# Dynamic Boltzmann Machines

Suyash Bagad, Rajat Patel

Guide: Prof. Udayan Ganguly

Mentor: Vivek Saraswat

May 6, 2019

## Defining Dynamic Boltzmann Machine

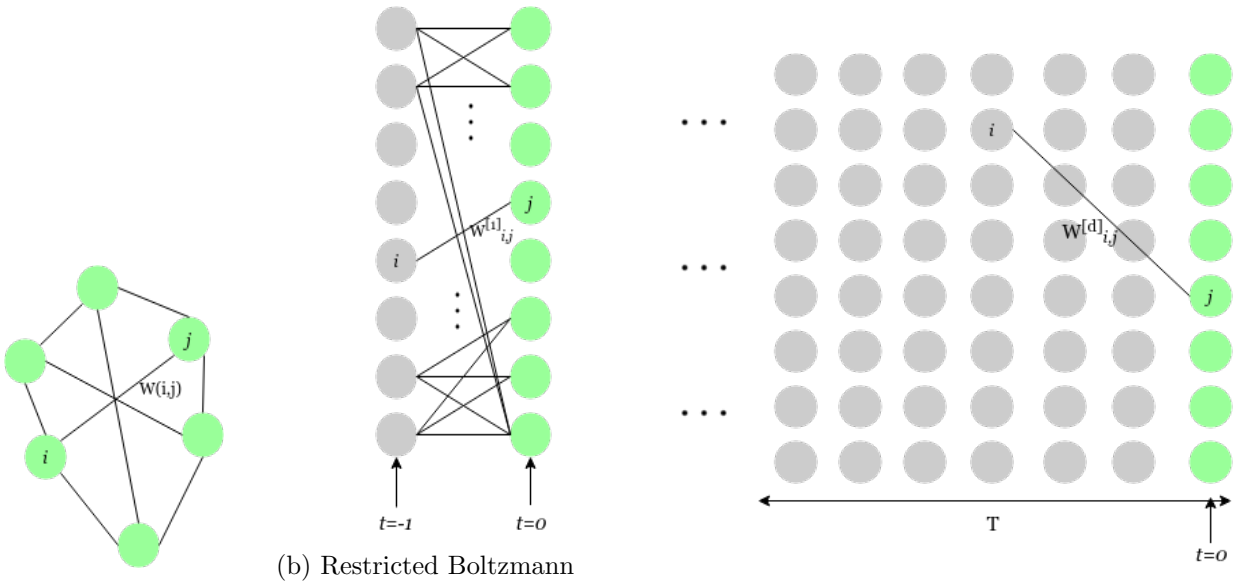
### 1.1 Conventional Boltzmann machine

A Boltzmann machine is a network of neurons that are mutually connected with weight (Figure 1a). Let $N$ be the number of units. $x_i \in \{0, 1\}$ is the value that $i$-th neuron takes and let $w_{i,j}$ be e the weight between the $i$-th unit and the $j$-th unit for $i, j \in [1, N]$. There aren't any self-loops in the formulation of Boltmann machines. Also, we assume that $w_{i,j} = w_{j,i}$ and $b_i$ is the bias term for $i$-th neuron. The parameters of a Boltzmann machine are defined in the following way:

$$\theta = (\mathbf{b}, \mathbf{W}), \mathbf{b} = (b_i)_{i\in[1,N]}, \mathbf{W} = (w_{i,j})_{(i,j)\in[1,N]^2}$$

and the state is defined as $\mathbf{x} = (x_i)_{i\in[1,N]}$. The energy of the state, $\mathbf{x}$, for the $N$ units of the Boltzmann machine having $\theta$ is given by

$$E_\theta(\mathbf{x}) = -\mathbf{b}^T\mathbf{x} - \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{1}$$




(a) Boltzmann machine (b) Restricted Boltzmann machine (c) Dynamic Boltzmann machine

Figure 1: The basic structures of the variants of the Boltzmann machines

The (equilibrium) probability of generating a particular state, $\mathbf{x}$, is given by

$$P_\theta(\mathbf{x}) = \frac{e^{-\tau E_\theta(\mathbf{x})}}{\sum_{\tilde{\mathbf{x}}} e^{-\tau E_\theta(\tilde{\mathbf{x}})}} \tag{2}$$

where the denominator is referred to as *Partition function* wherein the summation is taken over all of the possible states of a binary vector of length $N$ and $\tau$ is the parameter called temperature. For a given set of vectors $\mathcal{D}$, the Boltzmann machine can be trained by in the direction of maximizing the log likelihood of $\mathcal{D}$:

$$\nabla_\theta log P_\theta(\mathcal{D}) = \sum_{\mathbf{x}\in\mathcal{D}} \nabla_\theta log P_\theta(\mathbf{x}) \tag{3}$$

where $P_\theta(\mathcal{D}) = \prod_{\mathbf{x}\in\mathcal{D}} P_\theta(\mathbf{x})$ since the states are assumed to be independent of each other.

$$\implies \nabla_\theta log P_\theta(\mathbf{x}) = -\tau\big(\nabla_\theta E_\theta(\mathbf{x}) - \sum_{\tilde{x}} P_\theta(\tilde{x})\nabla_\theta E_\theta(\tilde{x})\big) \tag{4}$$

Now, to increase the likelihood of a data point, $\mathbf{x}$, we should update $W_{i,j}$ by

$$W_{i,j} \leftarrow W_{i,j} + \eta(x_i x_j - \langle X_i, X_j\rangle_\theta) \tag{5}$$

where $\eta$ is the learning rate and $\langle X_i, X_j\rangle_\theta$ denotes the expected value of the product of the values of the $i$-th neuron and the $j$-th neuron with respect to $P_\theta$.

### 1.2 Restricted Boltzmann machine

An interesting variant of the Boltzmann machines is the Restricted Boltzmann machine (Figure 1b) which has 2 layers and any two neurons from the same layer aren't connected. Each neuron in a particular layer is connected to all other neurons from the other layer. In this case, $P_\theta(\mathbf{x})$ is simplified. Let $t \in 1, 2$, $\mathbf{x}^{[t]}$ be the value of the neurons of layer $t$. The value of the units in the second layer $\mathbf{x}^{[2]}$, is conditionally independent of each other given $\mathbf{x}^{[1]}$ . Then, the conditional probability of $\mathbf{x}^{[2]}$ given $\mathbf{x}^{[1]}$ is given by

$$P_\theta(\mathbf{x}^{[2]}|\mathbf{x}^{[1]}) = \prod_{j\in[1,N]} P_{\theta,j}(x_j^{[2]}|\mathbf{x}^{[1]}) \tag{6}$$

$$= \prod_{j\in[1,N]} \frac{e^{-\tau E_\theta(x_j^{[2]}|\mathbf{x}^{[1]})}}{\sum_{x_j^{[2]}\in\{0,1\}} e^{-\tau E_\theta(x_j^{[2]}|\mathbf{x}^{[1]})}} \tag{7}$$

We define the energy corresponding to a particular state of neuron j in layer 2 as follows:

$$E_\theta(x_j^{[2]}|\mathbf{x}^{[1]}) = -b_j x_j^{[2]} - (\mathbf{x}^{[1]})^T W_{:,j}^{[1]} x_j^{[2]} \tag{8}$$

where $W^{[1]}$ is the weight matrix between neurons in layer 1 and all neurons in layer 2.

### 1.3 Dynamic Boltzmann machine

The Dynamic Boltzmann machine beautifully incorporates the notion of time in the existing framework of Boltzmann machines. It consists of infinite layers of neurons unfolded in time. There are no neuronal connections in space, the only connections between neurons are in time.

Formally, we define the DyBM-$T$ as the Boltzmann machine having $T$ layers from $-T+1$ to 0, where $T$ is a positive integer or $\infty$. Consider a network of $N$ neurons which aren't connected in space. We let $\mathbf{x} := (\mathbf{x}^{[t]})_{t\in(-T+1,0)}$ where $\mathbf{x}^{[t]}$ denotes the values taken by $N$ neurons at time $t$, $\mathbf{b}$ be the bias for the $N$ neurons at the current time instant ($t = 0$) and let $\mathbf{W}^{[\delta]} := (W^{[\delta]}_{i,j})_{(i,j)\in[1,N]^2}$ be matrix of weights where $W^{[\delta]}_{i,j}$ denotes the weight between the $i$-th neuron at time $-\delta$ and the $j$-th neuron at time 0 for any $t$. The layers for $t \leq -1$ could well have arbitrary values of weights and biases for connections among themselves but we'll see in the mathematical analysis that these values have no effect in the DyBM. All the weight and bias terms associated with the layer 0 would play a role in DyBM.

The DyBM is proposed as a model of a time-series in the following sense. Specifically, given a history $\mathbf{x}^{(-\infty,-1]}$ of a time series, the DyBM-$T$ gives the probability of the next values, $\mathbf{x}^{[0]}$ of the time series with $P_\theta(\mathbf{x}^{[0]}|\mathbf{x}^{(-T,-1]})$. Interestingly, DyBM-$T$ can be interpreted as a $(T-1)$-st order *Markov model*, where the next values are conditionally independent of the history given the values of the last $(T-1)$ steps.

