A BIOLOGICALLY BASED SPATIO-TEMPORAL FRAMEWORK FOR THE MATCHING AND ENCODING OF DATA

A Thesis

by

MICAELA AMANDA LANDIVAR

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	David C. Hyland
Committee Members,	Srinivas R. Vadali
	Harry A. Hogan
Head of Department,	Rodney Bowersox

August 2015

Major Subject: Aerospace Engineering

ABSTRACT

This thesis presents a neuron model and framework for the architecture and interaction of neurons in order to accomplish two tasks, 1) data matching, and 2) the storage and retrieval of information. The tasks are approached from the basis of biologically inspired spiking neural network theory. The fundamental aspects of this model are extracted and implemented in conjunction with the designed framework, resulting in a model that takes advantage of the spatio-temporal nature of neurons to match, store, and retrieve data. The driving features are the rest, or refractory, period of the neurons, and the finite, positively sloped post-synaptic responses. When superposed these responses may increase a neuron's potential past the threshold, causing the neuron to fire. A framework, the Competitive Classifying Unit, composed of groups of dynamic threshold neurons is used to match binary strings, with and without noise present. In the absence of noise, results show an increase in accuracy with decreasing standard deviation in the randomness of the neuron threshold. With noise present, the framework retains its ability to identify the specified sequence.

To realize the second task, an additional architectural structure for storing and retrieving data based on the spike time arrival is presented. Training pulse arrival times in conjunction with firings caused by upstream neurons result in synapse weight adjustments. Ultimately, the data is storable and retrievable due to the synapse connections developed between neurons in a network, synapse connections that are either strengthened or pared away during training. Due to the precise timing

requirements of system, a clock is required to measure the passage of time. This necessity which implies periodicity, synchronicity is supported by the number of upstream neuron firings required to cause a downstream neuron to fire, and all of this is supported by homeostasis constraints. Finally, the limits for data storage capacity (i.e.: the number and length of binary strings) is determined based on number of neurons in a neuron cluster.

DEDICATION

This work is dedicated to my God, my Dad, who has always been there for me and believed in me even when I didn't. This is a beginning and I hope to do better with each task He gives me.

ACKNOWLEDGEMENTS

I thank my advisor, Dr. Hyland for patiently teaching and explaining, for his contagious energy, for the joy he takes in tackling a problem, and for his support during my course work. I thank my committee members, Dr. Vadali, and Dr. Hogan, for their support and effort in helping me complete this work.

Thanks also go to my friends and colleagues and the aerospace department faculty and staff for their guidance and support, particularly, Dr. Bowersox, Karen Knabe, and Dr. Haisler.

And I thank my family and friends for being kind, supportive, and all around awesome.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

1. INTRODUCTION AND LITERATURE REVIEW

The human brain is an astounding organ. It is capable of storing approximately 2.5 petabytes according to current estimates, regulating involuntary body functions, processing, analyzing, and storing sensory stimuli, and gauging appropriate responses to the stimuli, to just scratch the surface of its ability. And that is exactly what humans have done in our study of its functionality: simply scratched the surface. Despite the copious amounts of experimentation and simulation that have been done beginning in earnest in the late 1800s and taking off in the mid-twentieth century, there is still much that is not understood [1]. This is unfortunate because the full appropriation of the brain's abilities would open up entirely new horizons for data storage, analysis and data processing, artificial intelligence, etc. Only taking into account the brain's ability to assimilate data from an information rich environment, encode (or compress) the data into "brain language," then decode the data, exemplifies that there is potential for more efficient methods of data handling. In addition, the brain's rapid data correlation ability could be applied to noise removal techniques in image processing, to the autonomous control of vehicles and obstacle avoidance, and to object identification, data mining and fast decision-making.

