Synchronised Interactions in Spiked Neuronal Networks[*]

Erol Gelenbe and Stelios Timotheou
Intelligent Systems and Networks Group
Department of Electrical and Electronic Engineering
Imperial College
London SW7 2BT, UK
{e.gelenbe,stelios.timotheou05}@imperial.ac.uk

Abstract

The study of artificial neural networks has originally been inspired by neurophysiology and cognitive science. It has resulted in a rich and diverse methodology and in numerous applications to machine intelligence, computer vision, pattern recognition and other applications. The random neural network (RNN) is a probabilistic model which was inspired by the spiking behaviour of neurons, and which has an elegant mathematical treatment which provides both its steady-state behaviour and offers efficient learning algorithms for recurrent networks. Second order interactions, where more than one neuron jointly act upon other cells, have been observed in nature; they generalise the binary (excitatory-inhibitory) interaction between pairs of cells and give rise to synchronous firing by many cells. In this paper we develop an extension of the RNN to the case of synchronous interactions which are based on at two cells that jointly excite a third cell; this local behaviour is in fact sufficient to create synchronous firing by large ensembles of cells. We describe the system state and derive its stationary solution as well as a $O(N^3)$ gradient descent learning algorithm for a recurrent network with N cells when both standard excitatory-inhibitory interactions, as well as synchronous firing, are present.

1 Introduction

The spiked random neural network (RNN) model and its gradient based learning algorithm were introduced in [2, 4, 8], and this model was shown to have a "product form" solution, so that the joint stationary probability distribution of the excitation state of all the cells in the network is equal to the product of the stationary marginal probabilities associated with each cell. It was later shown [18, 24] that the RNN is, like other neural network models, a means of approximating continuous and bounded functions. A generalisation of the RNN to represent multiple classes of signals was introduced in [19] and its learning algorithm was applied in [5, 22] to the reproduction and learning of colour image textures, where different classes of signals are a means to distinguish between the types of spikes that the cells in the network can exchange. The application of the RNN to the approximate solution of optimisation problems was discussed in [9, 14], while a similar model was suggested for the analysis of genetic algorithms [15]. An application of the RNN to the study of cortico-thalamic oscillations is discussed in [16]. The RNN has also been applied to the control of quality of service based routing for computer networks [21, 26].

Synchronised firing (SF), where several cells fire simultaneously, and neurons jointly act upon other cells, provide a richer form of inter-cellular interaction than the binary (excitatory-inhibitory) action between

[*]This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project and is jointly funded by a BAE Systems and EPSRC (UK Engineering and Physical Research Council) strategic partnership under Grant No. EP/C548051/1.

pairs of cells. SF has been observed among cultured cortical neurons [1, 6] and it is believed that it serves a prominent role in information processing functions of both sensory and motor systems [11]. Temporal firing synchrony is a result of functional coupling which dynamically varies according to the internal state of the neural system and the stimuli. Theoretical and experimental studies show that SF can occur both in homogeneous and clustered neuronal networks [20], and it was found out that under special population density conditions a neuronal culture can self-organise into linked neuron clusters [23]. These clustered neurons can generate synchronous firing activity similar to the one observed in homogeneous networks [17, 23] and appears to be related to the correlation in connectivity, which is usually measured in neuron cultures because it is difficult to identify the synaptic strength among neurons and hence determine the characteristic node connectivity [25].

In this paper we discuss an extension of the RNN which, in addition to the usual excitatory and inhibitory interactions between cells, also offers the possibility that cells synchronously act together on other cells. In this model certain cells can trigger of successive firing instants in other cells. This network also has “product form”, and we derive its gradient based learning algorithm which is shown to have the same computational complexity $O(N^3)$ for a network with N neurons, as for the recurrent RNN without synchronous firing patterns as shown in [8].

The model developed in this paper exhibits synchronised firing between cells, where one cell may trigger firing in another one. In fact, cascades of such triggered firings can occur in the model that we study. It appears that some the experimental observations of synchronised firing in cultured or sliced neuron cell ensembles are in fact bursts of firing resulting from the non-linear dynamics of the neuronal interactions. Our model describes the triggering of firing between two cells and also allows triggered firing by cascades of cells, and these cascades can also include feedback loops so that lengthy bursts of firing can also be modelled. Thus the present model can to a certain extent be used to mimic the spike bursts which have been experimentally observed.

1.1 The mathematical model

We consider a network with N neurons or cells which have independent and exponentially distributed firing times with firing rates $r_1, \ ... \ , r_N \geq 0$. As with the original RNN model, the internal state or excitation level of a neuron, say neuron i, is represented by the non-negative variable $k_i(t)$. We say that the neuron is excited if $k_i(t) \geq 1$, and that it is quiescent if $k_i(t) = 0$. A neuron may fire only if it is excited. If a neuron fires, then its excitation state will drop by 1: thus if neuron i fires at time t, then $k_i(t^+) = k_i(t) - 1$.