Using the conditional probability given by a DyBM-$T$, the probability of a sequence, $\mathbf{x} = \mathbf{x}^{(-L,0]}$ of length $L$ is given by (See figure 2)

$$p(\mathbf{x}) = \prod_{t\in[-L+1,-1]} P_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{(t-T,t-1]}) \tag{9}$$

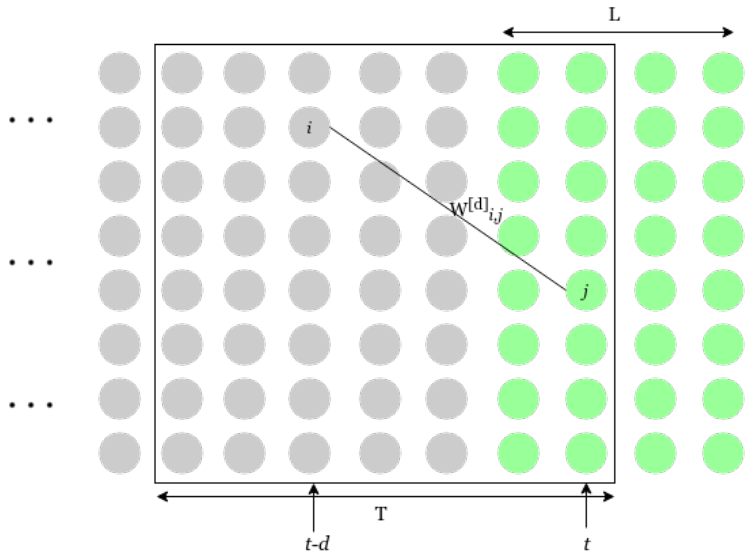


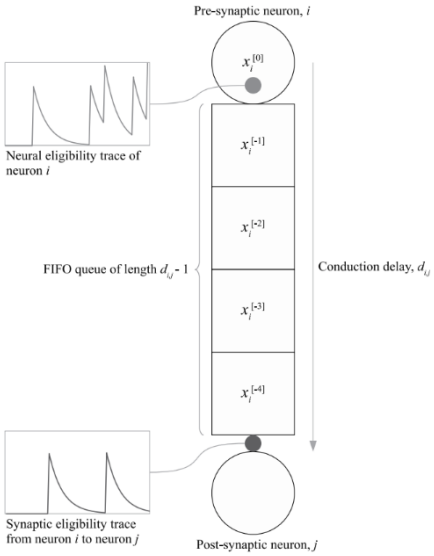

Figure 2: Conditional independence given by a DyBM-$T$ in an $L$-length sequence prediction

The values are set zero if there are no corresponding history.

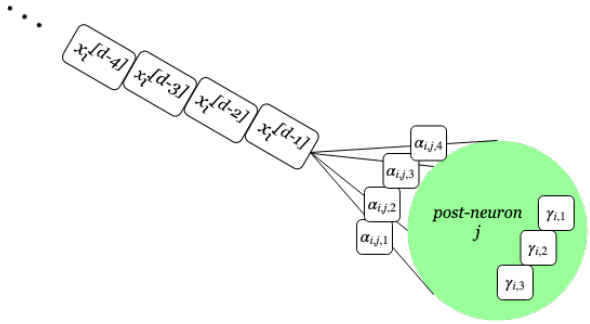
## 1.4 Structure of DyBM

A DyBM consists of a set of neurons having memory units and first-in-first-out (FIFO) queues. Let $N$ be the number of neurons. Each neuron takes a binary value of either 0 or 1 at each moment. For $j \in [1, N]$, let $x^{[t]}_j$ be the value of the $j$-th neuron at time $t$. A neuron $i$ can be

connected to neuron $j$ with a FIFO queue of length $d_{i,j}1$, where $d_{i,j}$ is the axonal or synaptic delay of conductance, or conduction delay, from pre-synaptic neuron $i$ to post-synaptic neuron $j$. See figure 3.




(a) Pre and Post-neurons with the FIFO queue of length $d_{i,j} - 1$




(b) Synaptic and neural eligibility traces, $\alpha_{i,j,k}\ \forall\ k\ \in \mathcal{K}$, $\gamma_{i,l}\ \forall\ l \in \mathcal{L}$, as derived in equations (26), (28)

Figure 3: A DyBM consists of a network of neurons and memory units. Between two neurons, we have two separate uni-directional FIFO queues (each of length $d_{i,j} - 1$) storing the past states of the pre-synaptic neuron. Also, each neuron has the memory unit for storing neural eligibility traces, which summarize the neurons activities in the past. A synaptic eligibility trace is associated with a synapse between a pre-synaptic neuron and a post-synaptic neuron, and summarizes the spikes that have arrived at the synapse, via the FIFO queue, from the pre-synaptic neuron.

The structural features that distinguish a DyBM from a Boltzmann machine are the eligibility traces. The DyBM aggregates information about the spikes in the past into neural eligibility traces ($\gamma_{i,l}\ \forall i \in [1, N], l \in \mathcal{L}$) and synaptic eligibility traces ($\alpha_{i,j,k}\ \forall (i,j) \in [1, N]^2, k \in \mathcal{K}$), which are stored in the memory units. The value of a synaptic eligibility trace increases when the spike from a pre-synaptic neuron reaches a post-synaptic neuron and decreases otherwise. Similarly, the value of the neural eligibility trace increases when the neuron fires; and decreases otherwise.

DyBM allows us to have multiple eligibility traces with varying decay rates for each neuron and for each synapse. The use of varying decay rates is aligned with the approximation of the hyperbolic decay for long-term memory. The use of sum of geometric series is attributed to fit the required hyperbolic delay. The use multiple traces for memorising events could be interpreted biologically as dendrites which too are critical in memory formation.

### Spike-timing dependent plasticity

We derive a learning rule for a DyBM-$T$ in such a way that the log likelihood of a given (set of) time-series is maximized. This is similar to what we have in the case of conventional Boltzmann machine. Setting the weights of a DyBM-$T$ to follow some specific constraints, we'll see that the learning rule is particularly simplified in the limit of $T \to \infty$. In this section, we'll also prove mathematically that such a learning rule for DyBM is close to the spike-timing dependent plasticity (STDP).

### 2.1 General learning rule of dynamic Boltzmann machines

The log likelihood of a given set, $\mathcal{D}$, of time-series can be maximized by maximizing the sum of the log likelihood of $\mathbf{x} \in \mathcal{D}$. By (9), the log likelihood of $\mathbf{x} = \mathbf{x}^{(-L,0]}$ has the following gradient:

$$\nabla_\theta log\ p(\mathbf{x}) = \sum_{t=-L+1}^{0} \nabla_\theta log\ P_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{(t-T,t-1]}) \tag{10}$$

Using the conditional independence of (50), the learning rule can be derived as:

$$\begin{aligned} &\implies \nabla_\theta log\ P_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{(t-T,t-1]}) \\ &\quad = -\tau^{-1} \sum_{j=1}^{N} \Big( \nabla_\theta E_{\theta,j}(x_j^{[t]}|\mathbf{x}^{[:t-1]}) - \sum_{\tilde{x}_j^{[t]} \in \{0,1\}} P_{\theta,j}(\tilde{x}_j^{[t]}|\mathbf{x}^{(t-T,t-1]}) \nabla_\theta E_{\theta,j}(\tilde{x}_j^{[t]}|\mathbf{x}^{[:t-1]}) \Big) \end{aligned} \tag{11}$$

Now, once we have $E_{\theta,j}(x_j^{[t]}|\mathbf{x}^{[:t-1]})$, a learning rule for parameters could be derived.

### 2.2 Deriving a specific learning rule

To derive a learning rule that has the characteristics of STDP with LTP and LTD, we consider the weight of the form illustrated in equation (15) and Figure 4.

We'll qualitatively explain the reason for selecting the profile of $\hat{W}_{i,j}^{[\delta]}$ as mentioned in equation (16). Firstly, neuron $i$ is the pre-neuron and $j$ is the post-neuron. Now, when the time taken by a spike to reach from $i$ to $j$ is $\delta = d_{i,j}^{+}$, where $d_{i,j}$ is the conduction delay between the said pair of neurons, it implies that the spike from pre-neuron caused a spike in the post-neuron and thus the weight should be maximum for this case. This explains the decaying profile of $\hat{W}_{i,j}^{[\delta]}$ as $\delta$ increases. On the other side, when a spike travelling from pre-neuron to post-neuron reaches the post-neuron even before the conduction delay, then there is supposed to be suppressing of the synaptic weight and thus a negative value is assigned to $\hat{W}_{i,j}^{[\delta]}$. For negative $\delta$, we can interpret it as the delay between the reversed order of $(i, j)$, i.e. now the pre-neuron is $j$ and post-neuron is $i$. Thus we get the profile of $\hat{W}_{j,i}^{[-\delta]}$.

