Bidirectional Learning for Time-series Models with Hidden Units

Takayuki Osogami [1] **Hiroshi Kajino** [1] **Taro Sekiyama** [1]

Abstract

Hidden units can play essential roles in modeling time-series having long-term dependency or non-linearity but make it difficult to learn associated parameters. Here we propose a way to learn such a time-series model by training a backward model for the time-reversed time-series, where the backward model has a common set of parameters as the original (forward) model. Our key observation is that only a subset of the parameters is hard to learn, and that subset is complementary between the forward model and the backward model. By training both of the two models, we can effectively learn the values of the parameters that are hard to learn if only either of the two models is trained. We apply bidirectional learning to a dynamic Boltzmann machine extended with hidden units. Numerical experiments with synthetic and real datasets clearly demonstrate advantages of bidirectional learning.

1. Introduction

Learning from time-series data is of paramount importance for prediction, anomaly detection, classification, and other critical tasks that appear in business and society. Various models of time-series have been studied in the literature to better learn from time-series data. These include vector auto-regressive (VAR) models (Lütkepohl, 2005), hidden Markov models (HMM) (Baum & Petrie, 1966), and recurrent neural networks (RNN) (Rumelhart et al., 1986), including long short term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and echo state networks (ESN) (Jaeger & Haas, 2004). With these models of time-series, one seeks to learn the relation between past values and future values.

In some of these models of time-series, hidden units (or latent variables) play essential roles in taking into account long term dependency or non-linearity in time-series. Hidden units, however, make it difficult to learn the parameters of those models. For example, the Baum-Welch algorithm (Baum & Petrie, 1966) learns the parameters of an HMM by iteration of expectation and maximization (*i.e.*, an EM algorithm). An RNN, including LSTM, is trained via back propagation through time (Rumelhart et al., 1986). These algorithms are time-consuming and do not necessarily find optimal values of the parameters.

This difficulty in learning a model with hidden units partly stems from the fact that the values of hidden units can only be reliably estimated after observing future values of target time-series. It then requires iteration or back propagation to learn the relation between the hidden values and preceding values. An ESN, on the other hand, gives up learning the hard-to-learn parameters between hidden values and preceding (visible or hidden) values and set those parameters randomly. The ESN only learns the relation between visible values and preceding (hidden) values (Jaeger & Haas, 2004).

We study a model of time-series whose parameters can be represented as a matrix $\mathbf{M}$ or a set of such matrices. An element $M_{i,j}$ of the matrix may, for example, represent the weight between a past value of a unit i and a future value of a unit j. In a VAR model, each matrix corresponds to the coefficients for a particular lag. In an RNN, a matrix corresponds to the weight between hidden units. We consider a situation where it is hard to estimate an appropriate value of $M_{i,j}$ for some of the units $j \in \mathcal{H}$ (in particular, $\mathcal{H}$ may denote the set of hidden units).

We propose a method of training a time-series model with hidden units in a bidirectional manner, one from the past and the other from the future. From a time-series model with parameter $\mathbf{M}$, we construct a backward model, whose parameters are represented by the transposed matrix $\mathbf{M}^\top$ or a set of such transposed matrices. The (i, j)-th element of $\mathbf{M}^\top$ is $M_{j,i}$, which represents the weight between a preceding value of a unit j and a succeeding value of a unit i. Our key idea is to let the preceding value represent a future value and the succeeding value represent a past value in the backward model. Then an intuitive meaning of $M_{j,i}$ in the backward model matches that in the original (forward) model. We use a common matrix $\mathbf{M}$ both in the forward model and in the backward model. Namely, the parameters are shared between the two models.

[1]IBM Research - Tokyo, Tokyo, Japan. Correspondence to: Takayuki Osogami <osogami@jp.ibm.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

The two models have an identical structure and trained in an identical manner with a stochastic gradient method (Bottou, 2009) except that we train the forward model using time-series in a standard (forward) manner and the backward model using the time-series from the future to the past. We alternately train the forward model to learn $\mathbf{M}$ and the backward model to learn $\mathbf{M}^\top$. An advantage of our bidirectional training is that the elements of $\mathbf{M}$ that are hard to estimate differ between when we train the forward model and when we train the backward model. For the forward model, it is hard to estimate $M_{i,j}$ for $j \in \mathcal{H}$ (*i.e.*, $\mathbf{M}_{:,\mathcal{H}}$). For the backward model, it is hard to estimate $M_{j,i}$ for $j \in \mathcal{H}$ (*i.e.*, $\mathbf{M}_{\mathcal{H},:} = (\mathbf{M}^\top)_{:,\mathcal{H}}$). Although $\mathbf{M}_{\mathcal{H},\mathcal{H}}$ is hard to estimate for both models, the other elements of $\mathbf{M}$ can be reliably estimated in either of the two models. This idea of bidirectional training for learning time-series models with hidden units constitutes our first contribution.

Here, we extend a Dynamic Boltzmann Machine (DyBM) to incorporate hidden units and apply bidirectional training to learn its parameters. The DyBM has been proposed by Osogami & Otsuka (2015a;b) and subsequently studied by Dasgupta et al. (2016); Dasgupta & Osogami (2017); Kajino (2017). The prior work on the DyBM does not consider hidden units but instead uses various features of past values to capture the long term dependency in time-series. For example, Dasgupta & Osogami (2017) use an ESN to create nonlinear features of past values. In our context, these features are fed into visible units of a DyBM. In particular, what features of past values are used is determined randomly without learning. Our analysis of the DyBM with hidden units illuminates the difficulty of learning some of its parameters. With bidirectional learning, we seek to learn the weight from the past visible values to the future hidden values, which corresponds to learning what features of past values are effective for prediction. The DyBM with hidden units and its analysis constitute our second contribution.

We validate the effectiveness of bidirectional training and the hidden units in DyBMs through numerical experiments using synthetic and real time-series. We will show that the DyBM with hidden units can be trained more effectively with bidirectional training and reduces the predictive error by up to 90 %.