In the original RNN model, neurons interact with each other via excitatory and inhibitory spikes which are sent out from a neuron when it fires; in the present model we will add one more interaction based on two cells getting together to act on a third one (the second order interaction) in excitatory mode. Furthermore neurons can receive excitatory and inhibitory spikes *from* the outside world (e. g. from a sensory input), and a cell may fire and send a spike *to* the outside world. Thus as a whole, the following events can occur at time t:

- A cell i receives an excitatory spike from the outside world; this increases the value of the internal state of the receiving neuron by $+1$. Excitatory spikes arrive to cell i from the outside world according to a Poisson process of rate $\Lambda(i)$.
- A cell i receives an inhibitory spike from the outside world at time t; if $k_i(t) > 0$, this decreases the value of the internal state of the receiving neuron by 1 and has no effect of $k_i(t) = 0$. Inhibitory spikes arrive to cell i from the outside world according to a Poisson process of rate $\lambda(i)$.
- Cell i fires at time t, if $k_i(t) > 0$ and the probability of this event is $r_i \Delta t + o(\Delta t)$ so that the cell is exponentially distributed with parameter r_i. If this happens $k_i(t^+) = k_i(t) - 1$. The resulting

spike will go to cell j as an excitatory spike with probability $p^+(i,j)$ or as an inhibitory spike with probability $p^-(i,j)$, or the spike departs the network going to the outside world with probability $d(i)$, or it creates a *synchronous interaction* together with cell j to affect some third cell m, with probability $Q(i,j,m)$.

- An excitatory or inhibitory spike arriving to a neuron j from another neuron i is treated by j exactly the same way as if such spikes arrive from the outside world.

- On the other hand, if at time t a spike from i reaches m to provoke a second order effect on j then the following happen: of course $k_i(t^+) = k_i(t) - 1$, but also $k_m(t^+) = k_m(t) - 1$ and $k_j(t^+) = k_j(t) + 1$ if $k_m(t) > 0$. However if $k_m(t) = 0$ then the only thing that will occur is that $k_i(t^+) = k_i(t) - 1$, and the firing of i will have no other effect.

Thus synchronous interactions take the form of a *joint excitation* by cells i, m on j, and can only occur if both cell i and m are excited. Note also that:

$$d(i) = 1 - \sum_{j=1}^{N} \left[p^+(i,j) + p^-(i,j) + \sum_{m=1}^{N} Q(i,j,m) \right] \tag{1}$$

1.2 Synchronous firing by many cells

$Q(i,j,m)$ is the probability that when i fires, then if j is excited it will also fire, resulting in an excitatory spike being sent to cell m. This synchronous behaviour between i and j can easily be extended to an arbitrary number of cells. Indeed, we could have a sequence of cells $j_1, \ \dots \ , j_{n+1}, j_{n+2}$ such that $Q(j_i, j_{i+1}, j_{i+2}) = 1$ for $1 \leq i \leq n$. In this case, if cells j_1 and j_2 are excited, then eventually all the cells $j_1, \ \dots \ , j_{n+1}, j_{n+2}$ will fire. Thus the generalised RNN model we have described can be used to model some quite general forms of synchronised firing.

2 Network behaviour in steady-state

Let the state of the network as a whole be denoted by $\underline{k}(t) = [k_1(t), \ k_2(t), \ \dots, \ k_N(t)]$. With the assumptions that have been made about Poisson arrivals, exponential firing times, and with the given probabilities of spikes going from one cell to another, the system state is a continuous time Markov chain. As a consequence, the probability distribution of the system state $\{\underline{k}(t) : t \geq 0\}$ satisfies a set of Chapman-Kolmogorov equations. Let us use the following vectors to denote specific values of the network state, where all of these vectors' values must be non-negative:

$$\underline{k} = [k_1, \ \dots, \ k_N]$$
$$\underline{k}_i^+ = [k_1, \dots, \ k_i + 1, \ \dots, \ k_N]$$
$$\underline{k}_i^- = [k_1, \dots, \ k_i - 1, \ \dots, \ k_N]$$
$$\underline{k}_{ij}^{+-} = [k_1, \dots, \ k_i + 1, \dots, \ k_j - 1, \ \dots, \ k_N]$$
$$\underline{k}_{ij}^{++} = [k_1, \dots, \ k_i + 1, \dots, \ k_j + 1, \ \dots, \ k_N]$$
$$\underline{k}_{ijm}^{++-} = [k_1, \dots, \ k_i + 1, \dots, \ k_j + 1, \dots, \ k_m - 1, \ \dots, \ k_N]$$