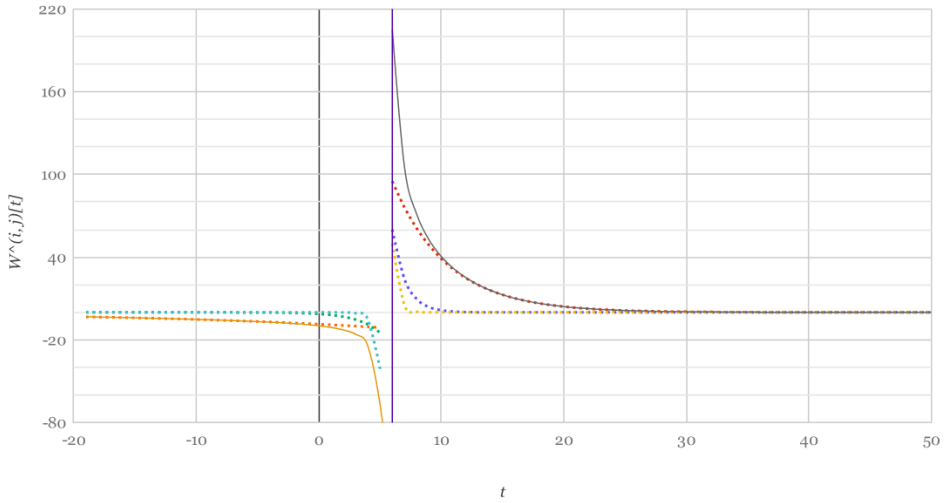
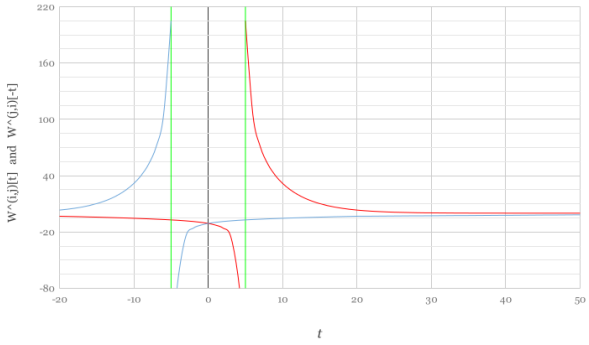
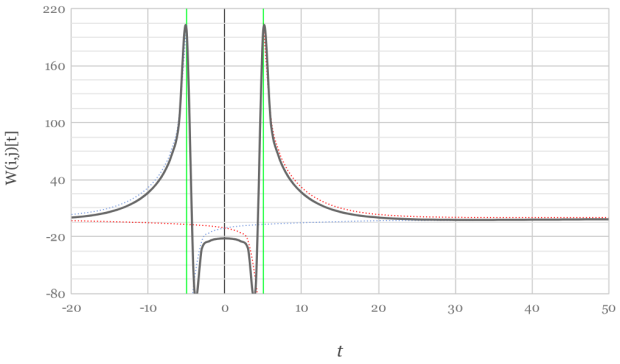



Figure 4: The figure shows equation (15) with particular forms of the equation (16). The solid curves represent $\hat{W}_{i,j}^{[\delta]}$ as a function of $\delta$, while the dotted ones represent the individual geometric series attributed to different values of $\lambda_k$, $\mu_l$. Note the the discontinuity of $\hat{W}_{i,j}^{[\delta]}$ at $d_{i,j}$.




(a) The red curve represents the profile of $\hat{W}_{i,j}^{[\delta]}$ for a particular set of values of $\lambda_k$, $\mu_l$ and $d_{i,j} = 5$. Similarly, the blue one represents the profile of $\hat{W}_{j,i}^{[-\delta]}$.




(b) The solid curve in grey shows the profile of $W_{i,j}^{[\delta]}$ which is essentially $\hat{W}_{i,j}^{[\delta]} + \hat{W}_{j,i}^{[-\delta]}$. Note here that the weight maximises for $\delta = d_{i,j}^{+}$ and $\delta = -d_{i,j}^{-}$

Figure 5: The weight constraints and the profile of weight $W_{i,j}^{[\delta]}$ between $i$-th neuron at time $(t-\delta)$ and $j$-th neuron at time $(t)$ with respect to delay $\delta$.

With the above specifications and letting $T \to \infty$, we can now represent the energy with respect to the $j$-th neuron at $t = 0$ given the values in $t \in (t-T, t-1]$ as written in equation (13). Thus we have total energy

$$E_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[:t-1]}) = \sum_{j=1}^{N} E_\theta(x_j^{[t]}|\mathbf{x}^{[:t-1]}) \tag{12}$$

$$E_\theta(x_j^{[t]}|\mathbf{x}^{[:t-1]}) = -b_j x_j^{[t]} + \sum_{\delta=-\infty}^{t-1} (\mathbf{x}^{[\delta]})^T W_{:,j}^{[t-\delta]} x_j^{[t]} \tag{13}$$

$$(\mathbf{x}^{[\delta]})^T W_{:,j}^{[t-\delta]} = \sum_{i=1}^{N} x_i^{[\delta]} W_{i,j}^{[t-\delta]} \tag{14}$$

One important thing to note here is that $\mathbf{W}^{[\delta]}$ is assumed to depend only on the time difference between the pre-neuron and the post-neuron. We could well have time-dependent weight matrix $\mathbf{W}^{[\delta]}(t)$, $t \in (-L, 0]$.
We need to have $W_{i,j}^{[\delta]}$ to proceed further. For tractable learning and inference, we assume the following form of weight:

$$W_{i,j}^{[\delta]} = \hat{W}_{i,j}^{[\delta]} + \hat{W}_{j,i}^{[-\delta]} \tag{15}$$

$$\hat{W}_{i,j}^{[\delta]} = \begin{cases} 0 & \delta = 0 \\ \sum_{k\in\mathcal{K}} u_{i,j,k} \lambda_k^{\delta - d_{i,j}} & \delta \geq d_{i,j} \\ \sum_{l\in\mathcal{L}} -v_{i,j,k} \mu_l^{-\delta} & otherwise \end{cases} \tag{16}$$

$$\hat{W}_{j,i}^{[-\delta]} = \begin{cases} 0 & \delta = 0 \\ \sum_{k\in\mathcal{K}} u_{j,i,k} \lambda_k^{-\delta - d_{i,j}} & \delta \leq -d_{i,j} \\ \sum_{l\in\mathcal{L}} -v_{j,i,k} \mu_l^{\delta} & otherwise \end{cases} \tag{17}$$

Here, $\lambda_k, \mu_l \in (0,1)\ \forall k \in \mathcal{K}, l \in \mathcal{L}$. We will learn the values of $u_{i,j,k}$ and $v_{i,j,l}$ based on training dataset. It is assumed that $\lambda_k, \mu_l\ \forall k \in \mathcal{K}, l \in \mathcal{L}$ and $d_{i,j}\ \forall (i,j) \in [1,N]^2$. With an analogy to biological neural networks, these given parameters $(\lambda_k, \mu_l, d_{i,j})$ are determined based on physical constraints or chemical properties, while the weight ($u_{i,j,k}$ and $v_{i,j,l}$) and the bias ($\mathbf{b}$) are learned based on the neural spikes ($\mathbf{x}$). The reason we use sum of multiple geometric series is that the sum of geometric functions can well approximate a hyperbolic function, whose value decays more slowly than any geometric functions. This slow decay is considered to be essential for long-term memory. We have

$$W_{i,j}^{[\delta]} = \begin{cases} 0 & & \delta = 0 \\ \sum_{k\in\mathcal{K}} u_{j,i,k} \lambda_k^{-\delta - d_{i,j}} & + \sum_{l\in\mathcal{L}} -v_{i,j,k} \mu_l^{-\delta} & \delta \leq -d_{i,j} \\ \sum_{l\in\mathcal{L}} -v_{j,i,k} \mu_l^{\delta} & + \sum_{l\in\mathcal{L}} -v_{i,j,k} \mu_l^{-\delta} & \delta \in (-d_{i,j}, d_{i,j}) \\ \sum_{k\in\mathcal{K}} u_{i,j,k} \lambda_k^{\delta - d_{i,j}} & + \sum_{l\in\mathcal{L}} -v_{j,i,k} \mu_l^{\delta} & \delta \geq d_{i,j} \end{cases} \tag{18}$$

Now, we write the expression for $W_{i,j}^{[-\delta]}$

$$W_{i,j}^{[-\delta]} = \begin{cases} 0 & & \delta = 0 \\ \sum_{k\in\mathcal{K}} u_{j,i,k} \lambda_k^{\delta - d_{i,j}} & + \sum_{l\in\mathcal{L}} -v_{i,j,k} \mu_l^{\delta} & \delta \geq d_{i,j} \\ \sum_{l\in\mathcal{L}} -v_{j,i,k} \mu_l^{-\delta} & + \sum_{l\in\mathcal{L}} -v_{i,j,k} \mu_l^{\delta} & \delta \in (-d_{i,j}, d_{i,j}) \\ \sum_{k\in\mathcal{K}} u_{i,j,k} \lambda_k^{-\delta - d_{i,j}} & + \sum_{l\in\mathcal{L}} -v_{j,i,k} \mu_l^{-\delta} & \delta \in (-\infty, d_{i,j}] \end{cases} \tag{19}$$