1.1. Related Work

Our bidirectional training is related to but different from bidirectional recurrent neural networks (BRNN) (Schuster & Paliwal, 1997; Baldi et al., 1999), including bidirectional LSTM (Graves & Schmidhuber, 2005; Chen & Chaudhari, 2005). Similar to our bidirectional training, a BRNN trains both a forward model and a backward model. These two models, however, do not share parameters, contrary to our bidirectional training. In fact, motivation and purpose of the BRNN are quite different from ours. The BRNN uses both the sequence from the past and the sequence from the future to better estimate missing values. The BRNN thus needs both of the two sequences for learning and for prediction. On the other hand, our bidirectional training uses both a forward sequence and a backward sequence at the time of learning, but the trained model uses only the sequence from the past to predict future unseen values. Namely, our bidirectional training is used to learn a model for predicting future values from past values, while the BRNN is used for estimating missing values from past and future values.

Our bidirectional training is also related to but different from Forward Backward Lasso Granger (FBLG) by Cheng et al. (2014). In FBLG, forward and backward VAR models are estimated with Lasso, and an averaged model is used to infer the Granger causality. The VAR models in FBLG do not have hidden units, while our bidirectional training is motivated by the need for training time-series models with hidden units. Winkler et al. (2016) also study a backward VAR in the context of the Granger causality but do not consider hidden units.

Another related work is structure learning of a VAR model (Bahadori et al., 2013) or a simpler linear dynamical system (Jalali & Sanghavi, 2012) that takes into account the existence of unobserved (or latent) variables. However, their goal is to reliably estimate the structure of the relation between observable variables by taking into account the unobserved variables. This is in contrast to the purpose of our bidirectional training, which aims at learning the relation between visible units and hidden units.

2. DyBM with Hidden Units

We study a particularly structured Boltzmann machine for time-series (see Figure 1). Corresponding to a segment of a time-series of length $T+1$, the Boltzmann machine in Figure 1 has $T+1$ layers in the horizontal (temporal) direction. Each layer corresponds to a time $t-\delta$ for $0 \leq \delta \leq T$. Each layer has two parts, hidden and visible. The visible part $\mathbf{x}^{[t-\delta]}$ at the δ-th layer represents the values of the time-series at time $t-\delta$. The hidden part $\mathbf{h}^{[t-\delta]}$ represents the values of hidden units at time $t-\delta$. Here, units within each layer do not have connections to each other. We let $\mathbf{x}^{[<t]} \equiv (\mathbf{x}^{[s]})_{t-T \leq s < t}$ and define $\mathbf{h}^{[<t]}$ analogously.

The Boltzmann machine in Figure 1 has bias parameter $\mathbf{b}$ and weight parameter ($\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$, $\mathbf{Z}$). Let $\theta \equiv (\mathbf{V}, \mathbf{W}, \mathbf{b})$ be the parameters connected to visible units $\mathbf{x}^{[t]}$ (from the units in the past, $\mathbf{x}^{[s]}$ or $\mathbf{h}^{[s]}$ for $s < t$) and $\phi \equiv (\mathbf{U}, \mathbf{Z})$. The energy of this Boltzmann machine is given as follows:

$$E_{\theta,\phi}(\mathbf{x}^{[t]}, \mathbf{h}^{[t]} | \mathbf{x}^{[<t]}, \mathbf{h}^{[<t]}) = E_\theta(\mathbf{x}^{[t]} | \mathbf{x}^{[<t]}, \mathbf{h}^{[<t]}) + E_\phi(\mathbf{h}^{[t]} | \mathbf{x}^{[<t]}, \mathbf{h}^{[<t]}), \quad (1)$$

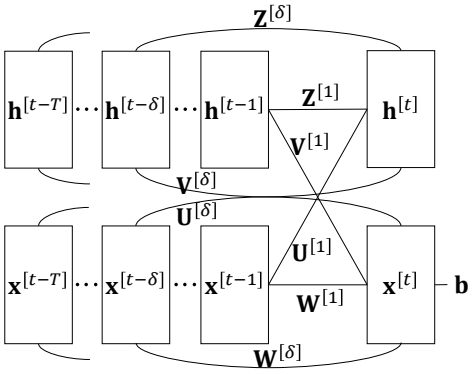


Figure 1. A (forward) DyBM with hidden units.

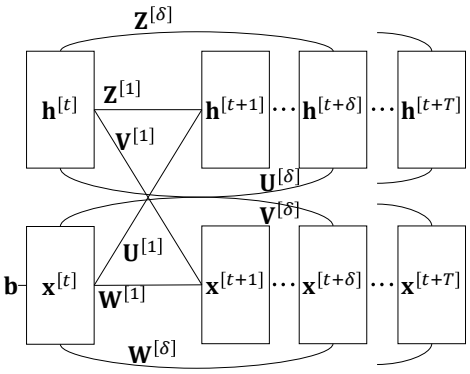


Figure 2. A backward DyBM.

where we define

$$E_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]}) \tag{2}$$
$$= -\mathbf{b}^\top \mathbf{x}^{[t]} - \sum_{\delta=1}^{T} (\mathbf{x}^{[t-\delta]})^\top \mathbf{W}^{[\delta]}\, \mathbf{x}^{[t]} - \sum_{\delta=1}^{T} (\mathbf{h}^{[t-\delta]})^\top \mathbf{V}^{[\delta]}\, \mathbf{x}^{[t]}$$

and define $E_\phi(\mathbf{h}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})$ from (2) by letting $\mathbf{W} \leftarrow \mathbf{U}$, $\mathbf{V} \leftarrow \mathbf{Z}$, $\mathbf{b} \leftarrow \mathbf{0}$, and $\mathbf{x}^{[t]} \leftarrow \mathbf{h}^{[h]}$.

Similar to the DyBM in Osogami & Otsuka (2015a), we study the case where $\mathbf{W} \equiv (\mathbf{W}^{[\delta]})_{1\le\delta\le T}$ and other matrices have the following parametric forms for $\delta \ge d$:

$$\mathbf{W}^{[\delta]} = \lambda^{\delta-d}\, \mathbf{W}^{[d]}, \quad \mathbf{V}^{[\delta]} = \lambda^{\delta-d}\, \mathbf{V}^{[d]}, \tag{3}$$
$$\mathbf{Z}^{[\delta]} = \lambda^{\delta-d}\, \mathbf{Z}^{[d]}, \quad \mathbf{U}^{[\delta]} = \lambda^{\delta-d}\, \mathbf{U}^{[d]}, \tag{4}$$

where λ is a decay rate satisfying $0 \le \lambda < 1$. Then, in the limit of $T \to \infty$, the energy in (2) can be represented as follows (and $E_\phi(\mathbf{h}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})$ has an analogous limit shown in (37) of the supplementary material):

