# A Functional Dynamic Boltzmann Machine

**Hiroshi Kajino**
IBM Research - Tokyo
KAJINO@jp.ibm.com

## Abstract

Dynamic Boltzmann machines (DyBMs) are recently developed generative models of a time series. They are designed to learn a time series by efficient online learning algorithms, whilst taking long-term dependencies into account with help of eligibility traces, recursively updatable memory units storing descriptive statistics of all the past data. The current DyBMs assume a finite-dimensional time series and cannot be applied to a functional time series, in which the dimension goes to infinity (*e.g.*, spatiotemporal data on a continuous space). In this paper, we present a *functional dynamic Boltzmann machine (F-DyBM)* as a generative model of a functional time series. A technical challenge is to devise an online learning algorithm with which F-DyBM, consisting of functions and integrals, can learn a functional time series using only finite observations of it. We rise to the above challenge by combining a kernel-based function approximation method along with a statistical interpolation method and finally derive closed-form update rules. We design numerical experiments to empirically confirm the effectiveness of our solutions. The experimental results demonstrate consistent error reductions as compared to baseline methods, from which we conclude the effectiveness of F-DyBM for functional time series prediction.

## 1 Introduction

This work is concerned with learning a time series for forecasting future events. In particular, we are focusing on a light-weight model that can be trained online while preserving the predictive performance as much as possible. Aside from recent high-performance but complex models like a family of recurrent neural networks (RNNs) [Rumelhart *et al.*, 1986; Hochreiter and Schmidhuber, 1997; Sundermeyer *et al.*, 2012], light-weight models are required when training and predicting on devices with low computational power such as mobile and IoT devices and when dealing with massive amount of stream data reported from a number of sensors. In this light, we focus on a family of vector autoregressive models (VARs) [Lütkepohl, 2005], and above all, one of its state-of-the-art variants called *dynamic Boltzmann machines* (DyBMs) [Osogami and Otsuka, 2015; Osogami, 2016; Dasgupta and Osogami, 2017].

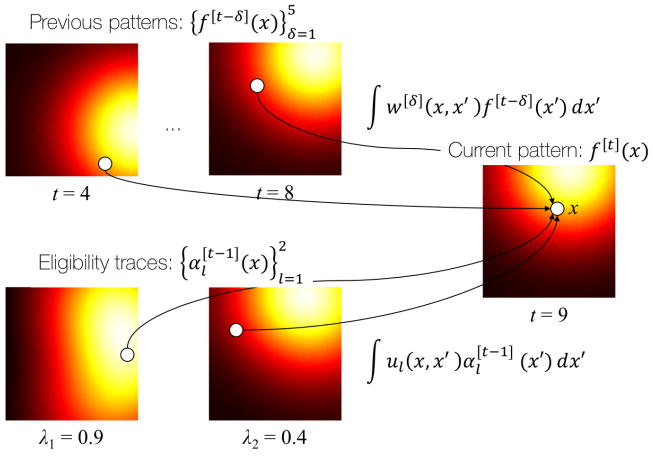


Figure 1: Illustration of F-DyBM, modeling a functional pattern $f^{[t]}(x)$ defined on a two dimensional space. Heat maps represent functional patterns. The current pattern depends on five past patterns and two eligibility traces (which summarize all the past patterns) through weight functions $w^{[\delta]}$ and $u_l$, respectively.

DyBMs are recently emerging generative models of a binary/real-valued multi-dimensional time series. One of their essential characteristics is a recursively updatable memory unit summarizing all the past data, which is dubbed as an *eligibility trace*. Its recursive update rule enables us to develop online learning algorithms for DyBMs while capturing long-term dependencies of a time series to increase its predictive ability. Osogami [2016] reported up to 20-30% performance gain by eligibility traces. Therefore, we employ DyBMs as a time-series modeling framework.

In this paper, we present a new variant of DyBMs called a *functional dynamic Boltzmann machine (F-DyBM)*, which is able to handle a partially-observable functional time series, where at each discrete time step $t \in \mathbb{Z}$, finite evaluations of a function $f^{[t]}(x)\colon \mathcal{X} \to \mathbb{R}$ are given. Our F-DyBM is mainly motivated by spatiotemporal data. Assume that a functional time series is a spatiotemporal time series collected from mo-

bile devices, where $f^{[t]}(x)$ corresponds to a temperature, water quality, or air quality observed at location $x \in \mathcal{X}$ and time step $t$. This example implies two properties that cannot be handled by the existing DyBMs designed for a vectorial time series. First, it is required to forecast a future value at *any* location $x$ in a continuous space $\mathcal{X}$, rather than finite fixed locations as in the case with a vectorial time series, in which each dimension corresponds to each fixed location. The existing DyBMs are not able to forecast values at infinite locations by themselves. Second, observational points can move and may appear and disappear abruptly, and in some cases, the identities of observations are lost due to privacy concerns, inhibiting us from constructing a vectorial time series. These two properties clearly highlight the essential difference between a vectorial time series and a functional one. To this end, we develop F-DyBM along with an online learning algorithm.