Substituting equation (19) in equation (13) (assuming $|t| < |d_{i,j}|$ without loss of generality), we get

$$\sum_{\delta=-\infty}^{t-1} (\mathbf{x}^{[\delta]})^T W_{:,j}^{[t-\delta]} = \sum_{\delta=-\infty}^{t-1} \sum_{i=1}^{N} x_i^{[\delta]} W_{i,j}^{[t-\delta]} = \sum_{i=1}^{N} \Big( \sum_{\delta=-\infty}^{t-d_{i.j}} x_i^{[\delta]} W_{i,j}^{[t-\delta]} + \sum_{\delta=t-d_{i.j}+1}^{t-1} x_i^{[\delta]} W_{i,j}^{[t-\delta]} \Big) \tag{20}$$

$$\begin{aligned}
\sum_{\delta=-\infty}^{t-d_{i.j}} x_i^{[\delta]} W_{i,j}^{[t-\delta]} &= \sum_{\delta=-\infty}^{-d_{i.j}} x_i^{[t+\delta]} W_{i,j}^{[-\delta]} \\
&= \sum_{\delta=-\infty}^{-d_{i.j}} x_i^{[t+\delta]} \Big( \sum_{k\in\mathcal{K}} u_{i,j,k} \lambda_k^{-\delta-d_{i,j}} + \sum_{l\in\mathcal{L}} -v_{j,i,k} \mu_l^{-\delta} \Big) \\
&= \sum_{k\in\mathcal{K}} u_{i,j,k} \Big( \sum_{\delta=-\infty}^{-d_{i.j}} \lambda_k^{-\delta-d_{i,j}} x_i^{[t+\delta]} \Big) - \sum_{l\in\mathcal{L}} v_{j,i,k} \Big( \sum_{\delta=-\infty}^{-d_{i.j}} \mu_l^{-\delta} x_i^{[t+\delta]} \Big)
\end{aligned} \tag{21}$$

$$\begin{aligned}
\sum_{\delta=t-d_{i.j}+1}^{t-1} x_i^{[\delta]} W_{i,j}^{[t-\delta]} &= \sum_{\delta=-d_{i.j}+1}^{-1} x_i^{[t+\delta]} W_{i,j}^{[-\delta]} \\
&= \sum_{\delta=-d_{i.j}+1}^{-1} x_i^{[t+\delta]} \Big( \sum_{l\in\mathcal{L}} -v_{j,i,k} \mu_l^{-\delta} + \sum_{l\in\mathcal{L}} -v_{i,j,k} \mu_l^{\delta} \Big) \\
&= -\sum_{l\in\mathcal{L}} v_{j,i,k} \Big( \sum_{\delta=-d_{i.j}+1}^{-1} \mu_l^{-\delta} x_i^{[t+\delta]} \Big) - \sum_{l\in\mathcal{L}} v_{i,j,k} \Big( \sum_{\delta=-d_{i.j}+1}^{-1} \mu_l^{\delta} x_i^{[t+\delta]} \Big)
\end{aligned} \tag{22}$$

Now, substituting equations (21), (22) in equation (20), we get

$$\begin{aligned}
\sum_{\delta=-\infty}^{t-1} (\mathbf{x}^{[\delta]})^T W_{:,j}^{[t-\delta]} = \sum_{i=1}^{N} \Big\{ & \sum_{k\in\mathcal{K}} u_{i,j,k} \Big( \sum_{\delta=-\infty}^{-d_{i.j}} \lambda_k^{-\delta-d_{i,j}} x_i^{[t+\delta]} \Big) - \sum_{l\in\mathcal{L}} v_{j,i,k} \Big( \sum_{\delta=-\infty}^{-d_{i.j}} \mu_l^{-\delta} x_i^{[t+\delta]} \Big) \\
& - \sum_{l\in\mathcal{L}} v_{j,i,k} \Big( \sum_{\delta=-d_{i.j}+1}^{-1} \mu_l^{-\delta} x_i^{[t+\delta]} \Big) - \sum_{l\in\mathcal{L}} v_{i,j,k} \Big( \sum_{\delta=-d_{i.j}+1}^{-1} \mu_l^{\delta} x_i^{[t+\delta]} \Big) \Big\}
\end{aligned} \tag{23}$$

$$\begin{aligned}
\implies \sum_{\delta=-\infty}^{-1} (\mathbf{x}^{[\delta]})^T W_{:,j}^{[t-\delta]} = \sum_{i=1}^{N} \Big\{ & \sum_{k\in\mathcal{K}} u_{i,j,k} \Big( \sum_{\delta=-\infty}^{-d_{i.j}} \lambda_k^{-\delta-d_{i,j}} x_i^{[t+\delta]} \Big) \\
& - \sum_{l\in\mathcal{L}} v_{i,j,k} \Big( \sum_{\delta=-d_{i.j}+1}^{-1} \mu_l^{\delta} x_i^{[t+\delta]} \Big) - \sum_{l\in\mathcal{L}} v_{j,i,k} \Big( \sum_{\delta=-\infty}^{-1} \mu_l^{-\delta} x_i^{[t+\delta]} \Big) \Big\}
\end{aligned} \tag{24}$$

$$\implies \sum_{\delta=-\infty}^{t-1} (\mathbf{x}^{[\delta]})^T W_{:,j}^{[-\delta]} = \sum_{i=1}^{N} \Big\{ \sum_{k\in\mathcal{K}} u_{i,j,k} \alpha_{i,j,k} - \sum_{l\in\mathcal{L}} v_{i,j,k} \beta_{i,j,l} - \sum_{l\in\mathcal{L}} v_{j,i,k} \gamma_{i,l} \Big\} \tag{25}$$

where $\alpha_{i,j,k}, \beta_{i,j,l}$ and $\gamma_{i,l}$ are the quantities, which we refer to as eligibility traces, that depend on $x_i^{(-\infty,t-1]}$, the history of the $i$-th unit with respect to time $t$:

$$\alpha_{i,j,k}^{[t-1]} := \sum_{\delta=-\infty}^{-d_{i.j}} \lambda_k^{-\delta-d_{i,j}} x_i^{[t+\delta]} \tag{26}$$

$$\beta_{i,j,l}^{[t-1]} := \sum_{\delta=-d_{i.j}+1}^{-1} \mu_l^{\delta} x_i^{[t+\delta]} \tag{27}$$

$$\gamma_{i,l}^{[t-1]} := \sum_{\delta=-\infty}^{-1} \mu_l^{-\delta} x_i^{[t+\delta]} \tag{28}$$

We can compute traces $\alpha_{i,j,k}$ & $\gamma_{i,l}$ recursively but not $\beta_{i,j,l}$ since the summation in the latter case doesn't start from $-\infty$.

$$\alpha_{i,j,k}^{[t]} \leftarrow \lambda_k(\alpha_{i,j,k}^{[t-1]} + x_j^{[t+d_{i,j}-1}) \tag{29}$$

$$\gamma_{i,l}^{[t]} \leftarrow \mu_l(\gamma_{i,l}^{[t-1]} + x_i^{[t]}) \tag{30}$$

Thus, the total energy can be written as:

$$E_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[:t-1]}) = \sum_{j=1}^{N} \Big\{ -b_j - \sum_{i=1}^{N}\sum_{k\in\mathcal{K}} u_{i,j,k}\alpha_{i,j,k}^{[t-1]} + \sum_{i=1}^{N}\sum_{l\in\mathcal{L}} v_{j,i,k}\beta_{i,j,l}^{[t-1]} + \sum_{i=1}^{N}\sum_{l\in\mathcal{L}} v_{i,j,k}\gamma_{i,l}^{[t-1]} \Big\} x_j^{[t]} \tag{31}$$

Typically, we would be using the time-series model in which we predict $\mathbf{x}^{[0]}$, the state at $t = 0$. But, here we compute the derivatives, in a general framework, of $E_{\theta,j}(x_j^{[t]}|\mathbf{x}^{[:t-1]})$ wrt to the parameters:

$$\frac{\partial}{\partial b_j} E_{\theta,j}(x_j^{[t]}|\mathbf{x}^{[:t-1]}) = -x_j^{[t]} \tag{32}$$

$$\frac{\partial}{\partial u_{i,j,k}} E_{\theta,j}(x_j^{[t]}|\mathbf{x}^{[:t-1]}) = -\alpha_{i,j,k}^{[t-1]} x_j^{[t]} \tag{33}$$

$$\frac{\partial}{\partial v_{i,j,k}} E_{\theta,j}(x_j^{[t]}|\mathbf{x}^{[:t-1]}) = \beta_{i,j,l}^{[t-1]} x_j^{[t]} \tag{34}$$

$$\frac{\partial}{\partial v_{j,i,k}} E_{\theta,j}(x_j^{[t]}|\mathbf{x}^{[:t-1]}) = \gamma_{i,l}^{[t-1]} x_j^{[t]} \tag{35}$$