$$E_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})$$
$$= -\mathbf{b}^\top \mathbf{x}^{[t]} - \sum_{\delta=1}^{d-1} (\mathbf{x}^{[t-\delta]})^\top \mathbf{W}^{[\delta]}\, \mathbf{x}^{[t]} - \sum_{\delta=1}^{d-1} (\mathbf{h}^{[t-\delta]})^\top \mathbf{V}^{[\delta]}\, \mathbf{x}^{[t]}$$
$$- (\boldsymbol{\alpha}^{[t-1]})^\top \mathbf{W}^{[d]}\, \mathbf{x}^{[t]} - (\boldsymbol{\beta}^{[t-1]})^\top \mathbf{V}^{[d]}\, \mathbf{x}^{[t]}, \tag{5}$$

where $\boldsymbol{\alpha}^{[t-1]}$ is referred to as an eligibility trace in Osogami & Otsuka (2015a) and defined as follows (here, we define an eligibility trace $\boldsymbol{\beta}^{[t-1]}$ for the hidden part analogously):

$$\boldsymbol{\alpha}^{[t-1]} \equiv \sum_{\delta=d}^{\infty} \lambda^{\delta-d}\, \mathbf{x}^{[t-\delta]}, \quad \boldsymbol{\beta}^{[t-1]} \equiv \sum_{\delta=d}^{\infty} \lambda^{\delta-d}\, \mathbf{h}^{[t-\delta]}. \tag{6}$$

The energy in (5) gives the conditional probability distribution over $\mathbf{x}^{[t]}$ given $\mathbf{x}^{[<t]}$ and $\mathbf{h}^{[<t]}$. For binary-valued time-series, we have

$$p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]}) = \frac{1}{Z} \exp(-E_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})) \tag{7}$$

for any binary vector $\mathbf{x}^{[t]}$, where Z is the normalization factor for the probabilities to sum up to one. Due to the structure in Figure 1, the values in $\mathbf{x}^{[t]} = (x_i^{[t]})_{i=1,2,\dots}$ are conditionally independent of each other given $\mathbf{x}^{[<t]}$ and $\mathbf{h}^{[<t]}$, so that we can represent

$$p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]}) = \prod_{i=1,2,\dots} p_{\theta,i}(x_i^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]}), \tag{8}$$

where the conditional probability $p_i(x_i^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})$ is defined with the energy associated with unit i (see Osogami & Otsuka (2015a)). For real-valued time-series, one can define the conditional density $p_i(x_i^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})$ with a Gaussian distribution whose mean is given from the energy associated with unit i (Dasgupta & Osogami, 2017; Osogami, 2016). Conditional distributions can be defined analogously for $\mathbf{h}^{[t]}$ (see (40)–(42) and (51) in the supplementary material).

3. Training a DyBM with Hidden Units

Here, we derive a learning rule for θ. We will also see that ϕ cannot be trained in an analogous manner.

Our DyBM with binary hidden units gives the probability of a time-series, $\mathbf{x} \equiv (\mathbf{x}^{[t]})_{t=\ell,\dots,u}$, by

$$p_{\theta,\phi}(\mathbf{x}) = \sum_{\tilde{\mathbf{h}}} p_\phi(\tilde{\mathbf{h}}|\mathbf{x}) \prod_{t=\ell}^{u} p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \tilde{\mathbf{h}}^{[<t]}) \tag{9}$$

where $\sum_{\tilde{\mathbf{h}}}$ denotes the summation over all of the possible values of hidden units from time $t = \ell$ to $t = u$, and

$$p_\phi(\tilde{\mathbf{h}}|\mathbf{x}) \equiv \prod_{s=\ell}^{u} p_\phi(\tilde{\mathbf{h}}^{[s]}|\mathbf{x}^{[<s]}, \tilde{\mathbf{h}}^{[<s]}), \tag{10}$$

where $p_\phi(\tilde{\mathbf{h}}^{[s]}|\mathbf{x}^{[<s]}, \tilde{\mathbf{h}}^{[<s]})$ is defined analogously to (7)–(8) and provided in (36) of the supplementary material, and we arbitrarily define $\mathbf{x}^{[s]} = \mathbf{0}$ and $\tilde{\mathbf{h}}^{[s]} = \mathbf{0}$ for $s < \ell$.

We seek to maximize the log likelihood of a given $\mathbf{x}$ by maximizing a lower bound given by Jensen's inequality:

$$\log p_{\theta,\phi}(\mathbf{x})$$
$$= \log\Big(\sum_{\tilde{\mathbf{h}}} p_\phi(\tilde{\mathbf{h}}|\mathbf{x}) \prod_{t=\ell}^{u} p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \tilde{\mathbf{h}}^{[<t]})\Big) \tag{11}$$
$$\geq \sum_{\tilde{\mathbf{h}}} p_\phi(\tilde{\mathbf{h}}|\mathbf{x}) \log\Big(\prod_{t=\ell}^{u} p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \tilde{\mathbf{h}}^{[<t]})\Big) \tag{12}$$
$$= \sum_{\tilde{\mathbf{h}}} p_\phi(\tilde{\mathbf{h}}|\mathbf{x}) \sum_{t=\ell}^{u} \log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \tilde{\mathbf{h}}^{[<t]}) \tag{13}$$
$$= \sum_{t=\ell}^{u} \sum_{\tilde{\mathbf{h}}^{[<t]}} p_\phi(\tilde{\mathbf{h}}^{[<t]}|\mathbf{x}^{[<t-1]}) \log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \tilde{\mathbf{h}}^{[<t]})$$
$$\equiv L_{\theta,\phi}(\mathbf{x}), \tag{14}$$

where the summation with respect to $\tilde{\mathbf{h}}^{[<t]}$ is over all of the possible values of $\tilde{\mathbf{h}}^{[s]}$ for $s \leq t-1$, and

$$p_\phi(\tilde{\mathbf{h}}^{[<t]}|\mathbf{x}^{[<t-1]}) \equiv \prod_{s=\ell}^{t-1} p_\phi(\tilde{\mathbf{h}}^{[s]}|\mathbf{x}^{[<s]}, \tilde{\mathbf{h}}^{[<s]}). \tag{15}$$

The gradient of the lower bound with respect to θ is:

$$\nabla_\theta L_{\theta,\phi}(\mathbf{x}) \tag{16}$$
$$= \sum_{t=\ell}^{u} \sum_{\tilde{\mathbf{h}}^{[<t]}} p_\phi(\tilde{\mathbf{h}}^{[<t]}|\mathbf{x}^{[<t-1]}) \nabla_\theta \log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \tilde{\mathbf{h}}^{[<t]}).$$

The right-hand side of (16) is a summation of expected gradients, which suggests a method of stochastic gradient. Namely, at each step t, we sample $\mathbf{h}^{[t-1]}$ according to $p_\phi(\mathbf{h}^{[t-1]}|\mathbf{x}^{[<t-1]}, \mathbf{h}^{[<t-1]})$ and update θ on the basis of

$$\nabla_\theta \log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]}). \tag{17}$$

This learning rule is equivalent to the one for the model where all of the units are visible, except that the values for the hidden units are given by sampled values.