If the steady-state distribution $\pi(\underline{k}) = \lim_{t\to\infty} P[\underline{k}(t) = \underline{k}]$ exists, it satisfy the Chapman-Kolmogorov equations given in steady-state:

$$
\begin{aligned}
\pi(\underline{k}) \sum_{i=1}^{N} \left[\Lambda(i) + (\lambda(i) + r_i)\mathbf{1}_{\{k_i>0\}}\right] = \\
\sum_{i=1}^{N} \Big\{ \pi\left(\underline{k}_i^{+}\right) r_i d(i) + \pi\left(\underline{k}_i^{-}\right)\Lambda(i)\mathbf{1}_{\{k_i>0\}} + \pi\left(\underline{k}_i^{+}\right)\lambda(i) \\
+ \sum_{j=1}^{N} \Big[\pi\left(\underline{k}_{ij}^{+-}\right) r_i p^{+}(i,j)\mathbf{1}_{\{k_j>0\}} + \sum_{m=1}^{N} \pi\left(\underline{k}_i^{+}\right) r_i Q(i,j,m)\mathbf{1}_{\{k_j=0\}} \\
+ \pi\left(\underline{k}_{ij}^{++}\right) r_i p^{-}(i,j) + \pi\left(\underline{k}_i^{+}\right) r_i p^{-}(i,j)\mathbf{1}_{\{k_j=0\}} \\
+ \sum_{m=1}^{N} \pi\left(\underline{k}_{ijm}^{++-}\right) r_i Q(i,j,m)\mathbf{1}_{\{k_m>0\}} \Big] \Big\} \qquad (2)
\end{aligned}
$$

The following is an application of a result earlier shown in [7].

Theorem: *Let $\lambda^-(i)$ and $\lambda^+(i)$, $i = 1, ...N$ be given by the following system of equations*

$$\lambda^-(i) = \lambda(i) + \sum_{j=1}^{N} r_j q_j [p^-(j,i) + \sum_{m=1}^{N} Q(j,i,m)] \qquad (3)$$

$$\lambda^+(i) = \sum_{j=1}^{N} r_j q_j p^+(j,i) + \sum_{j=1}^{N}\sum_{m=1}^{N} q_j q_m r_j Q(j,m,i) + \Lambda(i) \qquad (4)$$

where

$$q_i = \lambda^+(i) / \left(r_i + \lambda^-(i)\right) \qquad (5)$$

If a unique non-negative solution $\{\lambda^-(i),\ \lambda^+(i)\}$ exists for the non-linear system of equations (3), (4), (5) such that $q_i < 1\ \forall i$,then:

$$\pi(\underline{k}) = \prod_{i=1}^{N} (1 - q_i)\, q_i^{k_i} \qquad (6)$$

The theorem states that whenever a solution can be found to equations (3), (4), (5) such that all the $q_i < 1$, then the network's steady-state solution has the simple product form (6). The condition $q_i < 1$ can be viewed as a "stability condition" that guarantees that the excitation level of each neuron remains finite with probability one. Note also, that the average excitation level of neuron i in steady-state is then simply $q_i/(1 - q_i)$.

We will now introduce a notation which is similar to the one used in [8], where we replace the firing rates r_i and the probabilities $p^+(i,j)$, $p^-(i,j)$ and $Q(i,j,l)$ by "weights", which in this model represent the *rates at which the cells or neurons interact.* Let:

$$w^+(i,j) = r_i p^+(i,j), \qquad (7)$$

$$w^-(i,j) = r_i p^-(i,j), \qquad (8)$$

and

$$w(i,j,l) = r_i Q(i,j,l) \tag{9}$$

As a result we can write:

$$r_i = \frac{\sum_{j=1}^{N} \left[w^+(i,j) + w^-(i,j) + \sum_{m=1}^{N} w(i,j,m) \right]}{1 - d(i)} \tag{10}$$

and the denominator of q_i can be written as:

$$D(i) = r_i + \sum_{j=1}^{N} q_j [w^-(j,i) + \sum_{m=1}^{N} w(j,i,m)] + \lambda(i) \tag{11}$$

while its numerator becomes

$$N(i) = \sum_{j=1}^{N} q_j w^+(j,i) + \sum_{j=1}^{N}\sum_{m=1}^{N} q_j q_m w(j,m,i) + \Lambda(i) \tag{12}$$

so that $q_i = N(i)/D(i)$. The results summarised in this section will now be used to design an efficient learning algorithm for this network with second order effects.