The development of F-DyBM constitutes of a modeling task and an algorithm implementation task. The model of F-DyBM can be derived rather straightforwardly based on the Gaussian DyBM (G-DyBM) [Osogami, 2016; Dasgupta and Osogami, 2017]; replacing vectors with functions, weight matrices with weight functions, and matrix-vector multiplications with integrals. As a result, we obtain a model of a functional time series as depicted in Fig. 1. On contrary, a learning algorithm cannot be directly derived in the same way as modeling because of the following three technical challenges. First, neither a functional time series nor weight functions can be represented in a computer having only a finite memory. Second, the model now involves integrals, which in general cannot be efficiently computed. Third, typically, only finite observations of a functional pattern are available at each time, which breaks the fully-observable assumption in G-DyBM

Our idea to overcome these difficulties is twofold. The first idea, addressing the first and second challenges, is to model weight functions by finite kernel-based basis functions. Then, we are able to represent model parameters by finite parameters, and furthermore, the kernel trick allows us to analytically compute the integrals, finally resulting in closed-form learning rules. Second, to address the third challenge, we employ a Gaussian process as a core component of our model, and estimate a functional pattern from finite observations by the maximum-a-posteriori (MAP) estimation. These ideas successfully lead to an online learning algorithm for F-DyBM.

The effectiveness of F-DyBM is empirically demonstrated using five real spatiotemporal data sets. As we will discuss in the related work section, F-DyBM can be interpreted as an extension of VAR and functional autoregression (FAR) [Bosq, 2000] as well as G-DyBM; eligibility traces mainly differentiate F-DyBM from FAR and G-DyBM from VAR, and rigorous modeling of a functional time series differentiates F-DyBM from G-DyBM and FAR from VAR. Therefore, we design the experiment to validate the contribution of each of these innovations. The experimental results indicate that adding eligibility traces decreased the error by $12\%$ on average, and the function-based modeling decreased the error by $11.7\%$ on average as compared to a heuristic application of vector-based models. Hence, we conclude that F-DyBM achieves substantial performance improvement because of the two features.

**Notation.** We employ the following mathematical conventions. For a matrix $X = [x_1 \ \dots \ x_N]^\top \in \mathbb{R}^{N\times D}$ and a function $f\colon \mathbb{R}^D \to \mathbb{R}$, we define $f(X) := [f(x_1) \ \dots \ f(x_N)]^\top \in \mathbb{R}^N$. For matrices $X$ and $Y = [y_1 \ \dots \ y_M]^\top \in \mathbb{R}^{M\times D}$ and a function $K\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$, we define $K(X,Y)$ as an $N \times M$ matrix whose $(n,m)$-element corresponds to $K(x_n, y_m)$. Let us denote a value associated with time $t$ by $f^{[t]}$. A sequence of values from time $-\infty$ to $t-1$ (including $t-1$) is denoted by $f^{[<t]}$.

## 2 Problem Setting

We assume that observations are carried out in $\mathcal{X}$ (*e.g.*, the Euclidean space in the case of a standard spatiotemporal data). Let us denote a signal observed at $x \in \mathcal{X}$ at time $t \in \mathbb{Z}$ by $f^{[t]}(x) \in \mathbb{R}$. We call a function $f^{[t]}$ as a *pattern*. Our problem setting is that, at every time $t$, we are given finite observations of a pattern $f^{[t]}$, and are willing to predict the next pattern $f^{[t+1]}$. Let us denote the observational points at time $t$ by $X^{[t]} = \left[x_1^{[t]} \ \dots \ x_{N^{[t]}}^{[t]}\right]^\top$. The observational points as well as the number of them, $N^{[t]}$, can be time-variant.

## 3 Gaussian Dynamic Boltzmann Machine

This section briefly reviews a Gaussian dynamic Boltzmann machine (G-DyBM) [Osogami, 2016; Dasgupta and Osogami, 2017], which extends the original binary-valued DyBM [Osogami and Otsuka, 2015] to handle real values.

G-DyBM models an $N$-dimensional time series by $N$ neural units. The $n$-th unit ($n = 1, \dots, N$) outputs a signal $f_n^{[t]} \in \mathbb{R}$ at time $t$. We call the set of signals emitted from all the neurons a pattern, which is denoted by $\mathbf{f}^{[t]} = \left[f_1^{[t]} \ \dots \ f_N^{[t]}\right]^\top$. G-DyBM models the dynamics of the pattern as follows:

$$p(\mathbf{f}^{[t]} \mid \mathbf{f}^{[<t]}) = \prod_{n=1}^{N} \mathcal{N}(f_n^{[t]}; \mu_n^{[t]}, \sigma_n^2), \tag{1}$$

$$\boldsymbol{\mu}^{[t]} = \mathbf{b} + \sum_{\delta=1}^{d} W^{[\delta]} \mathbf{f}^{[t-\delta]} + \sum_{l=1}^{L} U_l \boldsymbol{\alpha}_l^{[t-1]}, \tag{2}$$

$$\boldsymbol{\alpha}_l^{[t-1]} = \sum_{\delta=1}^{\infty} \lambda_l^{\delta-1} \mathbf{f}^{[t-\delta]} \ (l = 1, \dots, L), \tag{3}$$

where parameters are $\mathbf{b}$, $\{W^{[\delta]}\}_{\delta=1}^d$, and $\{U_l\}_{l=1}^L$, and hyperparameters are $d$, $L$, $\{\sigma_n\}_{n=1}^N$, and $\{\lambda_l\}_{l=1}^L$.[1] The pattern at time $t$ is dependent on all the past patterns through the mean $\boldsymbol{\mu}^{[t]} = \left[\mu_1^{[t]} \ \dots \ \mu_N^{[t]}\right]^\top$ (Eq. (1)). The mean at time $t$ is calculated using two types of memory units (Eq. (2)). One stores the recent $d$ patterns $\{\mathbf{f}^{[t-\delta]}\}_{\delta=1}^d$, and the other called an *eligibility trace*, $\boldsymbol{\alpha}_l^{[t-1]}$ ($l = 1, \dots, L$) summarizes all the past patterns (Eq. (3)).