Therefore, the learning rule for θ follows directly from Osogami & Otsuka (2015a):

$$\mathbf{b} \leftarrow \mathbf{b} + \eta\,(\mathbf{x}^{[t]} - \langle \mathbf{X}^{[t]} \rangle_\theta) \tag{18}$$
$$\mathbf{W}^{[d]} \leftarrow \mathbf{W}^{[d]} + \eta\,\boldsymbol{\alpha}^{[t-1]}\,(\mathbf{x}^{[t]} - \langle \mathbf{X}^{[t]} \rangle_\theta)^\top \tag{19}$$
$$\mathbf{V}^{[d]} \leftarrow \mathbf{V}^{[d]} + \eta\,\boldsymbol{\beta}^{[t-1]}\,(\mathbf{x}^{[t]} - \langle \mathbf{X}^{[t]} \rangle_\theta)^\top \tag{20}$$
$$\mathbf{W}^{[\delta]} \leftarrow \mathbf{W}^{[\delta]} + \eta\,\mathbf{x}^{[t-\delta]}\,(\mathbf{x}^{[t]} - \langle \mathbf{X}^{[t]} \rangle_\theta)^\top \tag{21}$$
$$\mathbf{V}^{[\delta]} \leftarrow \mathbf{V}^{[\delta]} + \eta\,\mathbf{h}^{[t-\delta]}\,(\mathbf{x}^{[t]} - \langle \mathbf{X}^{[t]} \rangle_\theta)^\top \tag{22}$$

for $1 \leq \delta < d$, where $\langle \mathbf{X}^{[t]} \rangle_\theta$ denotes the expected values of $\mathbf{x}^{[t]}$ with respect to p_θ in (7).

Now we take the gradient of $L_{\theta,\phi}(\mathbf{x})$ with respect to ϕ:

$$\nabla_\phi L_{\theta,\phi}(\mathbf{x}) \tag{23}$$
$$= \sum_{t=\ell}^{u} \sum_{\tilde{\mathbf{h}}^{[<t]}} \nabla_\phi p_\phi(\tilde{\mathbf{h}}^{[<t]}|\mathbf{x}^{[<t-1]}) \log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \tilde{\mathbf{h}}^{[<t]}),$$

where

$$\nabla_\phi p_\phi(\tilde{\mathbf{h}}^{[<t]}|\mathbf{x}^{[<t-1]})$$
$$= \nabla_\phi \prod_{s=\ell}^{t-1} p_\phi(\tilde{\mathbf{h}}^{[s]}|\mathbf{x}^{[<s]}, \tilde{\mathbf{h}}^{[<s]}) \tag{24}$$
$$= \sum_{s=\ell}^{t-1} \nabla_\phi \log p_\phi(\tilde{\mathbf{h}}^{[s]}|\mathbf{x}^{[<s]}, \tilde{\mathbf{h}}^{[<s]}) \prod_{s'=\ell}^{t-1} p_\phi(\tilde{\mathbf{h}}^{[s']}|\mathbf{x}^{[<s']}, \tilde{\mathbf{h}}^{[<s']})$$
$$= p_\phi(\tilde{\mathbf{h}}^{[<t]}|\mathbf{x}^{[<t-1]}) \sum_{s=\ell}^{t-1} \nabla_\phi \log p_\phi(\tilde{\mathbf{h}}^{[s]}|\mathbf{x}^{[<s]}, \tilde{\mathbf{h}}^{[<s]}). \tag{25}$$

Plugging (25) into the right-hand side of (23), we obtain

$$\nabla_\phi L_{\theta,\phi}(\mathbf{x})$$
$$= \sum_{t=\ell}^{u} \sum_{\tilde{\mathbf{h}}^{[<t]}} p_\phi(\tilde{\mathbf{h}}^{[<t]}|\mathbf{x}^{[<t-1]}) \log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \tilde{\mathbf{h}}^{[<t]}) \sum_{s=\ell}^{t-1} \nabla_\phi \log p_\phi(\tilde{\mathbf{h}}^{[s]}|\mathbf{x}^{[<s]}, \tilde{\mathbf{h}}^{[<s]}). \tag{26}$$

Similar to (16), the expression of (26) suggests a method of stochastic gradient: at each time t, we sample $\mathbf{h}^{[t-1]}$ according to $p_\phi(\mathbf{h}^{[t-1]}|\mathbf{x}^{[<t-1]}, \mathbf{h}^{[<t-1]})$ and update ϕ on the basis of the following stochastic gradient:

$$\log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})\, G_{t-1}, \tag{27}$$

where

$$G_{t-1} \equiv \sum_{s=\ell}^{t-1} \nabla_\phi \log p_\phi(\mathbf{h}^{[s]}|\mathbf{x}^{[<s]}, \mathbf{h}^{[<s]}). \tag{28}$$

Computation of (26) involves mainly two interrelated inefficiencies. First, although (26) can be approximately computed using sampled hidden values $\tilde{\mathbf{h}}^{[<t]}$ in the same way as (16), the samples cannot be reused after updating ϕ because it was sampled from the distribution with the previous parameter. Second, since each summand of G_{t-1} is dependent on ϕ, G_{t-1} also has to be recomputed after each update. Thus, the computational complexity of (27) grows linearly with respect to the length of the time-series (*i.e.*, $t - \ell$), in contrast to (17), whose complexity is independent of that length. One could approximately compute (28) recursively:

$$G_t \leftarrow \gamma\, G_{t-1} + (1-\gamma)\nabla_\phi \log p_\phi(\mathbf{h}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]}), \tag{29}$$

where $\gamma \in [0, 1)$ is a discount factor. The recursive update rule with $\gamma < 1$ puts exponentially small weight γ^{t-s} on $\nabla_\phi \log p_\phi(\mathbf{h}^{[s]}|\mathbf{x}^{[<s]}, \mathbf{h}^{[<s]})$ computed with an old value of ϕ (*i.e.*, $s \ll t$). This recursively computed G_t is related to the momentum in gradient descent (Qian, 1999). See the supplementary material for specific learning rules suggested by (27)–(29).