3 A gradient descent learning algorithm of computational complexity $O(N^3)$

Gradient descent based algorithms in neural networks are used to select the "weights" of the network so that, if the network is presented with a given input $\mathbb{X}$, the network's output is a very good match for a desired output vector $\underline{y}$. In our case, the "input vector" $\mathbb{X}$ will be the vector of the *external excitatory and inhibitory arrival rates* $(\Lambda(1),\ \dots\ ,\Lambda(n))$ and $(\lambda(1),\ \dots\ ,\lambda(N))$. The output vector on the other hand will correspond to the steady-state values of the network, for instance a vector of the values taken by the N probabilities $(q_1,\ \dots\ ,q_N)$ which result from applying the input vector to the network. The gradient descent algorithm is usually provided with a set of input and output vectors which are used to adjust the network weights so that the difference between the given and obtained outputs is minimised over the set of inputs vectors.

Now consider a set of K input-output pairs $(\mathbb{X}, \mathbb{Y})$, with inputs given by $\mathbb{X} = [\mathbb{X}_1, \mathbb{X}_2, ..., \mathbb{X}_K]^T$, and $\mathbb{X}_k = [\underline{\Lambda}_k, \underline{\lambda}_k]^T$.

The vectors $\underline{\Lambda}_k = [\Lambda_k(1), ..., \Lambda_k(N)]$ and $\underline{\lambda}_k = [\lambda_k(1), ..., \lambda_k(N)]$ are obviously the external arrival rates of customers and signals entering the N neurons respectively. The K desired outputs are $\mathbb{Y} = [\underline{y}_1, ..., \underline{y}_K]^T$ with $\underline{y}_k = [y_{k1}, ..., y_{kN}]$ where $y_{ki} \in [0,1]$ are the desired outputs of the i^{th} neuron for the k^{th} training set.

A learning algorithm computes values of the network weights so as to find a local minimum of a cost or error function such as:

$$E = \frac{1}{2}\sum_{k=1}^{K} E_k = \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{N} c_i \left(f_i\left(q_{ik}\right) - y_{ik}\right)^2, \quad c_i \geq 0 \tag{13}$$

where $f_i\left(q_{ik}\right)$ is a differentiable function of q_{ik} associated with neuron i and pattern k. If for some reason we want a particular neuron i not to be considered as an output then we can set $c_i = 0$. Note that more general forms of E_k can be selected without significant changes in the algorithm we will consider, as long as E_k is non-negative, differentiable in all of the q_i, and has at least one minimum for the set of all parameter values $w^+(i,j),\ w^-(i,j) \geq 0$ and $w(i,j,l) \geq 0$.

3.1 Restricting the optimisation to $w(i,j,l) = w^-(i,j)a(j,l)$

In general we can select the $w(i,j,l)$ in an arbitrary manner as long as $w(i,j,l) \geq 0$, and $w^+(i,j),\ w^-(i,j) \geq 0$. However we see that $w(i,j,l)$ in fact acts as an inhibitory term from i to j, followed by an excitatory term from j to l. Thus we propose to simplify the computation involved in seeking a minimum of the cost function (13) by writing:

$$w(i,j,l) = w^-(i,j)a(j,l),\ \ for\ all\ 1 \leq i,j,l \leq N, \tag{14}$$

where $a(j,l) \geq 0$.

We will therefore design a gradient descent algorithm to obtain the unknown parameters of the network i.e. the matrices $\mathbb{W}^+ = \{w^+(i,j)\}, \mathbb{W}^- = \{w^-(i,j)\}$ and $\mathbb{A} = \{a(i,j)\}$ for $i,j = 1,...,N$ in order to minimize the cost function.

In the sequel we will use the generic term $w(u,v)$ to represent either $w(u,v) = w^+(u,v)$ or $w(u,v) = w^-(u,v)$ or $w(u,v) = a(u,v)$. The weights are updated using the gradient descent rule so that for the $k-th$ input-output pair, the $n-th$ computational step is:

$$w_{k,n+1}(u,v) = w_{k,n}(u,v) - \eta \frac{\partial E_k}{\partial w(u,v)}|_{w(u,v)=w_n(u,v),\ \mathbb{X}=\mathbb{X}_k,\ \underline{y}=\underline{y}_k} \tag{15}$$

where n denotes the update step, $\eta > 0$ is known as "the learning rate" and the partial derivative of the cost function on the right hand side is evaluated using the $n-th$ computed values of the weights. The derivative of the cost function is obviously:

$$\frac{\partial E_k}{\partial w(u,v)} = \sum_{i=1}^{N} c_i \left(f_i\left(q_{ik}\right) - y_{ik}\right) \frac{\partial f_i(q_i)}{\partial q_i} \frac{\partial q_i}{\partial w(u,v)} \Bigg|\ w(u,v) = w_{k-1}(u,v),\ q_i = q_{ik-1} \tag{16}$$

Differentiating $q_i = N(i)/D(i)$ with respect to the generic variable $w(u,v)$ yields:

$$\frac{\partial q_i}{\partial w(u,v)} = \frac{\frac{\partial N(i)}{\partial w(u,v)} D(i)}{D^2(i)} - \frac{\frac{\partial D(i)}{\partial w(u,v)} N(i)}{D^2(i)} = \frac{1}{D(i)} \left[\frac{\partial N(i)}{\partial w(u,v)} - \frac{\partial D(i)}{\partial w(u,v)} q_i\right] \tag{17}$$