[1] We fix $\sigma_n = 1$ for all $n$.

# 4 Functional Dynamic Boltzmann Machines

We present a functional dynamic Boltzmann machine (F-DyBM), which models the dynamics of a *functional* pattern emitted from an infinite number of neurons distributed in $\mathcal{X}$. We first present the model of F-DyBM, and then, we develop an online learning algorithm, which trains F-DyBM from finite observations of a functional pattern and enables us to forecast the next functional pattern.

## 4.1 Model

We assume that each neuron is associated with location $x \in \mathcal{X}$, instead of index $n$. Let us denote a pattern at time $t$ by $f^{[t]}\colon \mathcal{X} \to \mathbb{R}$. $f^{[t]}(x)$ represents a signal of a neuron at $x$ and time $t$. We derive F-DyBM by extending Eqs. (1), (2), and (3) to Eqs. (4), (5), and (6) respectively as follows.

Equation (1) is refined by substituting a Gaussian process for the Gaussian distribution:

$$p(f^{[t]} \mid f^{[<t]}) = \mathcal{GP}(f^{[t]}; \mu^{[t]}, K_{\sigma^2}), \tag{4}$$

where $f^{[t]}$ is a functional pattern, $\mu^{[t]}\colon \mathcal{X} \to \mathbb{R}$ is the mean function at time $t$, which is dependent on all the past patterns $f^{[<t]}$, and $K_{\sigma^2}\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a covariance function.[2] Equations (2) and (3) are refined as follows:

$$\begin{aligned}\mu^{[t]}(x) =& b(x) + \sum_{\delta=1}^{d} \int_{\mathcal{X}} w^{[\delta]}(x, x') f^{[t-\delta]}(x') \mathrm{d}x' \\ &+ \sum_{l=1}^{L} \int_{\mathcal{X}} u_l(x, x') \alpha_l^{[t-1]}(x') \mathrm{d}x',\end{aligned} \tag{5}$$

$$\alpha_l^{[t-1]}(x) = \sum_{\delta=1}^{\infty} \lambda_l^{\delta-1} f^{[t-\delta]}(x), \tag{6}$$

where we substitute weight functions for weight matrices and integrals for matrix multiplications. Model parameters are $b(x)$, $\{w^{[\delta]}(x, x')\}_{\delta=1}^{d}$, and $\{u_l(x, x')\}_{l=1}^{L}$, and hyperparameters are $d$, $L$, $K_{\sigma^2}$, and $\{\lambda_l\}_{l=1}^{L}$.

## 4.2 Online Learning Algorithm

So far, we have successfully compiled G-DyBM for a functional pattern. However, as discussed in the introduction, there still remain the following three technical challenges to deploy F-DyBM into the real world:

**(i)** The model involves functional parameters, $b$, $w^{[\delta]}$, and $u_l$, as well as functional patterns $f^{[t]}(x)$, which cannot be stored using finite memory.

**(ii)** Equation (5) involves integrals of the weight functions and previous patterns. In general, numerical integration is an error-prone and time-consuming procedure, and should be avoided whenever possible.

**(iii)** Our problem setting assumes that only finite observations of a pattern are available at each time step.

[2] We employ the following shorthand: $K_{\sigma^2}(x, x') := K(x, x') + \sigma^2 \delta_{x,x'}$ with denoting the Kronecker delta by $\delta_{x,x'}$.

**Main Ideas**

We introduce two main ideas to address the three technical challenges raised above.

The first idea is to assume a particular parametric form for functional parameters using finite parameters along with the covariance function $K_{\sigma^2}$ of the Gaussian process. Then, we can represent all of the functional parameters by finite-dimensional vectors, and in addition, the kernel trick allows us to avoid numerical integration, addressing both challenges, **(i)** and **(ii)**. In specific, weight functions, $w^{[\delta]}(x, x')$ and $u_l(x, x')$, and the bias function $b(x)$ are assumed to have the following parametric forms:

$$w^{[\delta]}(x, x') = K_{\sigma^2}(x, P) W^{[\delta]} K_{\sigma^2}(P, x') \ (\delta = 1, \ldots, d), \tag{7}$$

$$u_l(x, x') = K_{\sigma^2}(x, P) U_l K_{\sigma^2}(P, x') \ (l = 1, \ldots, L), \tag{8}$$