The problem of understanding brain functionality—for the purpose of utilizing its capabilities—has been attacked in various ways and in multiple disciplines [2]-[4]. The field has been aided not only by neurobiology but psychology. Psychologist's discoveries about cognitive development, "learning, forgetting, and recognition" have

contributed valuable pieces of information to determine how the brain performs [1]. In this process of discovery, the modelers come from two distinct categories which [1] designate as biologically focused and system focused, indicating the importance placed on the biological accuracy of the model as opposed to the functional capability of the model to learn.

As neurons are the basic unit of the brain, a great deal of research has gone into determining how they work and a plethora of neuron models have been developed to describe their behavior and also to determine effective connective architectures for the neurons. The most widely known and widespread neuron models, until recently, have been spatially dependent models based on the average firing rate of neurons [5]. The philosophical reasoning behind rate based models held that all the necessary information passed from downstream neuron to upstream neuron(s) was found in the average firing rate of that neuron. Hence, the information to be stored is solely spatially dependent. (There is no dependence on exactly when a neuron fires with respect to its neighbors.) While technology based on average firing rate models exists and is utilized, the great expectations of the creators and proponents were not met [6], [1].

After a lull, a recent renewal of interest in the study of neurons and neural networks came about due to the idea that input data is actually encoded in the spatiotemporal pattern of the spikes emitted by a given neuron as opposed to the rate. These new models are called spiking or pulsed neural network models (SNN or PNN), and they range from extremely detailed—capturing the changes in the individual ions

that determine the neuron potential—to functional—describing only the "high level" events of neuronal activity.

This research presents a framework to solve correlation (or matching) problems using a competitive neuron firing structure and a framework for learning or storing patterns. While attempting to devise a mechanism capable of mimicking the recognition capabilities of the brain, I began with the investigation of models within the "second generation" models of artificial neural networks. These are models based on the average firing rate of artificial neurons. Second generation models are coupled with some form of training, or error minimization, algorithm the most well-known of which is backpropagation.

The human brain is capable of perceiving objects (i.e.: sound, images) in the surrounding environment and recalling similar or matching stimuli extremely quickly. However, these response rates are not possible in second generation models because the models require the passage of time over which the number of incoming pulses will be averaged. In addition to the above obstacle, there are the pitfalls of backpropagation implementation as well as the implementation of other training algorithms (i.e.: offline error minimization) and the uncertainty of using gradient descent itself.

Early in this study, I attempted to replicate the recognition capability of the visual portion of the brain under such transformations as rotation, translation, and dilation, using gradient descent for error minimization to solve the problem of matching basic shapes (i.e.: squares, circles, triangles). Results showed variable accuracy for this

approach. The process is extremely sensitive to the speed of descent. Local minimums and maximums were often selected instead of global values.

Thus, the search carried on to an investigation of spiking neural networks, a neural network model that incorporates temporal information, not just spatially transmitted information, into the working models of brain activity. These models which store information in temporal patterns, or firing frequencies, are more in line with experimental data than second generation models as they can accommodate the higher reaction rates observed in humans. In selecting this model as a launching pad, I then continued to determine the major feature of this model: the finite slope of the post-synaptic pulse. These topics are covered in the first section.

Taking advantage of the finite nature of the postsynaptic pulse led to the construction of a competitive setup of neurons, the Competitive Classifying Unit (CCU), capable of matching a stimulus to a stored "memory pattern." The base unit of the CCU is the Shift Block, composed of two competing teams of neurons that detect the correlation between an input and ideal pattern. The Associator is composed of Shift Blocks, where the various blocks race against time to determine which incremental shift produces the greatest correlation between the ideal pattern and the input. Several Associator Units are interconnected to form the CCU. For simplicity, the CCU is presented here as operating on one dimensional binary sequences. A performance analysis of the CCU's functionality was carried out by finding an ideal, "memory pattern," within a given stimulus. The experiment was performed with and without noise

present. This subject is covered in the second section and the experiments are presented in section three.