Next we derive expressions for the derivatives $\frac{\partial N(i)}{\partial w(u,v)}$ and $\frac{\partial D(i)}{\partial w(u,v)}$ for all three parameters $w^+(u,v)$, $w^-(u,v)$ and $a(u,v)$.

$$\begin{aligned}\frac{\partial N(i)}{\partial w^+(u,v)} &= \sum_{j=1}^{N} \frac{\partial q_j}{\partial w^+(u,v)} w^+(j,i) + q_u \mathbf{1}_{\{v=i\}} + \sum_{j=1}^{N} \frac{\partial q_j}{\partial w^+(u,v)} \sum_{m=1}^{N} w^-(j,m)\, q_m a(m,i) \\ &\quad + \sum_{j=1}^{N} q_j \sum_{m=1}^{N} \frac{\partial q_m}{\partial w^+(u,v)} w^-(j,m)\, a(m,i)\end{aligned} \tag{18}$$

$$\begin{aligned}\frac{\partial N(i)}{\partial w^-(u,v)} &= \sum_{j=1}^{N}\frac{\partial q_j}{\partial w^-(u,v)}w^+(j,i) + \sum_{j=1}^{N}\frac{\partial q_j}{\partial w^-(u,v)}\sum_{m=1}^{N}w^-(j,m)\,q_m a(m,i) \\ &+\sum_{j=1}^{N}q_j\sum_{m=1}^{N}\frac{\partial q_m}{\partial w^-(u,v)}w^-(j,m)\,a(m,i) + q_u q_v a(v,i) \end{aligned} \tag{19}$$

$$\begin{aligned}\frac{\partial N(i)}{\partial a(u,v)} &= \sum_{j=1}^{N}\frac{\partial q_j}{\partial a(u,v)}w^+(j,i) + \sum_{j=1}^{N}\frac{\partial q_j}{\partial a(u,v)}\sum_{m=1}^{N}w^-(j,m)\,q_m a(m,i) \\ &+\sum_{j=1}^{N}q_j\sum_{m=1}^{N}\frac{\partial q_m}{\partial a(u,v)}w^-(j,m)\,a(m,i) + \sum_{j=1}^{N}q_j q_u w^-(j,u)\mathbf{1}_{\{v=i\}} \end{aligned} \tag{20}$$

$$\frac{\partial D(i)}{\partial w^+(u,v)} = \frac{\mathbf{1}_{\{u=i\}}}{1-d(i)} + \sum_{j=1}^{N}\frac{\partial q_j}{\partial w^+(u,v)}\left[w^-(j,i) + \sum_{m=1}^{N}w^-(j,i)a(i,m)\right] \tag{21}$$

$$\begin{aligned}\frac{\partial D(i)}{\partial w^-(u,v)} &= \mathbf{1}_{\{u=i\}}\frac{1+\sum_{m=1}^{N}a(v,m)}{1-d(i)} + q_u\mathbf{1}_{\{v=i\}} + q_u\sum_{m=1}^{N}a(v,m)\mathbf{1}_{\{v=i\}} \\ &+\sum_{j=1}^{N}\frac{\partial q_j}{\partial w^-(u,v)}\left[w^-(j,i) + \sum_{m=1}^{N}w^-(j,i)a(i,m)\right] \end{aligned} \tag{22}$$

$$\begin{aligned}\frac{\partial D(i)}{\partial a(u,v)} &= \frac{w^-(i,u)}{1-d(i)} + \sum_{j=1}^{N}q_j w^-(j,u)\mathbf{1}_{\{u=i\}} \\ &+\sum_{j=1}^{N}\frac{\partial q_j}{\partial a(u,v)}\left[w^-(j,i) + \sum_{m=1}^{N}w^-(j,i)a(i,m)\right] \end{aligned} \tag{23}$$

Substituting (18) and (21) into (17) gives:

$$\begin{aligned}\frac{\partial q_i}{\partial w^+(u,v)} &= \frac{1}{D(i)}\sum_{j=1}^{N}\frac{\partial q_j}{\partial w^+(u,v)}\Bigg\{w^+(j,i) + \sum_{m=1}^{N}q_m w^-(j,m)a(m,i) \\ &+a(j,i)\sum_{m=1}^{N}q_m w^-(m,j) - q_i w^-(j,i)\sum_{m=1}^{N}a(i,m) - q_i w^-(j,i)\Bigg\} \\ &+\frac{q_u}{D(i)}\left[\mathbf{1}_{\{v=i\}} - \frac{\mathbf{1}_{\{u=i\}}}{1-d(i)}\right] \end{aligned} \tag{24}$$