$$b(x) = K_{\sigma^2}(x, P)\mathbf{b}, \tag{9}$$

where $P = [p_1 \ \ldots \ p_N]^\top$ is a set of points in $\mathcal{X}$, and $W^{[\delta]}, U_l \in \mathbb{R}^{N \times M}$ and $\mathbf{b} \in \mathbb{R}^N$ are parameters. Then, by substituting Eqs. (7), (8), and (9) for Eq. (5), we obtain

$$\begin{aligned}\mu^{[t]}(x) = K_{\sigma^2}(x, P) \Bigg[ \mathbf{b} &+ \sum_{\delta=1}^{d} W^{[\delta]} f^{[t-\delta]}(P) \\ &+ \sum_{l=1}^{L} U_l \alpha_l^{[t-1]}(P) \Bigg],\end{aligned} \tag{10}$$

where the integrals can be analytically computed via the kernel trick because $f^{[t-\delta]}(x)$ and $\alpha_l^{[t-1]}(x)$ belong to the reproducing Hilbert kernel space of $K_{\sigma^2}(\cdot, \cdot)$ (see Eq. (4)). Thus, the two challenges **(i)** and **(ii)** are addressed.

Another benefit of the resultant model is that it is sufficient to memorize patterns at the *fixed* set of points, $P$, for computing Eq. (10), rather than the *time-varying* points $X^{[t]}$. This is especially significant when computing the eligibility trace using fixed-sized memory. In fact, the eligibility trace $\alpha_l^{[t-1]}(P)$ can be defined in a similar way to Eq. (3):

$$\alpha_l^{[t-1]}(P) = \sum_{\delta=1}^{\infty} \lambda_l^{\delta-1} f^{[t-\delta]}(P), \tag{11}$$

which is also recursively updatable.

The second idea addressing the third challenge **(iii)**, is to replace the functional patterns appearing in Eqs. (10) and (11) with their estimators. At time $t$, given $f^{[t]}(X^{[t]})$ and $\mu^{[t]}(x)$, the MAP estimator of $f^{[t]}(P)$ is obtained as

$$\hat{f}^{[t]}(P) = \mu^{[t]}(P) + K(P, X^{[t]}) K_{\sigma^2}(X^{[t]}, X^{[t]})^{-1} d^{[t]}(X^{[t]}), \tag{12}$$

where we denote $d^{[t]}(x) := f^{[t]}(x) - \mu^{[t]}(x)$. At each time $t$, we compute the MAP estimator $\hat{f}^{[t]}(P)$ as Eq. (12) and use it as a proxy for $f^{[t]}(P)$ in Eq. (10) from time $t+1$ onward. We choose to impute unobserved values by their estimators progressively, instead of relying on EM-based algorithms (*e.g.*, [Shumway and Stoffer, 1982]), in order to keep the online algorithm as efficient as possible.

**Algorithm 1** An online learning algorithm for F-DyBM.

1: **for** $t = 0, 1, 2, \ldots$ **do**
2: Predict the pattern at time $t$ using $\mu^{[t]}(x)$ as Eq. (10).
3: Observe the pattern at time $t$, $(X^{[t]}, f^{[t]}(X^{[t]}))$.
4: Update parameters as Eqs. (13)–(16).
5: Compute $\hat{f}^{[t]}(P)$ as Eq. (12).
6: Update $\mathcal{Q}$ and $\mathcal{E}$ using $\hat{f}^{[t]}(P)$ as Eq. (17).
7: **end for**

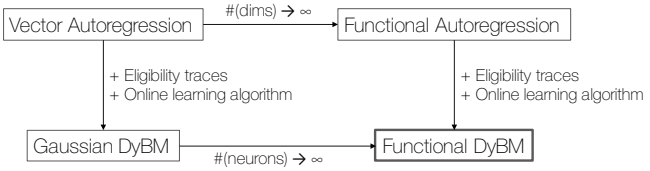


Figure 2: Relationship diagram among VAR, FAR, G-DyBM, and F-DyBM.

**Implementation**

Putting everything together, we derive an online learning algorithm for F-DyBM (Algorithm 1). It learns $\mathbf{\Theta} = \{\mathbf{b}\} \cup \{W^{[\delta]}\}_{\delta=1}^{d} \cup \{U_l\}_{l=1}^{L}$ in an online way, and enables us to predict the next unseen pattern. Its input is hyperparameters, $d$, $L$, $P \in \mathbb{R}^{N \times D}$, $K_{\sigma^2}$, and $\{\lambda_l\}_{l=1}^{L}$. First, two memory units, $\mathcal{Q}$ and $\mathcal{E}$, and the model parameters $\mathbf{\Theta}$ are initialized by filling 0, where $\mathcal{Q}$ is a queue storing $\{\hat{f}^{[t-\delta]}(P)\}_{\delta=1}^{d}$, and $\mathcal{E}$ is a set of eligibility traces $\{\alpha_l^{[t-1]}(P)\}_{l=1}^{L}$. For each time $t = 0, 1, 2, \ldots$, we first predict the next pattern as Eq. (10). Then, we observe $(X^{[t]}, f^{[t]}(X^{[t]}))$ and update $\mathbf{\Theta}$ as

$$\mathbf{\Theta} \leftarrow \mathbf{\Theta} + \eta^{[t]} \cdot \frac{\partial \mathcal{L}^{[t]}(\mathbf{\Theta})}{\partial \mathbf{\Theta}}, \tag{13}$$