In order for temporal calculations to be carried out, a way to measure the passage of time, a "clock," is necessary. This led to an investigation of homeostasis. Homeostasis refers to the neuron's natural drive toward a stable state with respect to the limited chemicals used in signal transmissions. It is shown that homeostasis favors periodic and synchronous firing in interconnected neurons, or neuron clusters. The periodic aspect of neuron interaction, coupled with a specific neuron interconnection pattern, provides the necessary clock. This theory is the subject of section four.

A simulation was performed, implementing a simplified form of the all-to-all connected network with homeostasis constraints, periodicity and synchrony resulting in a neuron cluster with the ability to "learn" simple patterns, repeat, and maintain these patterns in "memory." The learning capability of a simplified, sparse network like that presented in the theory is demonstrated in section five.

While the previous section presented a limited version of a fast learning network where certain connections were "hardwired," the subsequent section provides the algorithm for realizing a spike-time dependent, self-constructing network. The network starts with random connection strengths and builds up the connections as needed to learn the given training sequence. A MatLab simulation demonstrating the networks fast learning, the connections between neurons and the final output is presented. The explanation and step by step operation for the fast learning network are given in section

six. Finally, section seven, the conclusion, summarizes results, contains concluding remarks, and areas for further research.

1.1 Rate Based Artificial Neural Networks

The ability of artificial neural networks to classify inputs has made them a common tool for object recognition, identification of problem areas in patient X-Ray images, classification problems, such as handwriting recognition, even for odor identification, financial forecasting, etc. [6]-[9]. There are other options besides ANNs, such as feature-based tools, which break down image information into local and global features. A local feature is a property of a single pixel or small patch, while global features cover regions of an image. Both are represented by a descriptor vector. The local feature areas are located by a region of interest detector, such as the Harris-Affine invariant region, Harris point based detectors, or the difference of Gaussian detector. Several local features—i.e., color, gradient, intensity, etc.—together form a descriptor. Alternatively, global features take into account large pixel regions or the entire image in order to form the desired representation of the significant image information [10]. The most commonly used region of interest descriptor algorithms are SIFT, SURF, and PCA. Feature-based methods can be used in conjunction with artificial neural networks, as in the stacked generalization neural network of Lin-Cheng, to perform more accurate object recognition [11].

ANNs, on the other hand, can be used with any type of input, not just feature descriptors; however, more input parameters require a larger training set and longer

training and convergence times. The use of feature descriptors is one way to lessen the input space of the ANN, which is highly desirable for the above reasons.

Some approaches to image analysis using ANNs are used in facial recognition techniques and include the use of low resolution images to locate the desired feature's general region within the image, then using that location as input to attempt to match the feature to one in memory [6]. Dividing the image into superpixels or raster scanning before inputting to the ANN are other methods useful for matching an input to a replica stored in a library while simultaneously reducing the computational costs.

The first artificial math model of a single biological neuron using binary threshold functions was developed by McCulloch and Pitts in the 1940s [1]. Over the next thirty years came the integration of the optimal weight adjustment algorithms (learning algorithms), multilayer networks were developed, and new activation functions were implemented to replace the simple binary threshold functions [1].

The evolution of the field led to the realization of the three building blocks of any ANN: the structure of the most basic unit (the artificial neuron), the standard layout of the connections between each of the artificial neurons (the network architecture), and the weight adjustment algorithm (learning algorithms).