To derive the above equation we used the fact that:

$$\begin{aligned}\sum_{j=1}^{N} q_j \sum_{m=1}^{N} \frac{\partial q_m}{\partial w^+(u,v)} w^-(j,m)a(m,i) &= \sum_{m=1}^{N} \frac{\partial q_m}{\partial w^+(u,v)} \sum_{j=1}^{N} q_j w^-(j,m)a(m,i) \\ &= \sum_{j=1}^{N} \frac{\partial q_j}{\partial w^+(u,v)} \sum_{m=1}^{N} q_m w^-(m,j)a(j,i)\end{aligned}$$

Similarly, for $\frac{\partial q_i}{\partial w^-(u,v)}$ and $\frac{\partial q_i}{\partial a(u,v)}$ we obtain:

$$\begin{aligned}\frac{\partial q_i}{\partial w^-(u,v)} &= \frac{1}{D(i)} \sum_{j=1}^{N} \frac{\partial q_j}{\partial w^-(u,v)} \Bigg\{ w^+(j,i) + \sum_{m=1}^{N} q_m w^-(j,m)a(m,i) \\ &\quad + a(j,i) \sum_{m=1}^{N} q_m w^-(m,j) - q_i w^-(j,i) \sum_{m=1}^{N} a(i,m) - q_i w^-(j,i) \Bigg\} \\ &\quad + \frac{1}{D(i)} \left[q_u q_v a(v,i) - q_u \mathbf{1}_{\{u=i\}} \frac{1+\sum_{m=1}^{N} a(v,m)}{1-d(i)} - q_v q_u \mathbf{1}_{\{v=i\}} [1 + \sum_{m=1}^{N} a(v,m)] \right] \end{aligned} \tag{25}$$

$$\begin{aligned}\frac{\partial q_i}{\partial a(u,v)} &= \frac{1}{D(i)} \sum_{j=1}^{N} \frac{\partial q_j}{\partial a(u,v)} \Bigg\{ w^+(j,i) + \sum_{m=1}^{N} q_m w^-(j,m)a(m,i) \\ &\quad + a(j,i) \sum_{m=1}^{N} q_m w^-(m,j) - q_i w^-(j,i) \sum_{m=1}^{N} a(i,m) - q_i w^-(j,i) \Bigg\} \\ &\quad + \frac{1}{D(i)} \left[q_u (\mathbf{1}_{\{v=i\}} - \mathbf{1}_{\{u=i\}}) \sum_{j=1}^{N} q_j w^-(j,u) - \frac{q_i w^-(i,u)}{1-d(i)} \right] \end{aligned} \tag{26}$$

Now using the vector notation

$$\underline{q} = [q_1, q_2, \; ... \; , \; q_N] \tag{27}$$

and defining the $N \times N$ matrix

$$\begin{aligned}\mathbb{W} &= \frac{1}{D(j)} \cdot \Bigg\{ w^+(i,j) - q_j w^-(i,j) + \sum_{m=1}^{N} q_m w^-(i,m)a(m,j) \\ &\quad + a(i,j) \sum_{m=1}^{N} q_m w^-(m,i) - q_j w^-(i,j) \sum_{m=1}^{N} a(j,m) \Bigg\} \quad i,j = 1,2,...,N \end{aligned} \tag{28}$$

then (24) - (26) can be written in vector form as:

$$\frac{\partial \underline{q}}{\partial w^+(u,v)} = \frac{\partial \underline{q}}{\partial w^+(u,v)} \mathbb{W} + \underline{\gamma}^+(u,v) \tag{29}$$

$$\frac{\partial \underline{q}}{\partial w^-(u,v)} = \frac{\partial \underline{q}}{\partial w^-(u,v)} \mathbb{W} + \underline{\gamma}^-(u,v) \tag{30}$$

$$\frac{\partial \underline{q}}{\partial a(u,v)} = \frac{\partial \underline{q}}{\partial a(u,v)}\mathbb{W} + \underline{\gamma}^{*}(u,v) \tag{31}$$

where we have used:

$$\underline{\gamma}^{+}(u,v) = [\gamma_1^{+}(u,v), \gamma_2^{+}(u,v), ..., \gamma_N^{+}(u,v)], \tag{32}$$

$$\underline{\gamma}^{-}(u,v) = [\gamma_1^{-}(u,v), \gamma_2^{-}(u,v), ..., \gamma_N^{-}(u,v)], \tag{33}$$

and

$$\underline{\gamma}^{*}(u,v) = [\gamma_1^{*}(u,v), \gamma_2^{*}(u,v), ..., \gamma_N^{*}(u,v)], \tag{34}$$

which are given by:

$$\gamma_i^{+}(u,v) = \frac{1}{D(i)} \cdot \begin{cases} q_u - q_u/(1-d(i)) & u=i,\ v=i \\ -q_u/(1-d(i)) & u=i,\ v\neq i \\ q_u & u\neq i,\ v=i \\ 0 & u\neq i,\ v\neq i \end{cases} \tag{35}$$

$$\gamma_i^{-}(u,v) = \frac{1}{D(i)} \cdot \begin{cases} q_u q_v[a(v,i) - 1 - \sum_{m=1}^{N} a(v,m)] - q_u[1+\sum_{m=1}^{N} a(v,m)](1-d(i))^{-1} & v=i,\ u=i \\ q_u q_v[a(v,i) - 1 - \sum_{m=1}^{N} a(v,m)] & v=i,\ u\neq i \\ q_u q_v a(v,i) - q_u[1+\sum_{m=1}^{N} a(v,m)](1-d(i))^{-1} & v\neq i,\ u=i \\ q_u q_v a(v,i) & v\neq i,\ u\neq i \end{cases} \tag{36}$$

$$\gamma_i^{*}(u,v) = \frac{1}{D(i)} \cdot \begin{cases} -q_i w^{-}(i,u)(1-d(i))^{-1} & v=i,\ u=i \\ -q_i w^{-}(i,u)(1-d(i))^{-1} + q_u \sum_{j=1}^{N} q_j w^{-}(j,u) & v=i,\ u\neq i \\ -q_i w^{-}(i,u)(1-d(i))^{-1} - q_u \sum_{j=1}^{N} q_j w^{-}(j,u) & v\neq i,\ u=i \\ -q_i w^{-}(i,u)(1-d(i))^{-1} & v\neq i,\ u\neq i \end{cases} \tag{37}$$

Notice that (29) - (31) can also be written as:

$$\frac{\partial \underline{q}}{\partial w^{+}(u,v)} = \underline{\gamma}^{+}(u,v)\,(\mathbb{I} - \mathbb{W})^{-1} \tag{38}$$

$$\frac{\partial \underline{q}}{\partial w^{-}(u,v)} = \underline{\gamma}^{-}(u,v)\,(\mathbb{I} - \mathbb{W})^{-1} \tag{39}$$

$$\frac{\partial \underline{q}}{\partial a(u,v)} = \underline{\gamma}^{*}(u,v)\,(\mathbb{I} - \mathbb{W})^{-1} \tag{40}$$

where $\mathbb{I}$ is the $N \times N$ identity matrix.

3.2 Steps of the algorithm

Let us now summarise the steps for the learning algorithm:

1. Initialize the matrices $\mathbb{W}^{+}$, $\mathbb{W}^{-}$ and $\mathbb{A}$ and choose a value for η. The larger the value of η, the greater the change in the weight update in one step.
2. Set the input values for $\mathbb{X}_k = [\underline{\Lambda}_k, \underline{\lambda}_k]^T$ and $\underline{y}_k$ for a particular k.

3. Solve the system of the N non-linear equations (3)-(5) based on the above values.
4. Solve the three systems of the N linear equations (38), (39) and (40) using the values of q_{ik} obtained. The complexity of solving these systems is $O(N^3)$, because of the matrix inversion operation or $O(mN^2)$ if an m-step relaxation method is adopted.
5. Using the results from Steps 3 and 4, update the matrices $\mathbb{W}^+ = \{w^+(i,j)\}$, $\mathbb{W}^- = \{w^-(i,j)\}$ and $\mathbb{A} = \{a(i,j)\}$ using (15) and (16). If during some update a particular parameter does not satisfy the constraint that they must be non-negative, then either the particular update can be repeated with a smaller value of η or the particular parameter can be set to the closest values within the constraints.

4 Conclusions

Synchronous firing between neuronal cells has been observed in cultured cells, though in some cases this can be in fact burst firing in groups of cells due to intaracting non-linear properties. Since networks with such behaviours can potentially have a significant effect on the functional properties of biological neural networks, this paper studies such systems, proposes a mathematical model based on the spiked random neural network model, and derives a learning algorithm for such systems in the *recurrent* case. The learning algorithm we derive has a computational complexity which is proportional to the cube of the number of cells. This $O(N^3)$ gradient descent learning algorithm for a recurrent network with N cells includes both the standard excitatory-inhibitory interactions, as well as synchronous firing, within the same model. Thus we see that the computational complexity of the learning algorithm in this case is of the same order as for a recurrent network without synchronous interactions. This work opens up the perspective of further modeling studies where we may build more complex neuronal models in which single cells can fire synchronous spike trains, or contain otherwise even more complex interactions between cells or between regions of a very large network.

References

[1] K. Muramoto, K. Kobayashi, S. Nakanishi, Y. Matsuda and Y. Kuroda "Functional Synapse Formation between Cultured Neurons of Rat Cerebral Cortex", *Proc. Japan Academy 1988* (64): 319-322, 1988.