By plugging the above derivatives into (11), we have, for $i, j \in [1, N], k \in \mathcal{K}, l \in \mathcal{L}$, that

$$\frac{\partial}{\partial b_j} log\ P_\theta(x_j^{[t]}|\mathbf{x}^{[:t-1]}) = \tau^{-1}(x_j^{[t]} - \langle X_j^{[t]}\rangle_\theta) \tag{36}$$

$$\frac{\partial}{\partial u_{i,j,k}} log\ P_\theta(x_j^{[t]}|\mathbf{x}^{[:t-1]}) = \alpha_{i,j,k}^{[t-1]}\tau^{-1}(x_j^{[t]} - \langle X_j^{[t]}\rangle_\theta) \tag{37}$$

$$\frac{\partial}{\partial v_{i,j,k}} log\ P_\theta(x_j^{[t]}|\mathbf{x}^{[:t-1]}) = -\beta_{i,j,l}^{[t-1]}\tau^{-1}(x_j^{[t]} - \langle X_j^{[t]}\rangle_\theta) - \gamma_{j,l}^{[t-1]}\tau^{-1}(x_j^{[t]} - \langle X_j^{[t]}\rangle_\theta) \tag{38}$$

$$\begin{aligned}\langle X_j^{[t]}\rangle_\theta &= \mathbb{E}_{\theta,j}(x_j^{[t]}|\mathbf{x}^{[:t-1]}) \\ &= 1 \times P_{\theta,j}(1|\mathbf{x}^{[:t-1]}) + 0 \times P_{\theta,j}(0|\mathbf{x}^{[:t-1]}) \\ &= \frac{e^{-\tau^{-1}E_{\theta,j}(1|\mathbf{x}^{[:t-1]})}}{1+e^{-\tau^{-1}E_{\theta,j}(1|\mathbf{x}^{[:t-1]})}}\end{aligned} \tag{39}$$

Here, the first term of the denominator is 1, because $E_{\theta,j}(0|\mathbf{x}^{[:t-1]}) = 0$. To maximize the likelihood of a given set, $\mathcal{D} = \{\mathbf{x}^{[t]}|t \in (-L, 0]\}$, of sequences, the parameters $\theta$ can be updated with:

$$\theta \leftarrow \theta - \eta \sum_{\mathbf{x}\in\mathcal{D}} \nabla_\theta log\ P_\theta(\mathbf{x}|\mathbf{x}^{(:t-1]}) \tag{40}$$

Specifically, at time $t$, the DyBM updates its parameters according to

$$b_j \leftarrow b_j - \eta\tau^{-1}(x_j^{[t]} - \langle X_j^{[t]}\rangle_\theta) \tag{41}$$

$$u_{i,j,k} \leftarrow u_{i,j,k} - \alpha_{i,j,k}^{[t-1]}\tau^{-1}(x_j^{[t]} - \langle X_j^{[t]}\rangle_\theta) \tag{42}$$

$$v_{i,j,l} \leftarrow v_{i,j,l} + \beta_{i,j,l}^{[t-1]}\tau^{-1}(x_j^{[t]} - \langle X_j^{[t]}\rangle_\theta) + \gamma_{j,l}^{[t-1]}\tau^{-1}(x_j^{[t]} - \langle X_j^{[t]}\rangle_\theta) \tag{43}$$

### 2.3 Energy Correspondence with Hebbian Learning Rule

The conditional energy of $j$-th neuron of the DyBM at time $t$ is given by equation (44). The DyBM tries to attain steady state by reaching to a minimum energy state. Now, in order to lower it's energy, we train the parameters $b_j, u_{i,j,k}, v_{i,j,l}$ so as to minimise energy in every iteration. We can clearly note that higher the value of the bias, lower is the energy and thus higher is the probability of that neuron spiking.

$$E_{\theta,j}(x_j^{[t]}|\mathbf{x}^{[:t-1]}) = \Big\{ - b_j + \sum_{i=1}^{N}\sum_{k\in\mathcal{K}} u_{i,j,k}\alpha_{i,j,k}^{[t-1]} + \sum_{i=1}^{N}\sum_{l\in\mathcal{L}} v_{j,i,k}\beta_{i,j,l}^{[t-1]} + \sum_{i=1}^{N}\sum_{l\in\mathcal{L}} v_{i,j,k}\gamma_{i,l}^{[t-1]}\Big\}x_j^{[t]} \quad (44)$$

We now explain how each remaining term in the equation corresponds to either Long Term Potentation(LTP) or Long Term Depression(LTD). Refer to figure (6). Consider the case where pre-neuron spikes just before $t - d_{i,j}$ then value of $\alpha_{i,j,k}^{[t-1]}$ is large. Further, the energy will decrease when $\alpha_{i,j,k}^{[t-1]}$ is large and the corresponding $u_{i,j,k}$ is positive(excitatory synapse) and large too. Biologically this corresponds to pre-neuron $i$ spike reaching post neuron $j$ just before the post neuron spike which leads to LTP. So, the second term corresponds to LTP.

Similarly for third term the biological equivalence is the pre-neuron $i$ spike reaching post-neuron $j$ after the post neuron spikes which essentially leads to LTD. For our network if the pre-neuron $i$ spikes after $t - d_{i,j}$ then value of $\beta_{i,j,k}^{[t-1]}$ is high and for positive $v_{i,j,k}$(excitatory synapse) then the spiking of post-neuron $j$ at time $t$ is less likely.

For fourth term consider the preneuron to be $j$ and postneuron to be $i$. If $j$ spikes at $t$ then that spike reaches the post neuron $i$ at $t = t + d_{i,j}$. The spikes of post neuron $i$ before time $t$ will lead to LTD.

The DyBM has shown to memorize binary sequences(eg. an image) effectively. Further if the memorized sequence is shown with some anomaly then the model detects that anomaly and its location through a spike in energy.

## Extensions of DyBM

The neurons in DyBM can have only binary values. For using DyBM to learn and predict real valued time series DyBM has been extended to Gaussian DyBM and RNN-Gaussian DyBM.

### 3.1 Gaussian DyBM

Gaussian DyBM is extension of Gaussian Boltzmann Machine as DyBM is of Boltzmann Machine. The probability of neuron j at time $t$ is a gaussian distribution given as

$$p(x_j^{[t]}|\mathbf{x}^{[t-T,t-1]}) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x_j^{[t]}-\mu_j^{[t]})^2}{2\sigma_j^2}} \quad (45)$$

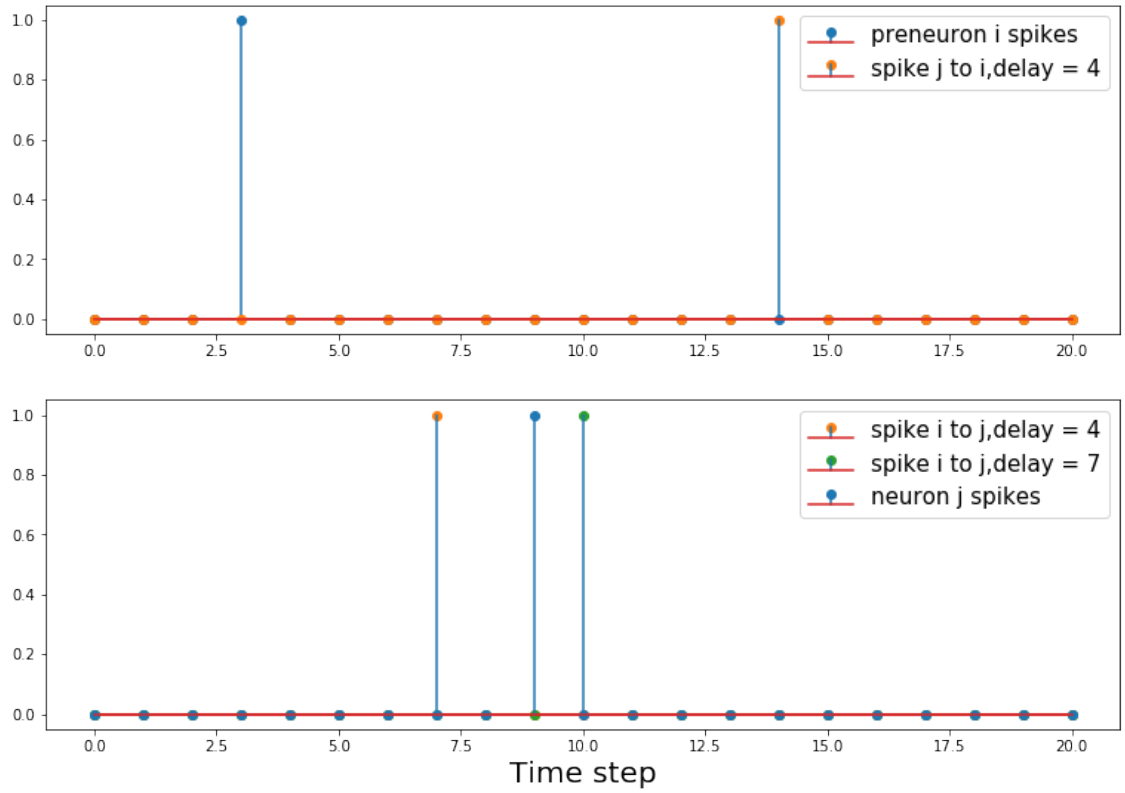
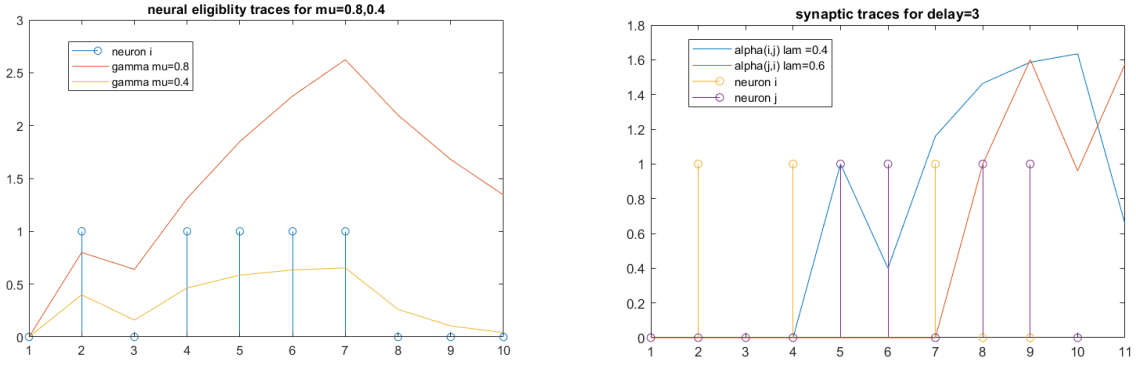