Observe in (26) that $\nabla_\phi L_{\theta,\phi}(\mathbf{x})$ consists of the products of $\log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})$ and $\nabla_\phi \log p_\phi(\mathbf{h}^{[s]}|\mathbf{x}^{[<s]}, \mathbf{h}^{[<s]})$ for $s < t$. Without the dependency on $\log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})$, the parameter ϕ is updated in a way that $\mathbf{h}^{[s]}$ is more likely to be generated (*i.e.*, the learning rule would be equivalent to that for visible units). Such an update rule is undesirable, because $\mathbf{h}^{[s]}$ has been sampled and is not necessarily what we want to sample again. The dependency on $\log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})$ suggests that ϕ is updated by a large amount if the sampled $\mathbf{h}^{[s]}$ happens to make the future values, $\mathbf{x}^{[t]}$ for $t > s$, likely. Intuitively, weighting $\nabla_\phi \log p_\phi(\mathbf{h}^{[s]}|\mathbf{x}^{[<s]}, \mathbf{h}^{[<s]})$ by $\log p_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[<t]}, \mathbf{h}^{[<t]})$ for $t > s$ is inevitable, because whether the particular values of hidden units are good for the purpose of predicting future values will only be known after seeing future values.

4. Learning with Reversed Time-series

Because the stochastic gradient for ϕ requires approximations that are not needed for θ, we might not be able to learn appropriate values of ϕ as effectively as θ. This motivates us to consider a backward DyBM in Figure 2, which has a common set of parameters, (θ, ϕ), as the forward DyBM in Figure 1 but defines the conditional distribution for time-series *from the future*. Specifically, the energy of the backward DyBM is represented analogously to (1) with the superscript $[< t]$ replaced by $[> t]$, where we define

$$E_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[>t]}, \mathbf{h}^{[>t]}) = -\mathbf{b}^\top \mathbf{x}^{[t]} - \sum_{\delta=1}^{T} (\mathbf{x}^{[t+\delta]})^\top (\mathbf{W}^{[\delta]})^\top \mathbf{x}^{[t]} - \sum_{\delta=1}^{T} (\mathbf{x}^{[t+\delta]})^\top (\mathbf{U}^{[\delta]})^\top \mathbf{h}^{[t]} \quad (30)$$

and $E_\phi(\mathbf{h}^{[t]}|\mathbf{x}^{[>t]}, \mathbf{h}^{[>t]})$ is defined from (30) by letting $\mathbf{W} \leftarrow \mathbf{V}$, $\mathbf{U} \leftarrow \mathbf{Z}$, $\mathbf{b} \leftarrow \mathbf{0}$, and $\mathbf{x}^{[t]} \leftarrow \mathbf{h}^{[t]}$. Similar to the forward DyBM, we assume that the weight has the parametric form of (4) and let $T \to \infty$.

Namely, the backward DyBM is obtained from the forward DyBM by the following changes: $\mathbf{W} \leftarrow \mathbf{W}^\top$, $\mathbf{Z} \leftarrow \mathbf{Z}^\top$, $\mathbf{U} \leftarrow \mathbf{V}^\top$, and $\mathbf{V} \leftarrow \mathbf{U}^\top$. The other difference between (2) and (30) is the sign of δ, but this is because the backward DyBM deals with time-series from the future.

Because the backward DyBM has the structure that is equivalent to that of the forward DyBM, it can be trained in the same manner as the forward DyBM but using time-series from the future. Specifically, $\theta' \equiv (\mathbf{U}^\top, \mathbf{W}^\top, \mathbf{b})$ in the backward DyBM is optimized analogously to θ in the forward DyBM. Likewise, $\phi' \equiv (\mathbf{V}^\top, \mathbf{Z}^\top)$ is optimized analogously to ϕ. Recall that ϕ and ϕ' are relatively hard to optimize, and θ and θ' are relatively easy to optimize.

Our key observation is that the parameter $\mathbf{U}$, which is in ϕ and is relatively hard to optimize in the forward DyBM, is in θ' and is relatively easy to optimize in the backward DyBM. By training both the forward DyBM and the backward DyBM, we expect to effectively find appropriate values of θ and θ'.

Consider a stochastic process $X(t)$ whose distribution is given by a forward DyBM. We remark that the distribution of the stochastic process $X(-t)$ that is defined by reversing $X(t)$ is generally different from what the corresponding backward DyBM gives unless the DyBMs have no hidden units. The exact distribution of $X(-t)$ needs to be given by marginalizing out the past values (*i.e.*, succeeding values for the backward process). Despite this discrepancy, we expect that bidirectional training is effective because of intuitive correspondence between the forward DyBM and the backward DyBM. In particular, $W_{i,j}^{[\delta]}$ in both DyBMs represent the strength of the correlation between the past value of unit i and the future value of unit j, where the time is separated by δ. A recommendation is, however, to perform backward training more moderately than forward training. We will show an example of a specific procedure in the next section.

5. Numerical Experiments

We now demonstrate the effectiveness of bidirectional training through numerical experiments in two settings. The purpose of the first setting is to study whether bidirectional learning of the DyBM with hidden units can indeed learn to predict what cannot be done without bidirectional learning or hidden units. We use a synthetic dataset that is designed specifically for this purpose. In the second setting, we study the effectiveness of bidirectional learning and hidden units on real datasets. We use the two datasets that have been used in Dasgupta & Osogami (2017) as well as a 391 dimensional time-series, which is substantially larger than the eight or lower dimensional time-series that are used in the other experiments. The experiments are carried out with a Python implementation on workstations having 48-64 GB memory and 2.6-4.0 GHz CPU.