where $\eta^{[t]}$ is a learning rate, and $\mathcal{L}^{[t]}(\mathbf{\Theta})$ is a conditional log-likelihood of the current observations, which is given by

$$\begin{aligned}\mathcal{L}^{[t]}(\mathbf{\Theta}) &:= \log p(f^{[t]}(X^{[t]}) \mid f^{[<t]}; \mathbf{\Theta}) \\ &= -\frac{1}{2} d^{[t]}(X^{[t]})^{\top} K_{\sigma^2}(X^{[t]}, X^{[t]})^{-1} d^{[t]}(X^{[t]}) + C,\end{aligned}$$

where $C$ is a constant independent of $\mathbf{\Theta}$. The gradients of $\mathcal{L}^{[t]}(\mathbf{\Theta})$ with respect to the model parameters are,

$$\frac{\partial}{\partial \mathbf{b}} \mathcal{L}^{[t]}(\mathbf{\Theta}) = K(P, X^{[t]}) K_{\sigma^2}(X^{[t]}, X^{[t]})^{-1} d^{[t]}(X^{[t]}), \tag{14}$$

$$\frac{\partial}{\partial W^{[\delta]}} \mathcal{L}^{[t]}(\mathbf{\Theta}) = \left[\frac{\partial}{\partial \mathbf{b}} \mathcal{L}^{[t]}(\mathbf{\Theta})\right] \hat{f}^{[t-\delta]}(P)^{\top}, \tag{15}$$

$$\frac{\partial}{\partial U_l} \mathcal{L}^{[t]}(\mathbf{\Theta}) = \left[\frac{\partial}{\partial \mathbf{b}} \mathcal{L}^{[t]}(\mathbf{\Theta})\right] \alpha_l^{[t-1]}(P)^{\top}. \tag{16}$$

Finally, we update the memory units $\mathcal{Q}$ and $\mathcal{E}$; $\hat{f}^{[t]}(P)$ is computed as Eq. (12) and is enqueued to $\mathcal{Q}$, and $\mathcal{E}$ is updated as

$$\alpha_l^{[t]}(P) = \lambda_l \alpha_l^{[t-1]}(P) + \hat{f}^{[t]}(P) \ (l = 1, \ldots, L). \tag{17}$$

## 5 Related Work

As we have shown in the previous section, F-DyBM is derived from G-DyBM by taking the number of neurons to infinity. Besides, Osogami [2016] and Dasgupta and Osogami [2017] discussed a close connection between G-DyBM and VAR, which has been extensively used in econometrics. Therefore, we spare this section to discuss the relationships among a family of DyBMs and VARs, as illustrated in Fig. 2.

First, we discuss the relationships among VAR, G-DyBM, and F-DyBM. The relationship between F-DyBM and G-DyBM has been discussed so far; F-DyBM extends G-DyBM by increasing the number of neurons to infinity. The relationship between G-DyBM and VAR was discussed by Osogami [2016]; G-DyBM extends VAR by adding eligibility traces. In fact, VAR($d$), a VAR model of the $d$-th order can be derived by setting $U_l = \mathbf{O}$ ($l = 1, \ldots, L$) in Eq. (2). The extension from VAR to G-DyBM complies with the design principle of DyBMs to learn a time series online, because the recursive update rule of eligibility traces enables us to equip G-DyBM with an efficient online learning algorithm. This research direction to extend the model while keeping its learning algorithm online is unique in the research domain of VARs, because most of the existing VARs do not focus on the online setting. This direction is also unique in neural network modeling for time-series data. In the same way as DyBM, recurrent neural networks have a memory unit (*i.e.*, a hidden unit) $\mathbf{h}^{[t]}$ that summarizes all the historical patterns by recurrent connection defined as,

$$\mathbf{h}^{[t]} = \sigma\left(W_h \mathbf{f}^{[t]} + U_h \mathbf{h}^{[t-1]} + \mathbf{b}_h\right),$$

in the case of Elman network [Elman, 1990]. However, in order to train the parameters $W_h$, $U_h$, and $\mathbf{b}_h$, backpropagation-through-time is required for gradient computation, and forward computation of $\mathbf{h}^{[t]}$ from the beginning of a time series is also required for prediciton whenever the parameters are updated. Thus, most of the learning algorithms for RNNs are typically not suitable to the online setting as compared with those for DyBMs. Note that the only exception is a family of echo-state networks [Jaeger and Haas, 2004], where the parameters above are fixed, and only the connection from $\mathbf{h}^{[t]}$ to the next pattern $\mathbf{f}^{[t+1]}$ is learned, resulting in a simple linear regression task. Dasgupta and Osogami [2017] augment G-DyBM by an echo-state network, and the resultant learning algorithm is almost as efficient as the original G-DyBM.