[2] E. Gelenbe "Random neural networks with positive and negative signals and product form solution", *Neural Computation*, 1 (4): 502-510, 1989.

[3] A. Stafylopatis and E. Gelenbe "Synchronisation in parallel processing of finite sequences", *IEE Proceedings-E*, Vol. 138 (5): 329-334, September 1991.

[4] E. Gelenbe and M. Schassberger "Stability of product form G-Networks", *Probability in the Engineering and Informational Sciences*, 6: 271-276, 1992.

[5] V. Atalay and E. Gelenbe "Parallel algorithm for colour texture generation using the random neural network model", *International Journal of Pattern Recognition and Artificial Intelligence*, 6 (2 & 3): 437-446, 1992.

[6] H. P. Robinson, M. Kawahara, Y. Jimbo, K. Torimitsu, Y. Kuroda and A. Kawana "Periodic synchronized bursting and intracellular calcium transients elicited by low magnesium in cultured cortical neurons", *J. Neurophysiology*, 70 (4): 1606-1616, 1993.

[7] E. Gelenbe "G-networks with triggered customer movement", *Journal of Applied Probability*, 30 (3): 742-748, 1993.

[8] E. Gelenbe "Learning in the recurrent random network", *Neural Computation*, 5: 154-164, 1993.

[9] E. Gelenbe, V. Koubi and F. Pekergin "Dynamical random neural approach to the traveling salesman problem," *ELEKTRIK*, 2(2): 1-10, 1994.

[10] E. Gelenbe "G-networks: An unifying model for queueing networks and neural networks," *Annals of Operations Research*, 48 (1–4): 433-461, 1994.

[11] P. Konig and A. K. Engel "Correlated firing in sensory-motor systems", *Current Opinion in Neurobiology*, 5 (4): 511-519, 1995.

[12] J. M. Fourneau, E. Gelenbe, R. Suros "G-networks with multiple classes of positive and negative customers," *Theoretical Computer Science*, 155: 141-156, 1996.

[13] E. Gelenbe, T. Feng and K. R. R. Krishnan "Neural network methods for volumetric magnetic resonance imaging of the human brain," *Proceedings of the IEEE*, 84 (1): 1488–1496, October 1996.

[14] E. Gelenbe, A. Ghanwani and V. Srinivasan "Improved neural heuristics for multicast routing", *IEEE Journal of Selected Areas of Communications*, 15 (2): 147-155, February 1997.

[15] E. Gelenbe "Genetic algorithms with analytical solution", *Robotics and Autonomous Systems*, 22: 59-64, 1997.

[16] E. Gelenbe and C. Cramer "Oscillatory corticothalamic response to somatosensory input", *Biosystems*, 48 (1-3): 67-75, 1998.

[17] P. Mann-Metzer and Y. Yarom "Electrotonic coupling interacts with intrinsic properties to generate synchronized activity in cerebellar networks of inhibitory interneurons", *J. Neuroscience*, 19 (9): 3298-3306, 1999.

[18] E. Gelenbe and Z. -H. Mao, Y. -D. Li "Function approximation with spiked random networks", *IEEE Trans. on Neural Networks*, 10 (1): 3–9, 1999.

[19] E. Gelenbe and J. M. Fourneau "Random neural networks with multiple classes of signals," *Neural Computation*, 11 (4):953–963, 1999.

[20] R. Segev, Y. Shapira, M. Benveniste, and E. Ben-Jacob "Observations and modeling of synchronized burst in in two-dimensional neural networks", *Physical Review E*, 64 (1): 011920-1–011920-9, 2001.

[21] E. Gelenbe, R. Lent and Z. Xu, "Measurement and performance of a cognitive packet network", *Computer Networks*, 37: 691-791, 2001.

[22] E. Gelenbe and K. Hussain "Learning in the multiple class random neural network", *IEEE Trans. on Neural Networks* 13 (6): 1257–1267, 2002.

[23] R. Segev, M. Benveniste, Y. Shapira and E. Ben-Jacob "Formation of electrically active clusterized neural networks", *Physical Review Letters*, 90 (16): 168101-1–168101-4, 2003.

[24] E. Gelenbe, Z. -H. Mao and Yan-Da Li " Function approximation by random neural networks with a bounded number of layers", *J. Differential Equations and Dynamical Systems*, 12 (1&2): 143–170, 2004.

[25] L. C. Jia, M. Sano, P. -Y. Lai and C. K. Chan "Connectivities and synchronous firing in cortical neuronal networks" *Physical Review Letters*, 93 (8): 088101-1–088101-4, 2004.

[26] E. Gelenbe, R. Lent and A. Nunez "Self-Aware networks and QoS", *Proceedings of the IEEE*, 92 (9): 1478–1489, 2004.