5.1. Specific Learning Algorithms to Evaluate

For each dataset, we train a DyBM with or without hidden units. Because all of the datasets are real valued, visible units are Gaussian and give predictions by (5) with $\mathbf{x}^{[t]}$ omit-

Algorithm 1 Specific steps of bidirectional learning evaluated in experiments

T: The total number of iterations
T_0: The number of iterations of bidirectional learning
F: The relative frequency of forward learning
for $t = 1$ to T **do**
  if $t < T_0$ and t mod $F + 1 = 0$ **then**
    Backward learning to update $(\mathbf{U}, \mathbf{W}, \mathbf{b})$
  else
    Forward learning to update $(\mathbf{V}, \mathbf{W}, \mathbf{b})$
  end if
end for

ted (see (52) in the supplementary material). To reduce the variability in the experiments, we use the expected value for the output of a hidden unit instead of sampling a binary value. By the law of large numbers, the use of expected value corresponds to having an infinitely many binary hidden units that are conditionally independent and identically distributed (i.i.d.) given the internal state of the DyBM. See also (Sutskever et al., 2008; Sutskever & Hinton, 2007) for the related use of the expected values for hidden units.

A DyBM with hidden unit is trained bidirectionally or only with forward learning. A DyBM without hidden units is trained only with forward learning. While bidirectional learning has several design choices, here we evaluate the specific algorithm shown in Algorithm 1. In particular, we perform bidirectional learning for the first T_0 iterations, where the backward training is apply every $F+1$ steps. For the rest of $T - T_0$ iterations, we perform forward learning only. Throughout we set $F = 2$. Note that $\mathbf{Z}$ is fixed with its initial values throughout learning. Here we do not update $\mathbf{U}$ in forward learning and $\mathbf{V}$ in backward learning. This is partly because the learning rule of (27)-(28) has no effect when we use the expected values in hidden units.

Before applying Algorithm 1, the bias $\mathbf{b}$ is initialized to zero, and the weight ($\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$, $\mathbf{Z}$) is initialized with i.i.d. normal random variable with mean 0 and standard deviation of 0.01 (Hinton, 2012) except the following two changes. First, we set the mean of $\mathbf{W}^{[1]}$ as an identity matrix for real datasets, because using the previous value for prediction is clearly beneficial for these datasets. Second, we use the small standard deviation of 0.001 for the large dataset of 391 dimensions for faster convergence. The learning rate is adjusted according to AdaGrad (Duchi et al., 2011), where the the initial learning rate is optimized as we discuss in the following.

Throughout the experiments, we set the decay rate of the eligibility traces in (6) to zero: $\boldsymbol{\alpha}^{[t-1]} = \mathbf{x}^{[t-d]}$ and $\boldsymbol{\beta}^{[t-1]} = \mathbf{h}^{[t-d]}$. The delay d and the number of hidden units are varied for each dataset.

5.2. Synthetic Data

We first demonstrate the effectiveness of our bidirectional training in a synthetic setting of learning a one-dimensional noisy sawtooth wave, which is generated according to

$$x^{[t]} = \frac{t}{C} - \left\lfloor \frac{t}{C} \right\rfloor + \varepsilon_t, \text{ for } t = 0, 1, \dots \tag{31}$$

where C is the period of the noisy sawtooth wave, and ε_t is an i.i.d. normal random variable, whose mean is fixed at 0 and standard deviation at 0.01. The noisy sawtooth wave has large discontinuity at the end of each period, which makes hidden units essential for learning.

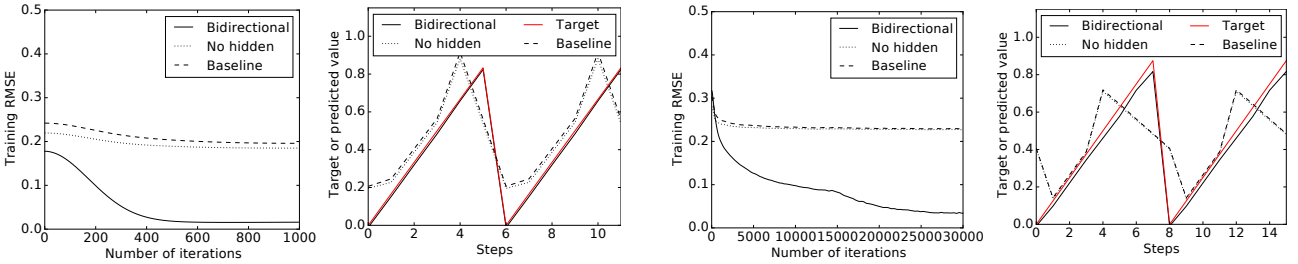
Here we train a DyBM with one hidden unit or no hidden units. Throughout, the delay is set $d = 4$, and the learning rate is initialized to $\eta = 1.0$. Bidirectional learning is continued for $T_0 = T/2$ iterations, where T is varied depending on C. In each iteration, we use one period of the noisy sawtooth wave and update the parameters with stochastic gradients.

Figure 3(a) and (c) show how learning progresses over iterations. The period of the noisy sawtooth wave is $C = 6$ in (a) and $C = 8$ in (c). Training is continued for $T = 1,000$ iterations for $C = 6$ and $T = 30,000$ iterations for $C = 8$. In every $F + 1$ iterations, we let the DyBM predict the one-step-ahead value for each of the C steps of one period during forward learning. We evaluate the root mean squared error (RMSE) of one-step-ahead predictions against true values. For clarity, the RMSE curves are smoothed with a Gaussian filter with window size of 50.

In the figure, the solid curves show the results with the bidirectionally trained DyBMs (*Bidirectional*). As a baseline, we also train the DyBM only with forward training and show the results with dashed curves (*Baseline*). The dotted curves show the results with the DyBM with no hidden units (*No hidden*). The comparison suggests that *Bidirectional* can substantially (by a factor of 10) improve the predictive accuracy over *Baseline*. Notice also that the hidden unit can hurt the predictive accuracy without bidirectional training. *No hidden* often exhibits lower RMSE than *Baseline*.

In Figure 3(c), the reduction of the RMSE accelerates at T_0 iterations, after which bidirectional learning is no longer performed. This suggests that, after learning appropriate values of $\mathbf{U}$ (namely, what features should be used for prediction) via bidirectional learning, it is better to optimally learn $(\mathbf{V}, \mathbf{W}, \mathbf{b})$ given the learned values of $\mathbf{U}$. Namely, although bidirectional learning help learn appropriate values of $\mathbf{U}$, it does not necessarily optimize all of the parameters of the DyBM. Recall also the discussion at the end of Section 4 that the backward DyBM is not exactly the same as the time-reversed DyBM.