1.1.1 Artificial Neuron Structure

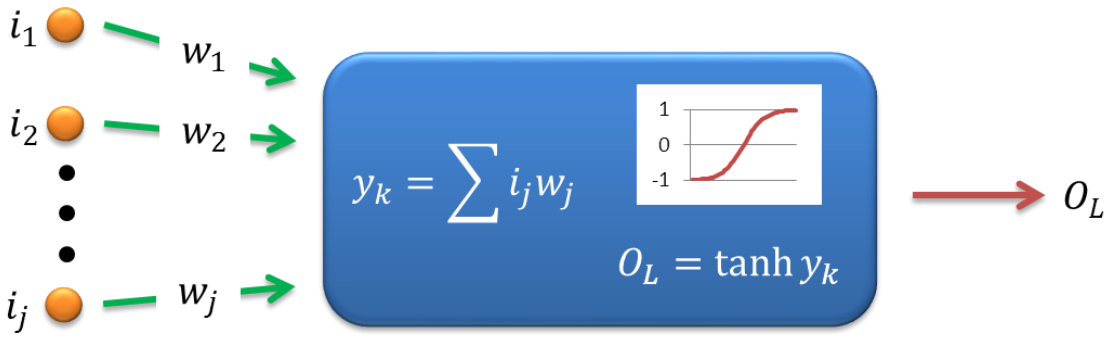

Figure 1.1. Diagram of the artificial neuron structure.

The artificial neuron (AN) structure is composed of the same features in any type of ANN, though there are multiple options for types of features that may be selected. For example, as illustrated in Figure 1.1 above, every AN takes in inputs, i_j. Each input is multiplied by its particular weight coefficient, w_j. Next, in most ANNs, each product is summed to a single value, y_k, within the neuron of interest (NOI). Then, the sum is input to the activation function, the output of which is the AN's output. There are many different types of activation functions from which to choose, but the sigmoid function is a well-known option [12].

1.1.2 Artificial Neural Network Architecture

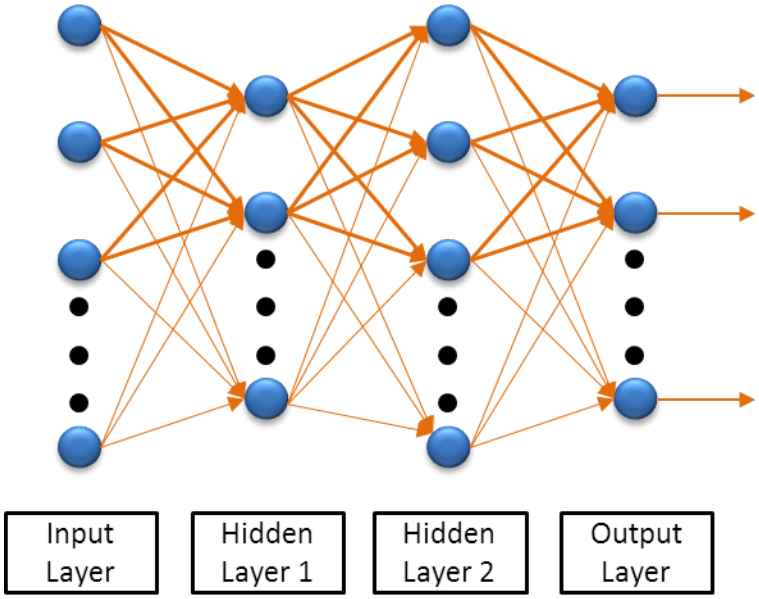

Figure 1.2. Diagram of ANN feedforward architecture.

There are many options for network architecture, but we will only look at the most commonly used architecture: the feedforward network. As the name suggests, information is fed forward from each preceding layer only to each subsequent layer, as shown in Figure 1.2. This is opposed to other architectures, such as fully connected networks, or generic layered networks. Fully connected networks connect each neuron to every other neuron and generic layered networks allow connections that skip the layer immediately following it or have intra-layer connections, both of which do not appear in the feedforward architecture.