Figure 6: The figure shows the LTP corresponding to 2nd term of energy equation and LTD corresponding to 3rd and 4th term of energy equation. The neuron $i$ spikes at t=3 and reaches neuron $j$ at t=7(delay=4) and t=10(delay=7). Neuron $j$ spikes at t=9 and so LTP happens for synapse(i,j) if delay is 4 and LTD happens if delay=7. Now, the spike from neuron $j$ reaches neuron $i$ at t=14. So the synapse(j,i) will have LTD as neuron $i$ already spiked at t=3




(a) Neural eligibility traces with time

(b) Synaptic eligibilty traces with time

Figure 7

where $\mu_j^{[t]}$ and $\sigma_j^{[t]}$ are given as

$$\mu_j^{[t]} = b_j^{[t]} + \sum_{\delta=1}^{T} \sum_{i=1}^{N} w_{i,j}^{[\delta]} x_i^{[t-\delta]} \tag{46}$$

For T $\rightarrow \infty$ case to avoid exploding parameters we impose following restrictions on weights for $\delta \geq d_{i,j}$ which are similar to DyBM case.

$$w_{i,j}^{[\delta]} = \sum_{k=1}^{K} \lambda^{\delta - d_{i,j}} u_{i,j,k} \tag{47}$$

Unlike DyBM there are no restrictions on weight for $0 \leq \delta < d_{i,j}$. The advantage for not allowing this restriction is that we can model the DyBM as extension of Vector Auto-Regressive models which widely used in prediction and estimation. By substituting [47] we can rewrite [46] as

$$\mu_j^{[t]} = b_j^{[t]} + \sum_{\delta=1}^{\delta - d_{i,j}} \sum_{i=1}^{N} w_{i,j}^{[\delta]} x_i^{[t-\delta]} + \sum_{i=1}^{N} \sum_{k=1}^{K} \alpha_{i,j,k}^{[t-1]} u_{i,j,k} \tag{48}$$

$$\alpha_{i,j,k}^{[t-1]} = \sum_{\delta=d_{i,j}}^{\infty} \lambda^{\delta - d_{i,j}} x_i^{[t-\delta]} \tag{49}$$

As the probability distribution is gaussian now the neuron can take any real value and hence we have extended the DyBM to include real valued data.

### 3.2 RNN-Gaussian DyBM

The Gaussian DyBM is still a higly linear model and hence cannot predict nonlinearities in time series. We now formulate the RNN-Gaussian DyBM, as a nonlinear extension of the Gaussian DyBM model.

We update the bias parameter vector b, at each time using a Recurrent Neural Net layer. This RNN layer computes a nonlinear feature map of the past time series input to the Gaussian DyBM. The output of RNN layer is connected to bias layer with a weight matrix linearly. These weights along with DyBM can be updated online using any variant of Gradient descent.We consider a Gaussian DyBM connected with a M-dimensional RNN and the input to that RNN is N-dimensional time-series input data vector at time $t-1$. Where in, for most settings M > N. Specifically, for RNN-Gaussian DyBM we consider the bias vector to be time-dependent. The bias vector is updated as:

$$b^{[t]} = b^{[t-1]} + A^T \Phi^{[t]} \tag{50}$$

Here $\Phi^{[t]}$ is the hidden state of RNN which remembers the non-linear features of the input time series. $A$ is the weight matrix projecting M-dimensional RNN output to N-dimensional bias. For training this model we maximize the log-likelihood of input time-series.
The RNN Gaussian DyBM has shown comparable results with a simple LSTM on same time series with 10-13 times better runtime. We include our observations and results on Multi-dimensional noisy sine wave, delta-train and other real world datasets in the results section.

### 3.3 Delay Pruning

Delay Pruning provides a method of training DyBM, and in general neural networks with FIFO queues, with regularization and then choosing the best performing model for improved prediction

on test dataset. Specifically it refers to truncating the FIFO queue lengths to zero, by setting their respective delay values to unit length, for randomly selected axons with a probability p. Even in the presence of a relatively small training and test dataset, Delay Pruning prevents over-fitting to give good generalized performance.

On reduced MNIST moving dataset DyBM with delay pruning shows better results than DyBM with dropouts and basic DyBM.

### 3.4 Functional DyBM

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., spatio-temporal data on a continuous space). The above challenge is addressed by combining a kernel-based function approximation method along with a statistical interpolation method and finally deriving closed-form update rules.

The spiking probability of a neuron is taken as a time dependent Gaussian process. The summation in basic Gaussian DyBM are converted into integrals. It is hard to evaluate this integrals directly in terms of resources and accuracy. The kernel trick( the functional series and all intermediate series required to get to energy term belong to Reproducing kernel Hilbert space) is applied which is widely used in machine learning for mapping finite dimensional data to infinite dimensional data.

There is improvement needed for getting efficient learning algorithms and a proper theoretical study is required for all the properties on Functional DyBM. It still shows significant improvement over Functional Auto-Regressive Models which are extensions of Vector Auto-Regressive models.

## Analysis and Experiments

Using Gaussian DyBM and RNN Gaussian DyBM we analyze synaptic eligibility traces and there variation with different parameters of multidimensional sine wave. We further compare our results on real world dataset of monthy sunspot prediction dataset[] and Pacific exchange rate service[].

### 4.1 Multidimensional Noisy Sine Wave

We train the Gaussian DyBM for 2-dimensional noisy sine wave. The standard deviation of noise $\sigma = 0.05$ and total time steps N=600 are fixed throughout this section. For the DyBM no. of traces K=10, length of FIFO buffer=7 are also fixed throughout this section. Figure 8 shows the results obtained for time period T=200. The input sine waves are 180° phase shifted with respect to each other.
We now analyze the weights $u_{i,j,k}$ corresponding to each trace for T=10,100 and 400. The results are shown in Figure 9. For T=100 case we expect the next prediction to depend on recent past as sine is linear in some regions. From the Figure 9(a) we observe the weights of traces having small $\lambda$ is higher and nearly same. It means that next prediction depends on recent past only. From Fiure 9(b) for T=10 as we can see there is peak near $\lambda$ =0.9. As T=10 is changing very fast the next prediction is dependent only on the immediate past. As the 180° delay directly maps to the inverse of next prediction there is peak at $\lambda$ =0.9. From Figure 9(c) for T=400 we can see the values are nearly flat for all decay rates. As T=400 case will lead the sine wave to be highly linear and the observation follows. The discussion was for self loops(i=j). For synapses connecting

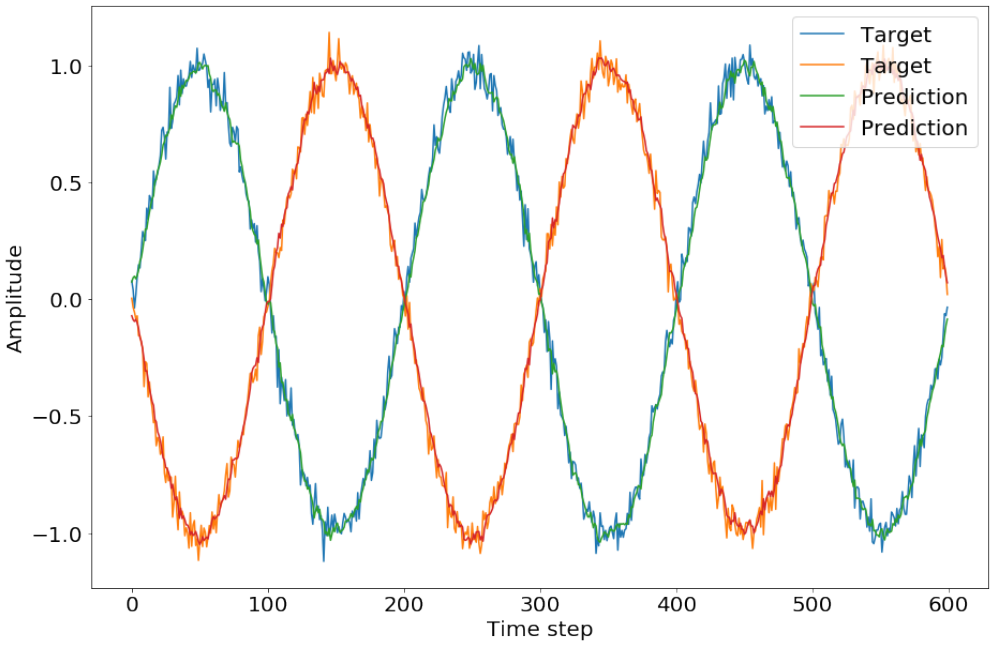
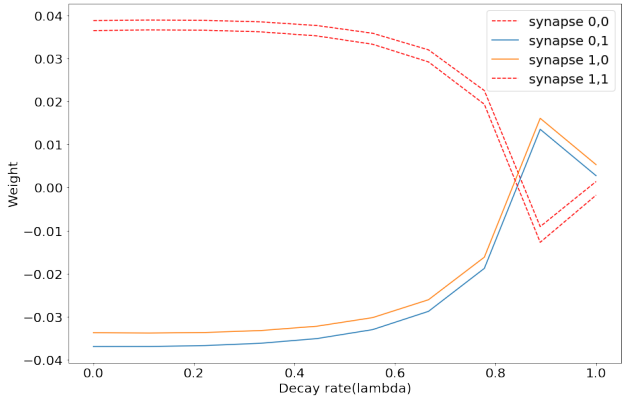