Then, we review functional autoregression (FAR) [Bosq, 2000], and discuss its relationship with other models. FAR extends VAR by increasing the dimension of the vector space to infinity in order to model a functional time series. In a similar way to G-DyBM and VAR, we can show that F-DyBM extends FAR by attaching eligibility traces. Besides, F-DyBM is different from FAR in its application as well as the algorithmic aspects. While F-DyBM considers $\mathcal{X}$ as a multi-dimensional space in general, FAR has been mostly applied to a one-dimensional case, *i.e.*, $\mathcal{X} \subseteq \mathbb{R}$, such as a periodic continuous time series. For example, suppose we have a continuous time series $g(x)$ ($x \in \mathbb{R}$) that has period 1, where $x$ represents time. Then, for each time step $t \in \mathbb{N}$, a functional time series $f^{[t]}(x)$ is defined as $f^{[t]}(x) = g(t + x)$ ($x \in [0, 1)$). Such applications of FAR range over climate [Besse

*et al.*, 2000], mortality and fertility rates [Hyndman and Ullah, 2007], the concentration of ozone [Damon and Guillas, 2002], and Eurodollar futures rates [Kargin and Onatski, 2008] to name a few. As for a learning algorithm, Dideriksen *et al.* [2012] reviewed and compared two popular methods for training FAR. The first one is referred to as estimated kernel (EK), which expands and approximates weight functions using finite functional principle components of functional data with the largest eigenvalues [Ramsay and Silverman, 2005]. EK may be preferable to our method in that EK forms basis functions adaptively dedicated to the data set at hand (while ours pre-defines basis functions). However, we avoid from employing EK as a learning algorithm because EK cannot be easily applied to our online setting. The second one is referred to as predictive factors (PF) [Kargin and Onatski, 2008], which employs basis functions that are helpful for prediction rather than explaining the data as EK does. In a similar reason to EK, we do not apply PF to F-DyBM.

# 6 Experiment

We empirically investigate the effectiveness of F-DyBM. F-DyBM is unique in that it has eligibility traces and it models a functional time series rather than a vectorial time series. Thus, we design an experiment to quantitatively evaluate the benefit from these two features.

## 6.1 Methods Evaluated

We evaluated the four methods descried in the related work: VAR, FAR, G-DyBM, and F-DyBM. VAR and FAR are equipped with online learning algorithms in the same way as G-DyBM and F-DyBM. Since VAR and G-DyBM cannot deal with a functional time series as they are, we rely on the following *wrapping method*.

**Wrapping method.** The basic idea is to associate each unit of G-DyBM or VAR with a point in $\mathcal{X}$ and to interpolate a value at any point via a Gaussian process. Let us assume that the $n$-th unit is associated with a point $p_n \in \mathcal{X}$, and let us denote the set of the points by $P = [p_1 \quad \dots \quad p_N]^\top$. Let us assume that a function $f^{[t]}$ is distributed according to the Gaussian process $\mathcal{GP}(0, K_{\sigma^2})$. Then, given observations $f^{[t]}(X^{[t]})$, the signals at $P$ can be estimated as

$$\hat{f}^{[t]}(P) = K(P, X^{[t]}) K_{\sigma^2}(X^{[t]}, X^{[t]})^{-1} f^{[t]}(X^{[t]}).$$

Then, we feed $\hat{f}^{[t]}(P)$ to VAR or G-DyBM for online learning. Prediction can be also implemented using the Gaussian process. Given a prediction by VAR or G-DyBM $\boldsymbol{\mu}^{[t]}$, a prediction at any point $x \in \mathcal{X}$ can be estimated as

$$\hat{f}^{[t]}(x) = K(x, P) K_{\sigma^2}(P, P)^{-1} \boldsymbol{\mu}^{[t]}.$$

We chose these four methods because performance comparison between F-DyBM and G-DyBM and that between FAR and VAR allow us to quantify the contribution of our rigorous treatment of a functional pattern, and performance comparison between F-DyBM and FAR and that between G-DyBM and VAR reveal the contribution of eligibility traces.

## 6.2 Data Sets

We employ five real data sets for performance evaluation. In specific, since our primary motivating example is spatiotemporal prediction, we evaluate the performance on the following two types of spatiotemporal data sets.

**NOAA Global Surface Temperature V4.01**

We use a temperature data set provided by the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA. It contains the global temperature anomalies from January 1880 to present, resulting in the length $1638$.[3] The globe is spatially gridded by $5^\circ \times 5^\circ$, and a temperature anomaly at each location is reported. We selected $392$ locations that observe the temperature without interruption. For each observational location, we define $x \in \mathbb{R}^3$ to be the location on the earth represented by a point on a unit ball in the orthogonal coordinate system.

**AirData**

In addition to the temperature data, we employ four air quality data sets retrieved from AirData[4]. We used the daily summary data of criteria gases (CO, $NO_2$, $O_3$, and $SO_2$) from 1990 to November 2016. Each data set contains daily concentration data of each criteria gas measured at outdoor monitors located in the United States. Since each gas has interrelated but its own generating mechanism, their concentration data have different dynamics. Therefore, it is valuable to evaluate the predictive performance on each of the four data sets.

We preprocessed the data sets as follows. For each data set, we extracted the data summarized by its primary standard of National Ambient Air Quality Standards. Then, we chose records with `Event Type = None` to exclude anomaly records caused by wildfire for example. After that, we dropped multiple records at the same day and the same location by selecting the first record. Finally, we converted each concentration measurement $c \in \mathbb{R}_+$ to $\log(c + 10^{-5})$, and regarded it as $f^{[t]}(x)$, where $t$ corresponds to a day, and $x \in \mathbb{R}^3$ is defined in the same way as the NOAA data set. Each data set has the same length 9,831 and time-variant numbers of observational points $N^{[t]}$. The means and standard deviations of $N^{[t]}$ of CO, $NO_2$, $O_3$, and $SO_2$ are $401.3 \pm 87.6$, $371.0 \pm 37.8$, $870.0 \pm 260.9$, and $540.7 \pm 107.5$, respectively.