1.1.3 Backpropagation Learning Algorithm

Lastly, we will examine the backpropagation learning scheme. There are many other learning algorithms a few of which are additive momentum, self-adaptive learning rate, resilient backpropagation, quasi-Newton, conjugate gradient backpropagation, Bayesian regulation backpropagation, etc. [13]. The most commonly used learning algorithm is backpropagation [14]. As mentioned previously, backpropagation makes use of the gradient descent: weights are adjusted such that the error between the "correct" output, c, and the network's calculated output, o, is minimized [15]. Different error functions may be used, such as a simple vector difference or a norm-based function of the difference, such as the mean squared error,

$$J = \frac{1}{2}(c - o)^2. \tag{1.1}$$

As the error is a function of the network output and the output is a function of the weights, the change in weights required to drive the error to a minimum can be calculated as a function of the "correct" output, the actual output, and the derivative of the activation function [15]. The change in weights for each preceding layer can then be calculated or "back propagated" to the previous layer, resulting in the first change in weight equation of the form

$$\Delta w = \alpha(c - o)f'(o) \tag{1.2}$$

where α is a proportionality constant, often referred to as the "learning rate" and $f'(o)$ is the derivative of the activation function as a function of the network output. The change

in weights is used to adjust the weights. This process is iterated until the weights converge, indicating the network has learned the "correct" value [15].

1.1.4 Training

Once the neurons have been defined, the connection, architecture type, and learning algorithm chosen, the network is iteratively trained in such a way as to provide for the desired amount of flexibility in its functionality. Overtraining results in a network with little capacity for generalization—its answer space will only contain exact or near exact matches of the training inputs. Conversely, the answer space for an undertrained network will contain many spurious results. Hence, the network must be trained to a point within a range between undertraining and overtraining [16].

1.2 Correlation of Simple Shapes Using Gradient Descent for Error Minimization

The theoretical basis for the following assessment of gradient descent and ultimately the backpropagation algorithm is the idea that neurons can be thought of as applying transformation equations to the data they receive. For example, let us take a picture of a simple shape—a rectangle—and divide that surface into small segments representing the areas over which a neuron will operate. (Each rectangle represents the portion of the image operated on by a single neuron.)

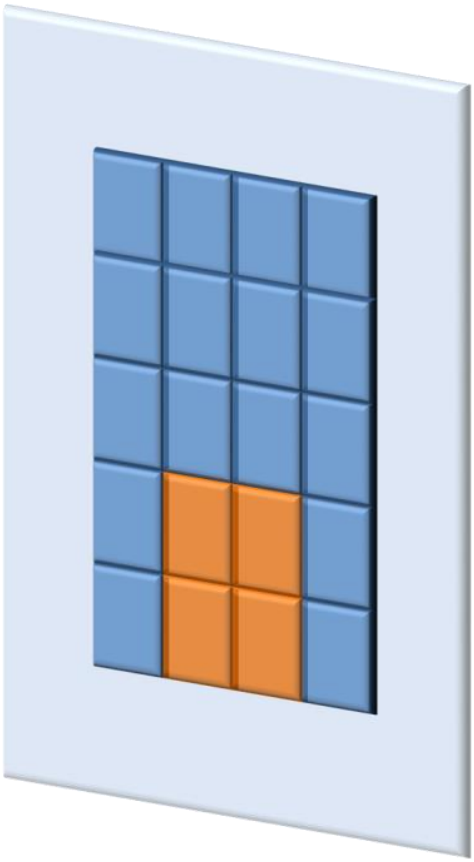

Figure 1.3. Depiction of an image divided into operable segments for neuron operations.

By forming specific connections (synapses) between neurons, we can perform transformations on an image. Firstly, we can represent the same image, meaning no real transformation has occurred as in Figure 1.4.

