there was no delay and decay rate (λ) hyper-parameters. In all VAR models $\lambda = 0.0$. For RNN-G-DyBM models the best test score achieved across the entire range of decay rates is reported.

MODEL	Average runtime/epoch (s)
LSTM	11.2132
RNN-G-DyBM (across delays)	0.7014
VAR (across delays)	0.3553

Table 2: The average CPU time taken in seconds, to train a model per epoch. The reported values are for the sunspot nonlinear time series prediction task.

Real-data time series prediction

Having evaluated the performance of RNN-G-DyBM in a controlled setting, we now evaluate its performance on two real-data sets. We demonstrate that RNN-G-DyBM consistently outperforms the baseline VAR models as well as, betters in some cases, the performance obtained with the popular LSTM RNN model. We first evaluate using a highly non-stationary multidimensional retail-price time series dataset.

Weekly retail gasoline and diesel prices in U.S.[3]**:** This dataset consists of real valued time series of 1223 steps($t = 1,\ldots,1223$) of weekly (April 5th, 1993 to September 5th, 2016) prices of gasoline and diesel (in US dollar/gallon) with 8 dimensions covering different regions in U.S.A. We normalize the data within the interval $[0,1.0]$. We use the first 67% of the time series observations (819 time steps) as the training set and the remaining 33% (404 time steps) as the test data set. We train the RNN-G-DyBM with $N = 8$ units, and $M = 20$ hidden RNN units with varying d and λ. The objective in this task was to make one-week ahead predictions in an online manner, as in the synthetic experiments. We once again use the VAR model as the baseline. Additionally, we also learn with LSTM RNN (Gers, Schmidhuber, and Cummins 2000) with 20 hidden units. We set the relevant hyper-parameters such that they were consistent across all the models. In all models the parameters of the network was updated online, using the stochastic optimization method RMSProp (Tieleman and Hinton 2012) with an initial learning rate of 0.001. LSTM was implemented in Keras *see supplementary* (Dasgupta and Osogami 2016).

[3]Data obtained from http://www.eia.gov/petroleum/

Monthly sunspot number prediction[4]**:** In the second task, we use the historic benchmark of monthly sunspot number (Hipel and McLeod 1994) collected in Zurich from Jan. 1749 to Dec. 1983. This is a one-dimensional nonlinear time series with 2820 time steps. As before, we normalize the data within the interval $[0,1.0]$ and used 67% for training and 33% for testing the models. With goal of monthly prediction we trained RNN-G-DyBM with $N = 1$ unit, and $M = 50$ hidden RNN units.The LSTM model also had 50 hidden units. All other settings were same as the other real-data set.

Figure 3 shows the one-week ahead prediction accuracy of RNN-G-DyBM on the retail-price time series for delay $d = 2$. We evaluate the error using the root mean squared error (RMSE) measure averaged over 20 epochs calculated for the normalized time series. The figure shows that the RNN-G-DyBM (solid curves) clearly outperforms VAR (stared points) by nearly more than 30% (depending on the settings), on both training and test predictions. However for larger decay rates the test RMSE increases suggesting that over-fitting can occur if hyper-parameters are not selected properly. In comparison, while directly using the G-DyBM model on this task, the best case test RMSE was 0.0718 with $d = 3$, thus RNN-G-DyBM improves upon G-DyBM by more than 21% .

As observed in Table 1, on this task the RNN-G-DyBM with $d = 3$ achieved the best performance, which remarkably beats even the LSTM model by a margin of ≈ 16 %. Considerable performance gain against the VAR baseline was also observed (Table 1. columns two) using the sunspot time series. Due to the inherent nonlinearity in this data, both the VAR and G-DyBM models perform very poorly even for higher delays. Notably, the best case test RMSE obtained when using the G-DyBM model was 0.1327 (with $d = 3$) i.e. 40% lower as compared to the best RNN-G-DyBM model. With 50 hidden units the LSTM model performs slightly better in this case with a normalized test error of 0.07342 as compared to the 0.0770 for the RNN-G-DyBM with $d = 3$ (comparison of the true vs predicted time series on this task for all the three models can be seen in the *supplementary data* (Dasgupta and Osogami 2016). Visual inspection shows little difference between RNN-G-DyBM and LSTM prediction.).

In Table 2. we record the average CPU time taken (in seconds) to execute a single training epoch on the sunspot data, across the three models. As observed, the RNN-G-DyBM not only achieves comparable performance to the LSTM but learns in only 0.7014 sec./epoch as compared to the 11.2132 sec./epoch (16 times more) for the LSTM model. Notably, after 50 epochs LSTM achieves a best test RMSE of 0.0674(retail-price dataset) taking ≈ 566 secs., while after 50 epochs the RNN-G-DyBM realises a best test RMSE of 0.0564 in ≈ 35 secs, on the same task. As such, the nonlinear RNN-G-DyBM model is highly scalable in an online learning environment. Expectedly, the VAR model without any eligibility traces and hidden units runs much faster, albeit with significantly lower predictive accuracy.