Figure 3(b) and (d) show the values predicted by bidirection-



(a) RMSE ($C = 6$) (b) Prediction ($C = 6$) (c) RMSE ($C = 8$) (d) Prediction ($C = 8$)

Figure 3. Experimental results with noisy sawtooth waves. *Bidirectional* is a bidirectionally trained DyBM with one hidden unit, *Baseline* is a DyBM with hidden units trained only with forward learning, and *No hidden* is a bidirectionally trained DyBM without hidden units. In (a) and (c), the RMSE of one-step-ahead prediction with respect to training data is plotted against the number of iterations of training. In (b) and (d), the predicted values after training is plotted for two periods together with corresponding target values (red curves).

ally trained DyBMs (black curves) and the corresponding target values (red curves). For clarity, we use a noiseless sawtooth wave as the target by letting $\varepsilon_t = 0$ in (31). Observe that the bidirectionally trained DyBM well predicts the next value of the sawtooth wave, substantially better than the baseline (or the DyBM with no hidden unit). In particular, the bidirectionally trained DyBM can well predict the sharp drop at the end of each period. This is in contrast to the baseline, whose prediction is rather smoothed out over the period.

5.3. Real Data

Next, we demonstrate the effectiveness of bidirectional training on three real datasets, including the two datasets used in Dasgupta & Osogami (2017). The first dataset is the monthly sunspot number[1], which we will refer to as Sunspot. This time-series has one dimension and 2,820 steps (corresponding to January 1749 to December 1983). The second dataset is the weekly retail gasoline and diesel prices[2], which we will refer to as Price. This time-series has eight dimensions (corresponding to eight locations in the US) and 1,223 steps (corresponding to April 5th, 1993 to September 5th, 2016). Following Dasgupta & Osogami (2017), the first 67 % of each time-series is used for training, and the remaining 33 % is used for test. See Dasgupta & Osogami (2017) for further details about the first two datasets. The third dataset is the NOAA Global Surface Temperature[3], which consists of a real valued time-series of 1,635 steps ($t = 1, \ldots, 1635$) with 391 dimensions. We use the first 80 % of the time-series for training and the remaining 20 % for test.

We normalize the values of each dataset in a way that the values in a training data are in [0,1] for each dimension. Notice that this normalization differs from that in (Dasgupta & Osogami, 2017), where the values in training or test data are in [0,1] for each dimension. However, the use of test data for normalization is less appropriate. This difference in normalization has no effect on the Price dataset, but the results on the Sunspot dataset need to be renormalized to be compared against those in (Dasgupta & Osogami, 2017).

Here we train a DyBM with four hidden units or no hidden units. In each iteration, we use the whole training data (except the first d steps for forward learning and the last d steps for backward learning, which are used only to update the internal state of the DyBM) once and update the parameters with stochastic gradients.

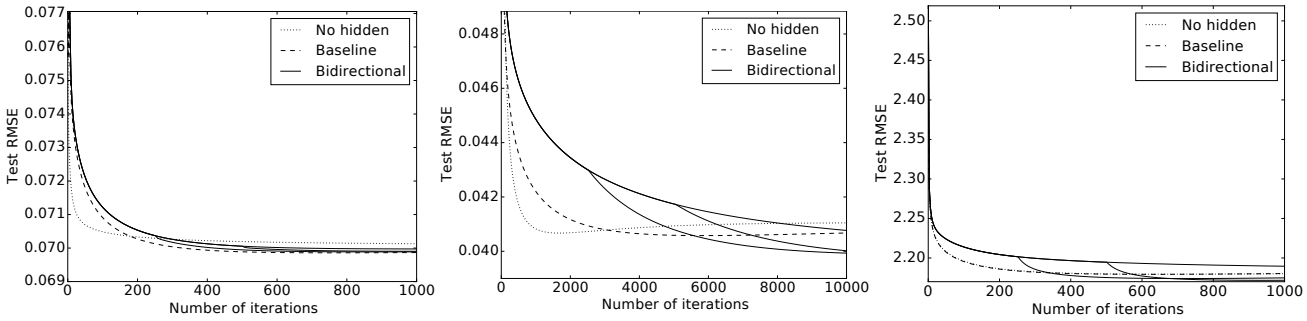
We find that the speed of convergence is sensitive to the initial learning rate. Here, we choose the initial learning rate from $\{2^0, 2^{-1}, 2^{-2}, \ldots\}$. Specifically, we choose 2^{-k} with the smallest k such that the training RMSE after the initial $T/100$ iterations is smaller than that with $2^{-(k+1)}$. Because the training RMSE tends to decrease with k up to a point and then increases with k, we usually choose the initial learning rate that minimizes the training RMSE after those initial iterations.

Figure 4 shows the RMSE of one-step-ahead prediction with respect to the test data after every $F + 1$ iterations of training. We show the results where the delay is set $d = 30$ for Sunspot, $d = 3$ for Price, and $d = 2$ for Temperature. However, we have also run experiments with $d \in \{20, 40\}$ for Sunspot and $d \in \{2, 4\}$ for Price and have found that these do not improve the accuracy for any of the three methods. For the large dataset of Temperature, we have been able to perform limited experiments due to its relatively heavy computational requirements. Again, we compare *Bidirectional* against *Baseline* and *No hidden*. However, we now vary $T_0 \in \{T/4, T/2, T\}$, so that each figure has three solid curves. The range of the vertical axis is chosen in a way that the upper limit corresponds to the RMSE with a naïve

[1] https://datamarket.com/data/set/22t4/

[2] https://www.eia.gov/dnav/pet/pet_pri_gnd_a_epm0_pte_dpgal_w.htm

[3] V4.00 of Air Temperature (air.mon.anom.nc) from https://www.esrl.noaa.gov/psd/data/gridded/data.noaaglobaltemp.html



(a) Sunspot dataset ($d = 30$) (b) Price dataset ($d = 3$) (c) Temperature dataset ($d = 2$)

Figure 4. Experimental results with real datasets. *Bidirectional* is a bidirectionally trained DyBM with four hidden units, *Baseline* is a DyBM with four hidden units trained only with forward learning, and *No hidden* is a DyBM with no hidden units trained only with forward learning. The RMSE of one-step-ahead prediction with respect to test data is plotted against the number of iterations of training.

prediction of using the preceding values as prediction.