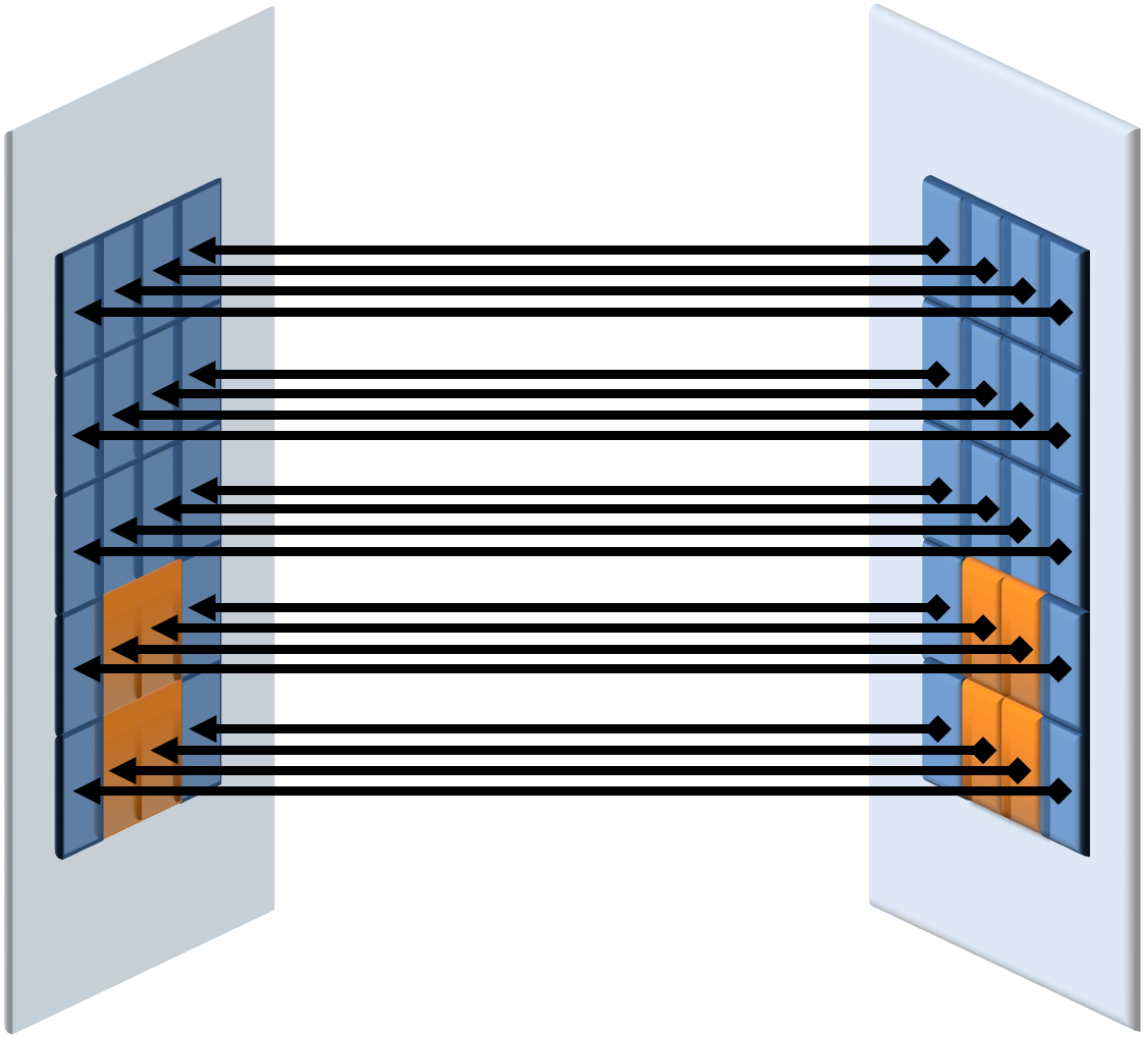

Figure 1.4. Depiction of synapse mappings for an identity transformation.

Neural networks allow for the maintenance of existing synapses while adding more synapses that can perform different transformations, as in Figure 1.5.