[4]Publicly available at https://datamarket.com/data/set/22t4/

Conclusion

In this paper we first extended the dynamic Boltzmann machine (DyBM), into the Gaussian DyBM (G-DyBM) to model real valued time series. The G-DyBM can be seen as an extension of vector autoregression model. We further extended this to the RNN-Gaussian DyBM in order to model inherent nonlinearities in time series data. Experimental results demonstrate the effectiveness of the RNN-G-DyBM model with significant performance gain over VAR. The RNN-G-DyBM also outperforms popular LSTM models at a considerably reduced computational cost. Our model is highly scalable similar to binary DyBM (Dasgupta, Yoshizumi, and Osogami 2016) that was shown to give significant performance improvement on the high-dimensional moving MNIST task. Furthermore, unlike models requiring back-propagation, in RNN-G-DyBM each parameter can be updated in a distributed manner in constant time with Eqs.14-18. This update is independent of data dimension or the maximum delay. This makes the RNN-G-DyBM model highly robust and scalable for online high-dimensional time series prediction scenarios.

Acknowledgments

This work was partly funded by CREST, JST.

References

Ackley, D. H.; Hinton, G. E.; and Sejnowski, T. J. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science* 9:147169.

Bahadori, M. T.; Liu, Y.; and Xing, E. P. 2013. Fast structure learning in generalized stochastic processes with latent factors. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 284–292. ACM.

Connor, J.; Atlas, L. E.; and Martin, D. R. 1992. Recurrent networks and NARMA modeling. In *Advances in Neural Information Processing Systems 4*. 301–308.

Dasgupta, S., and Osogami, T. 2016. Nonlinear dynamic boltzmann machines for time-series prediction. Technical Report RT0975, IBM Research. http://ibm.biz/NDyBMSup.

Dasgupta, S.; Yoshizumi, T.; and Osogami, T. 2016. Regularized dynamic boltzmann machine with delay pruning for unsupervised learning of temporal sequences. In *Proceedings of the 23rd International Conference on Pattern Recognition* (ICPR).

Gers, A. F.; Schmidhuber, J.; and Cummins, F. 2000. Learning to forget: Continual prediction with lstm. *Neural computation* 12(10):2451–2471.

Hebb, D. O. 1949. *The organization of behavior: A neuropsychological approach*. Wiley.

Hinton, G. E., and Brown, A. D. 1999. Spiking Boltzmann machines. In *Advances in Neural Information Processing Systems 12*. 122128.

Hinton, G. E., and Salakhutdinov, R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313:504507.

Hinton, G. E., and Sejnowski, T. J. 1983. Optimal perceptual inference. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 448–453.

Hipel, W. K., and McLeod, A. I. 1994. *Time series modelling of water resources and environmental systems*, volume 45. Elsevier.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Jaeger, H., and Haass, H. 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science* 304(5667):78–80.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations* (ICLR), arXiv:1412.6980.

Lütkepohl, H. 2005. *New introduction to multiple time series analysis*. Springer Science & Business Media.

Marks, T., and Movellan, J. 2001. Diffusion networks, products of experts, and factor analysis. In *Proceedings of the Third International Conference on Independent Component Analysis and Blind Source Separation*.

Mittelman, R.; Kuipers, B.; Savarese, S.; and Lee, H. 2014. Structured recurrent temporal restricted Boltzmann machines. In *Proc. 31st Annual International Conference on Machine Learning (ICML 2014)*, 16471655.

Osogami, T., and Otsuka, M. 2015a. Learning dynamic Boltzmann machines with spike-timing dependent plasticity. Technical Report RT0967, IBM Research.

Osogami, T., and Otsuka, M. 2015b. Seven neurons memorizing sequences of alphabetical images via spike-timing dependent plasticity. *Scientific Reports* 5:14149.

Sutskever, I., and Hinton, G. E. 2007. Learning multilevel distributed representations for high-dimensional sequences. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS-07)*, volume 2, 548555.

Sutskever, I.; Hinton, G. E.; and Taylor, G. W. 2009. The recurrent temporal restricted Boltzmann machine. In *Advances in Neural Information Processing Systems 21*. 16011608.

Taylor, G. W., and Hinton, G. E. 2009. Factored conditional restricted Boltzmann machines for modeling motion style. In *Proc. 26th Annual International Conference on Machine Learning (ICML 2009)*, 10251032.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning* 4(2).

Welling, M.; Rosen-Zvi, M.; and Hinton, G. E. 2004. Exponential family harmoniums with an application to information retrieval. In *Advances in Neural Information Processing Systems 17*. 1481–1488.

Zhang, G.; Patuwo, B. E.; and Hu, M. Y. 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* 14(1):3562.