For the Sunspot dataset (a), we find that *Bidirectional* does not improve upon *Baseline*, although having hidden units (*Bidirectional* and *Baseline*) can make the RMSE lower than *No hidden*. This means that the randomly set values of $\mathbf{U}$ is effective, and bidirectional learning does not find better values of $\mathbf{U}$. After 1,000 iterations, however, *Bidirectional* with $T_0 = T/4$ or $T_0 = T/2$ achieves the RMSE that is essentially indistinguishable from that with *Baseline*. The best RMSE achieved by the *Baseline* is 0.0698, which corresponds to 0.0657 when the dataset is normalized in the way of (Dasgupta & Osogami, 2017) and is lower than 0.0734 reported in (Dasgupta & Osogami, 2017).

For Price (b) and Temperature (c), *Bidirectional* improves upon *Baseline* particularly when the bidirectional learning is stopped after $T_0 < T$ iterations. *Bidirectional* reduces the RMSE more slowly than *Baseline* or *No hidden* but eventually outperforms the others, and the reduction of the RMSE can be accelerated by stopping the bidirectional training after $T_0 < T$ iterations. In particular, the best RMSE achieved by *Bidirectional* with $T_0 = T/4$ is 0.0399, which is lower than 0.0564 reported in (Dasgupta & Osogami, 2017). In addition, while *Baseline* and *No hidden* (namely, forward learning only) starts overfitting to training data and increases the RMSE with respect to the test data after some iterations, bidirectional training appears to avoid such overfit.

6. Conclusion

We have proposed bidirectional training for time-series models with hidden units. Namely, we consider two models that have a common set of parameters, where one model is trained with forward time-series and the other with backward time-series. Our key idea is that some of the parameters that are difficult to learn in one model can be effectively learned in the other model. Numerical experiments suggest that bidirectional training has the additional effect of avoiding overfit to training data.

The DyBM with hidden units analyzed in Sections 2–4 is new, and its analysis has two highlights, which have led to proposing bidirectional training. The first highlight is that the learning rule for $\mathbf{V}$ (hidden-to-visible weight) in (20)–(22) becomes equivalent to those for $\mathbf{W}$ (visible-to-visible weight) in (20)–(22) when the lower bound (14) is maximized. The second highlight is that we cannot learn the weight to hidden units ($\phi = (\mathbf{U}, \mathbf{Z})$) in the same way as the weight to visible units ($\theta = (\mathbf{V}, \mathbf{W}, \mathbf{b})$) due to the form of the gradient in (27).

Although we have demonstrated the effectiveness of bidirectional training in specific cases, its capabilities are not fully explored. Bidirectional training has many design choices that need further study. For example, one might want to use the gradients in (27)-(28), possibly with the approximation in (29), to update $(\mathbf{U}, \mathbf{Z})$ in forward learning and $(\mathbf{V}, \mathbf{Z})$ in backward learning. It would also be interesting to apply bidirectional training to other time-series models having parameters that represent the dependency between hidden values at one time and visible values at another time. In addition to DyBM and VAR with hidden units, these include Spiking Boltzmann Machine (Hinton & Brown, 1999) and Conditional Restricted Boltzmann Machine (Taylor et al., 2007). Because bidirectional training is largely complementary to other techniques for learning time-series, it would be interesting to investigate how bidirectional training improves performance when it is combined with these other techniques. We expect that this work opens up a line of research on more effective methods of bidirectional training.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR1304, Japan.

References

Bahadori, M. T., Liu, Y., and Xing, E. P. 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*, pp. 284–292, 2013.

Baldi, P., Brunak, S., Frasconi, P., Soda, G., and Pollastri, G. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.

Baum, L. E. and Petrie, T. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37:1554–1563, 1966.

Bottou, L. Online learning and stochastic approximations. In Saad, D. (ed.), *On-Line Learning in Neural Networks*, chapter 2, pp. 9–42. Cambridge University Press, 2009.

Chen, J. and Chaudhari, N. Protein secondary structure prediction with bidirectional LSTM networks. In *International Joint Conference on Neural Networks: Post-conference Workshop on Computational Intelligence Approaches for the Analysis of Bio-data (CI-BIO)*, August 2005.

Cheng, D., Bahadori, M. T., and Liu, Y. FBLG: A simple and effective approach for temporal dependence discovery from time series data. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 382–391, 2014.

Dasgupta, S. and Osogami, T. Nonlinear dynamic Boltzmann machines for time-series prediction. In *The 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, pp. 1833–1839, January 2017.

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

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Graves, A. and Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM networks and other neural network architectures. *Neural Networks*, 18(5-6): 602–610, 2005.

Hinton, G. E. A practical guide to training restricted Boltzmann machines. In Montavon, G., Orr, G. B., and Müller, K.-R. (eds.), *Neural Networks: Tricks of the Trade*, chapter 24, pp. 599–619. Springer-Verlag Berlin Heidelberg, 2012.

Hinton, G. E. and Brown, A. D. Spiking Boltzmann machines. In *Advances in Neural Information Processing Systems 12*, pp. 122–128. November 1999.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):17351780, 1997.

Jaeger, H. and Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):7880, 2004.

Jalali, A. and Sanghavi, S. Learning the dependence graph of time series with latent factors. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 473–480, 2012.

Kajino, H. A functional dynamic Boltzmann machine. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, August 2017.

Lütkepohl, H. *New Introduction to Multiple Time Series Analysis*. Springer-Verlag Berlin Heidelberg, 2005.

Osogami, T. Learning binary or real-valued time-series via spike-timing dependent plasticity. *CoRR*, abs/1612.04897, 2016. URL http://arxiv.org/abs/1612.04897.

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

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

Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Networks: The Official Journal of the International Neural Network Society*, 12(1):145–151, 1999.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L., and PDP Research Group (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, chapter 8. MIT Press, 1986.

Schuster, M. and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Sutskever, I. and Hinton, G. E. Learning multilevel distributed representations for high-dimensional sequences. In *Proceedings of the Eleventh International Conference*

on Artificial Intelligence and Statistics (AISTATS-07), volume 2, pp. 548–555. Journal of Machine Learning Research - Proceedings Track, 2007.

Sutskever, I., Hinton, G. E., and Taylor, G. W. The recurrent temporal restricted Boltzmann machine. In *Advances in Neural Information Processing Systems 21*, pp. 1601–1608. December 2008.

Taylor, G. W., Hinton, G. E., and Roweis, S. T. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems 19*, pp. 1345–1352. 2007.

Winkler, I., Panknin, D., Bartz, D., Müller, K.-R., and Haufe, S. Validity of time reversal for testing granger causality. *IEEE Transactions on Signal Processing*, 64(11):2746–2760, 2016.