## 6.3 Protocol and Configuration

We design the following experimental protocol for performance evaluation. Assume that we have a time series and multiple instances of each model that have different sets of hyperparameters. Our aim is to choose the best instance from the set of candidates and evaluate its generalization ability. To do so, we first divide the time series into training, validation, and test sets in chronological order with ratio $3 : 3 : 4$.

[3] Retrieved from `http://www.esrl.noaa.gov/psd/data/gridded/data.noaaglobaltemp.html` on Aug. 23, 2016. A temperature anomaly refers to a departure from a reference value such as a long-term average

[4] AirData is maintained by the United States Environmental Protection Agency, and we retrieved it from Air Quality System Data Mart (`http://aqsdr1.epa.gov/aqsweb/aqstmp/airdata/download_files.html`) on Dec. 28, 2016.

Table 1: Means and standard deviations of test RMSEs.

| | NOAA | CO | $NO_2$ | $O_3$ | $SO_2$ |
|---|---|---|---|---|---|
| VAR | $1.164 \pm 0.001$ | $2.184 \pm 0.068$ | $1.723 \pm 0.040$ | $1.383 \pm 0.042$ | $3.931 \pm 0.018$ |
| FAR | $1.158 \pm 0.003$ | $1.880 \pm 0.124$ | $1.668 \pm 0.067$ | $0.561 \pm 0.024$ | $3.888 \pm 0.042$ |
| G-DyBM | $1.155 \pm 0.003$ | $1.872 \pm 0.099$ | $1.604 \pm 0.047$ | $0.598 \pm 0.056$ | $3.852 \pm 0.034$ |
| **F-DyBM** | $1.131 \pm 0.009$ | $1.768 \pm 0.073$ | $1.520 \pm 0.014$ | $0.457 \pm 0.049$ | $3.750 \pm 0.058$ |

Then, we train each instance by running its online learning algorithm on the training set. The trained instances are then exposed to the validation set. At each step, each instance predicts one step ahead, and we compute the root-mean-square error (RMSE) using the prediction and the actual observation. The mean of the RMSEs across the entire validation set is computed for each instance, and we choose the instance with the smallest mean RMSE as the best. Finally, we evaluate the performance of the best one on the test set, and report the mean RMSE. Note that in our setting the algorithm runs through each subset of a time series only once, and the model parameters are always updated using a new observation.

For all the models, we used the RBF kernel $K(x, x'; \gamma) = \exp(-\gamma \|x - x'\|^2)$, and prepared the instances by all the combinations of the following hyperparameters: $N = 25$, $d = 3$, $\eta^{[0]} \in \{2^{-n}\}_{n=19}^{23}$, $\sigma^2 \in \{2^{-n}\}_{n=0}^{2}$, $\gamma \in \{2^n\}_{n=0}^{5}$ for all the models and $L = 3$, $\lambda_1 = 0.1$, $\lambda_2 = 0.5$, $\lambda_3 = 0.9$ for G-DyBM and F-DyBM. Since the validation scores of VAR were sometimes exceptionally worse than the others, we additionally tried $\gamma \in \{2^n\}_{n=-3}^{2}$ for VAR. All the models are trained by using SGD, and the learning rate is controlled by rmsprop [Tieleman and Hinton, 2012]. Points $P \in \mathbb{R}^{N \times 3}$ were uniform-randomly sampled from $[-1, 1]^3$ and normalized to the unit vectors. Due to the randomness arising from $P$, we executed the above procedure for ten times, and report the mean and standard deviation for each model.

### 6.4 Results

Experimental results are summarized in Table 1. We first notice consistent performance improvements resulting from direct modeling of a functional time series when comparing F-DyBM with G-DyBM and FAR with VAR. In specific, the RMSE of F-DyBM is 76–98% (92% on average) of that of G-DyBM, and the RMSE of FAR is 41–99% (84% on average) of that of VAR. Also, performance comparison between F-DyBM and FAR, and G-DyBM and VAR demonstrates consistent benefit from eligibility traces; the RMSE of F-DyBM is 81–98% (92% on average) of that of FAR, and the RMSE of G-DyBM is 43–99% (84% on average) of that of VAR. In summary, this numerical experiment demonstrates consistent performance improvements by turning on each feature of F-DyBM, which verifies its effectiveness.

## 7 Concluding Remark and Future Work

This paper has presented a functional dynamic Boltzmann machine (F-DyBM) as an extension of Gaussian DyBM with the aim of learning a functional time series online while still considering all the past patterns. Whereas it is relatively straightforward to extend the model, devising an online learning algorithm yields three technical challenges regarding how to deal with and operate functions and how to learn from finite observations of a functional pattern. We have addressed these difficulties with two ideas: to model weight functions by a kernel function and finite parameters and to replace the functional patterns with their MAP estimators progressively. These two ideas successfully lead an online learning algorithm for F-DyBM. We designed an experiment to demonstrate the effectiveness of two essential characteristics of F-DyBM, *i.e.*, eligibility traces and careful modeling of a functional time series. As a result, we have observed that eligibility traces decrease the error by 12.0% on average and the functional modeling decreases the error by 11.7%.