Figure 8: 2-D Noisy Sine Wave: Predicted and Target, T=100, RMSE=0.076

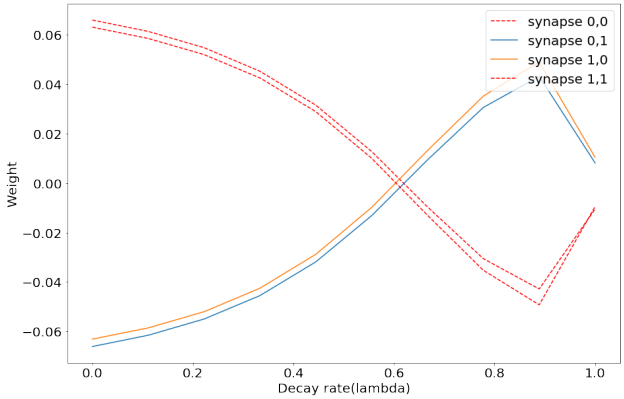
another neuron as the input signals are 180° shifted the weights remain the same just the sign changes.

From equation [48] and [49] we can create a Linear Time Invariant Filter using the weights obtained and the contribution of synaptic traces can be evaluated. Figure 9(d) shows the filter response for T=10,100 and 400. The results are as expected from the previous discussion. For T=100 and 400 we have a decaying response which allows contribution from recent past. For T=10 we observe inverted peak which allows values to be predicted from T/2 delay which has same values with inverted sign.

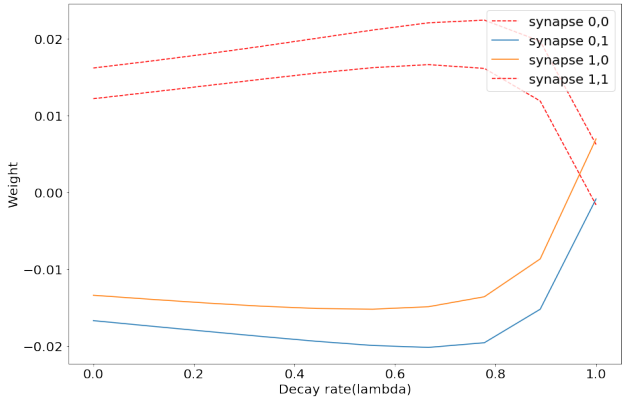
Gaussian DyBM models the values from past using LTI filters and changes the filter response to focus on important parts. The decay rates for each traces control how the filter is formed. So, with significantly less parameters than a normal times series Boltzmann Machine we are able to model and predict a time series.



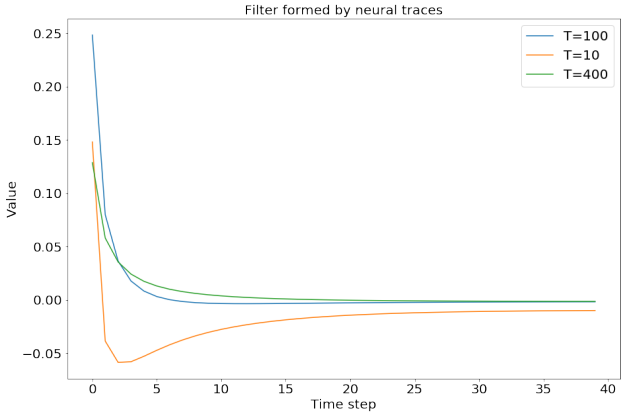

(a) Weights $u_{i,j,k}$ of Synaptic traces, T=100




(b) Weights $u_{i,j,k}$ of Synaptic traces, T=10

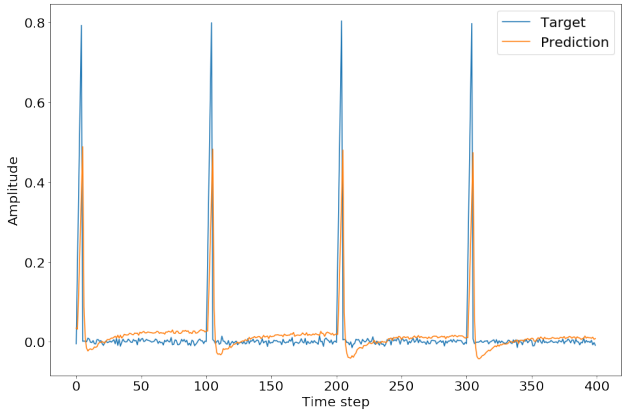



(c) Weights $u_{i,j,k}$ of Synaptic traces, T=400

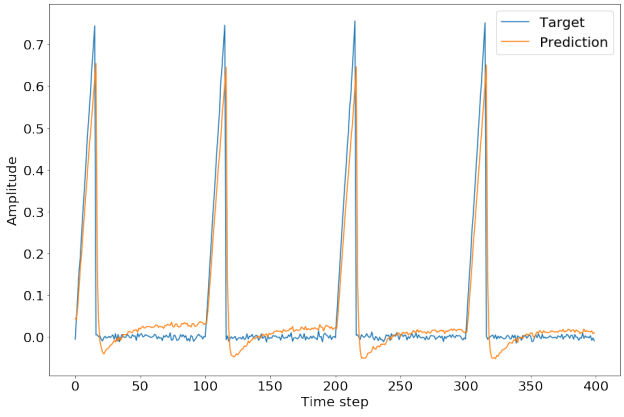



(d) Filter Response

Figure 9




(a) Spike Train Input, Spike Width=5




(b) Spike Train Input, Spike Width=15

Figure 10

## 4.2 Results on Real Datasets

To validate the results on a real life time-series, we have used RNN Gaussian DyBM to predict a variable in the following tasks:

| | Number |
|---|---|
| 0 | 58.0 |
| 1 | 62.6 |
| 2 | 70.0 |
| 3 | 55.7 |
| 4 | 85.0 |

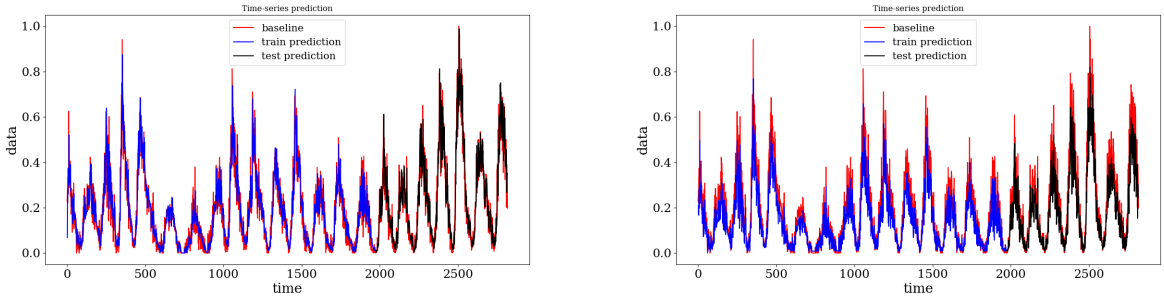
Table 1: A sample of the data from Sunspot number dataset. Number of Observations/datapoints in are 2817.

| yyyy/mm/dd | Wdy | CAD | EUR | JPY | GBP | CHF | AUD | HKD | NZD | KRW | MXN | INR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2007/03/01 | Thu | 1.1713 | 0.75911 | 117.518 | 0.51075 | 1.22278 | 1.2719 | 7.8117 | 1.4400 | 941.56 | 11.1872 | 44.084 |
| 2007/03/02 | Fri | 1.1753 | 0.75865 | 116.841 | 0.51397 | 1.21742 | 1.2765 | 7.8149 | 1.4519 | 943.26 | 11.1827 | 44.118 |
| 2007/03/05 | Mon | 1.1811 | 0.76372 | 116.022 | 0.51992 | 1.22191 | 1.2941 | 7.8152 | 1.4743 | 951.73 | 11.1741 | 44.436 |
| 2007/03/06 | Tue | 1.1760 | 0.76294 | 116.516 | 0.51898 | 1.22424 | 1.2926 | 7.8154 | 1.4694 | 947.62 | 11.1469 | 44.194 |
| 2007/03/07 | Wed | 1.1786 | 0.76063 | 116.428 | 0.51779 | 1.22097 | 1.2880 | 7.8179 | 1.4625 | 948.95 | 11.1610 | 44.192 |

Table 2: A sample of the data from Pacific Exchange rate services dataset. Number of Observations/datapoints in each dimension (here, country's exhange rate with respect to USD)are 3040. Total number of countries analysed is 11.

| Dataset | Variable(s) | Link |
|---|---|---|
| PACIFIC Exchange rate service | Top 10 foreign currencies (Mar, 2007 to Mar, 2019)[1] and INR | http://fx.sauder.ubc.ca/data.html |
| Monthly sunspot number, Zurich, 1749-1983 | Sunspot number | http://www.sidc.be/silso/datafiles-old |

We use LSTM networks as a benchmark to compare the results of DyBM since LSTMs have recently emerged as the state-of-the-art model for time-series prediction. We observe that the DyBM model occasionally outperforms the results of LSTM. We compare the results using $RMSE$ as a metric.