One future direction will be to develop further more efficient learning algorithms. Since our current implementation requires to solve a linear system of size $N^{[t]}$ at each time step, time complexity of each step amounts to $\mathrm{O}\left((N^{[t]})^3\right)$ (while the wrapping method requires $\mathrm{O}\left((N^{[t]})^3 + N^3\right)$). In the Gaussian process community, inference algorithms with much lower time complexity have been extensively studied [Quiñonero-Candela and Rasmussen, 2005; Rasmussen and Williams, 2006; Wilson *et al.*, 2015], and we believe leveraging these innovations will lead to a scalable algorithm.

Another research direction is to use F-DyBM for language modeling, predicting the next word given the former words in a document. F-DyBM can model this task by regarding a position in a document as a time step, a word vector obtained by `word2vec` [Mikolov *et al.*, 2013] as $x$, and $f^{[t]}(x)$ as a one-hot representation of the word appearing at position $t$. Since F-DyBM can leverage all the predecessor words and their similarities for prediction, we believe F-DyBM will achieve more than moderate performance with much efficiency.

Finally, it is fruitful to examine the theoretical properties of F-DyBM, especially the estimation error induced by progressively replacing $f^{[t-\delta]}(Q)$ with its MAP estimator in Eq. (10). Recently, Anava *et al.* [2015] have presented an online prediction algorithm for an AR model that can handle missing data, and they also have derived its regret bound. However, there are still several obstacles that hinder us from applying their theory to our algorithm. For example, since their theory does not assume the underlying model generating a time series, they consider the regret between a particular inefficient predictor and its efficient but approximated predictor, not the regret between their algorithm and the best possible AR model in the fully-observable case. Therefore, we believe that the first step towards investigating the theoretical properties of F-DyBM is to examine the regret bound against the fully-observable case.

## References


[Anava *et al.*, 2015] Oren Anava, Elad Hazan, and Assaf Zeevi. Online time series prediction with missing data. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2191–2199, 2015.

[Besse *et al.*, 2000] Philippe C. Besse, Hervé Cardot, and David B. Stephenson. Autoregressive forecasting of some functional climatic variations. *Scandinavian Journal of Statistics*, 27:673–687, 2000.

[Bosq, 2000] Denis Bosq. *Linear Processes in Function Spaces: Theory and Applications*, chapter 5. Springer New York, 2000.

[Damon and Guillas, 2002] Julien Damon and Serge Guillas. The inclusion of exogenous variables in functional autoregressive ozone forecasting. *Environmetrics*, 13:759–774, 2002.

[Dasgupta and Osogami, 2017] Sakyasingha Dasgupta and Takayuki Osogami. Nonlinear dynamic Boltzmann machines for time-series prediction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[Didericksen *et al.*, 2012] Devin Didericksen, Piotr Kokoszka, and Xi Zhang. Empirical properties of forecasts with the functional autoregressive model. *Computational Statistics*, 27(2):285–298, 2012.

[Elman, 1990] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Hyndman and Ullah, 2007] Rob J. Hyndman and Md Shahid Ullah. Robust forecasting of mortality and fertility rates: a functional data approach. *Computational Statistics & Data Analysis*, 51(10):4942–4956, 2007.

[Jaeger and Haas, 2004] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[Kargin and Onatski, 2008] Vladislav Kargin and Alexei Onatski. Curve forecasting by functional autoregression. *Journal of Multivariate Analysis*, 99(10):2508–2526, 2008.

[Lütkepohl, 2005] Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Part I. Springer Berlin Heidelberg, 2005.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.

[Osogami and Otsuka, 2015] Takayuki Osogami and Makoto Otsuka. Seven neurons memorizing sequences of alphabetical images via spike-timing dependent plasticity. *Scientific Reports*, 5:14149 EP –, 09 2015.

[Osogami, 2016] Takayuki Osogami. Learning binary or real-valued time-series via spike-timing dependent plasticity. In *The First NIPS Workshop Computing with Spikes*, 2016.

[Quiñonero-Candela and Rasmussen, 2005] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[Ramsay and Silverman, 2005] James Ramsay and Bernard W. Silverman. *Functional Data Analysis*, chapter 8. Springer-Verlag New York, 2005.

[Rasmussen and Williams, 2006] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*, chapter 8. MIT Press, 2006.

[Rumelhart *et al.*, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning internal representations by error propagation*, chapter 8, pages 318–362. MIT Press, 1986.

[Shumway and Stoffer, 1982] Robert H. Shumway and David S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

[Sundermeyer *et al.*, 2012] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *Interspeech*, pages 194–197, 2012.

[Tieleman and Hinton, 2012] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, coursera: Neural networks for machine learning, 2012.

[Wilson *et al.*, 2015] Andrew Gordon Wilson, Christopher Dann, and Hannes Nickisch. Thoughts on massively scalable Gaussian processes. Technical report, eprint arXiv:1511.01870, 2015.