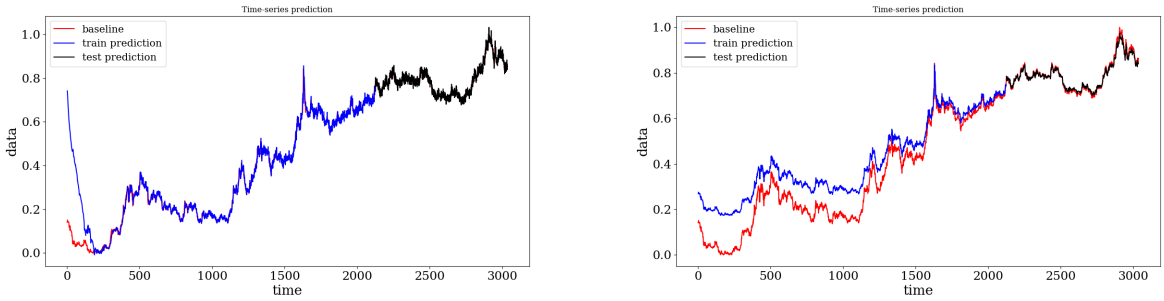



(a) Time-series pred. using RNN-Gaussian DyBM (b) Time-series prediction using LSTM

Figure 11: For the sunspot number dataset, we can clearly see that trained as well as test parts divided in time are closely resembling the actual data. Parameters used: $d = 4,\ \lambda = [0.0, 0.15, 0.3]$.

| | DyBM | LSTM |
|---|---|---|
| Mean train error | 0.07036 | 0.07810 |
| Mean test error | 0.08432 | 0.09655 |
| Per epoch time to learn | 0.80092 | 35.33157 |

Table 3: Results for figure 11

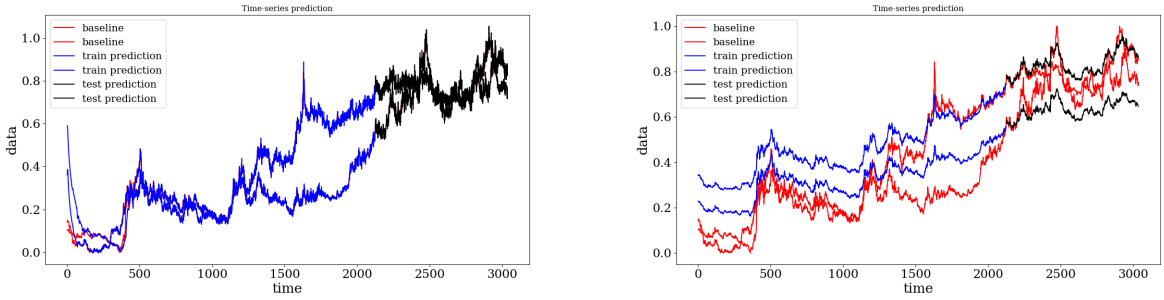



(a) Time-series pred. using RNN-Gaussian DyBM (b) Time-series prediction using LSTM

Figure 12: For the sunspot number dataset, we can clearly see that trained as well as test parts divided in time are closely resembling the actual data. Parameters used: $d = 4,\ \lambda = [0.0, 0.15, 0.3]$.

| | DyBM | LSTM |
|---|---|---|
| Mean train error | 0.08196 | 0.09791 |
| Mean test error | 0.01513 | 0.01082 |
| Per epoch time to learn | 0.80660 | 67.41648 |

Table 4: Results for figure 12

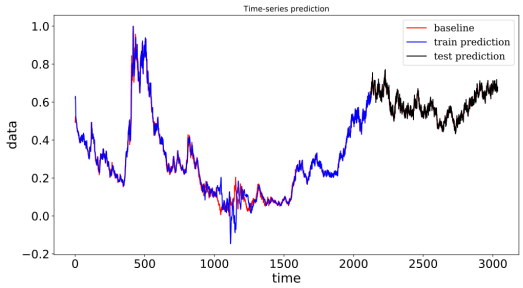



(a) Time-series pred. using RNN-Gaussian DyBM (b) Time-series prediction using LSTM

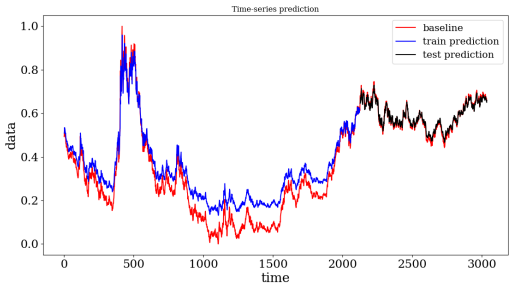
Figure 13: For multi-dimensional time-series, the DyBM completely seems to outperform LSTM. Here, we have 2-D time-series of exhange rates of MXN and INR wrt USD. Parameters used: $d = 4,\ \lambda = [0.0, 0.15, 0.3]$.

| | DyBM | LSTM |
|---|---|---|
| Mean train error | 0.05623 | 0.13958 |
| Mean test error | 0.03628 | 0.09103 |
| Per epoch time to learn | 0.98261 | 111.95985 |

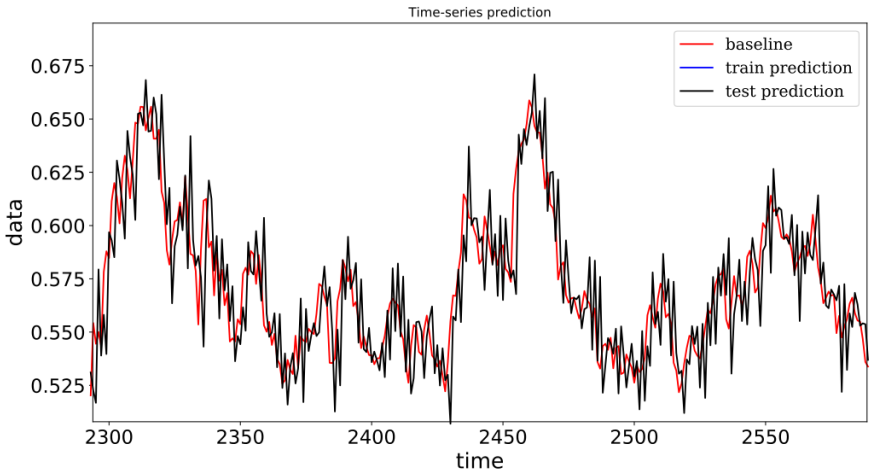
Table 5: Results for figure 13




(a) Time-series pred. using RNN-Gaussian DyBM




(b) Time-series prediction using LSTM




(c) Here, this a zoomed in picture of the testing phase for DyBM. There is a slight jitter seen in the prediction which is because in RNN gaussian, we estimate the $\sigma, \mu$ corresponding to the data and not the exact value of each datapoint. This variance depends on the decay vector used. The more decay size we have, greater is the variance for a data is which is fast-changing.

Figure 14: We then analysed on a single dimensional time-series exchange rate of AUD wrt USD.

### 4.2.1 Conclusion

A couple of very promising distinctions between DyBM and LSTM is the convergence rate and time/epoch in traning. The DyBM learns a time-series very fast as compared to LSTM. Thus the test error is consistently better in case of DyBM. Further, the time/epoch is always 15-25 times better in DyBM as compared to that of LSTM. This is attributed to the fast STDP-like learning rules in DyBM and also the amount of resources in DyBM. In DyBM, we need just those many number of neurons as that equal to the dimension of time-series we are trying to learn.

## References

[1] Osogami, T. Learning binary or real-valued timeseries via spike-timing dependent plasticity. CoRR, abs/1612.04897, 2016. URL: `http://arxiv.org/ abs/1612.04897`.

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

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

[4] Dasgupta, S. and Osogami, T. Nonlinear dynamic Boltzmann machines for time-series prediction. In The 31st AAAI Conference on Artificial Intelligence (AAAI-17), pp. 18331839, January 2017.

[5] 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.

[6] Hinton, G. E. and Brown, A. D. Spiking Boltzmann machines. In Advances in Neural Information Processing Systems 12, pp. 122128. November 1999.

[7] Hiroshi Kajino. A Functional Dynamic Boltzmann Machine. , 2017, Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Pages 1987-1993, URL: `https://doi.org/10.24963/ijcai.2017/276`

[8] Takayuki Osogami, Hiroshi Kajino, Taro Sekiyama. Bidirectional Learning for Time-series Models with Hidden Units. International Conference on Machine Learning PMLR 70:2711-2720, 2017.