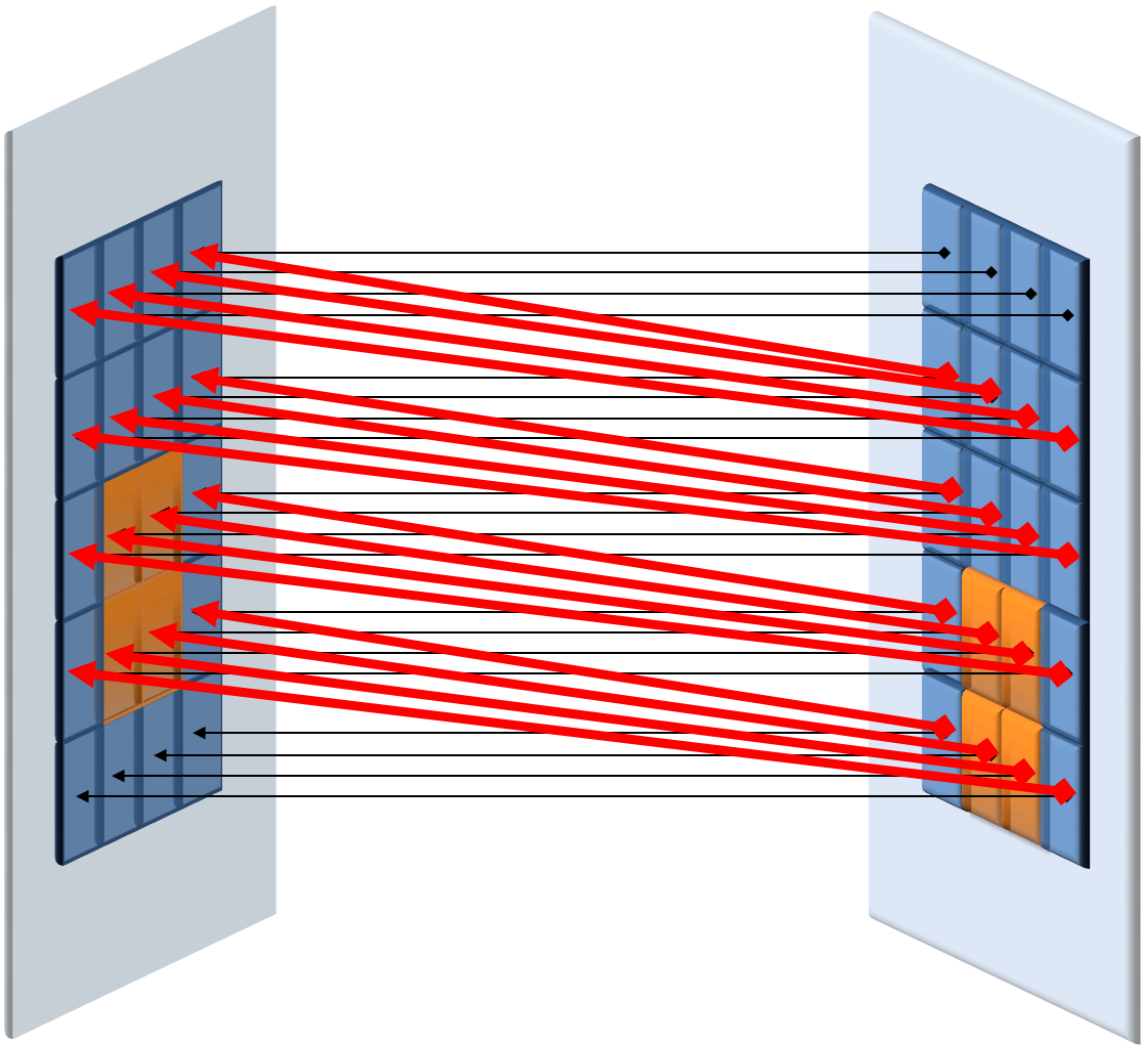

Figure 1.5. Depiction of synapse mappings for a +1 translational transformation.

In the above figure, the previous connections exist but the active synapses transform the original rectangle (right) one segment up (left). In the next figure, the same concept is portrayed but for a +3 translation.

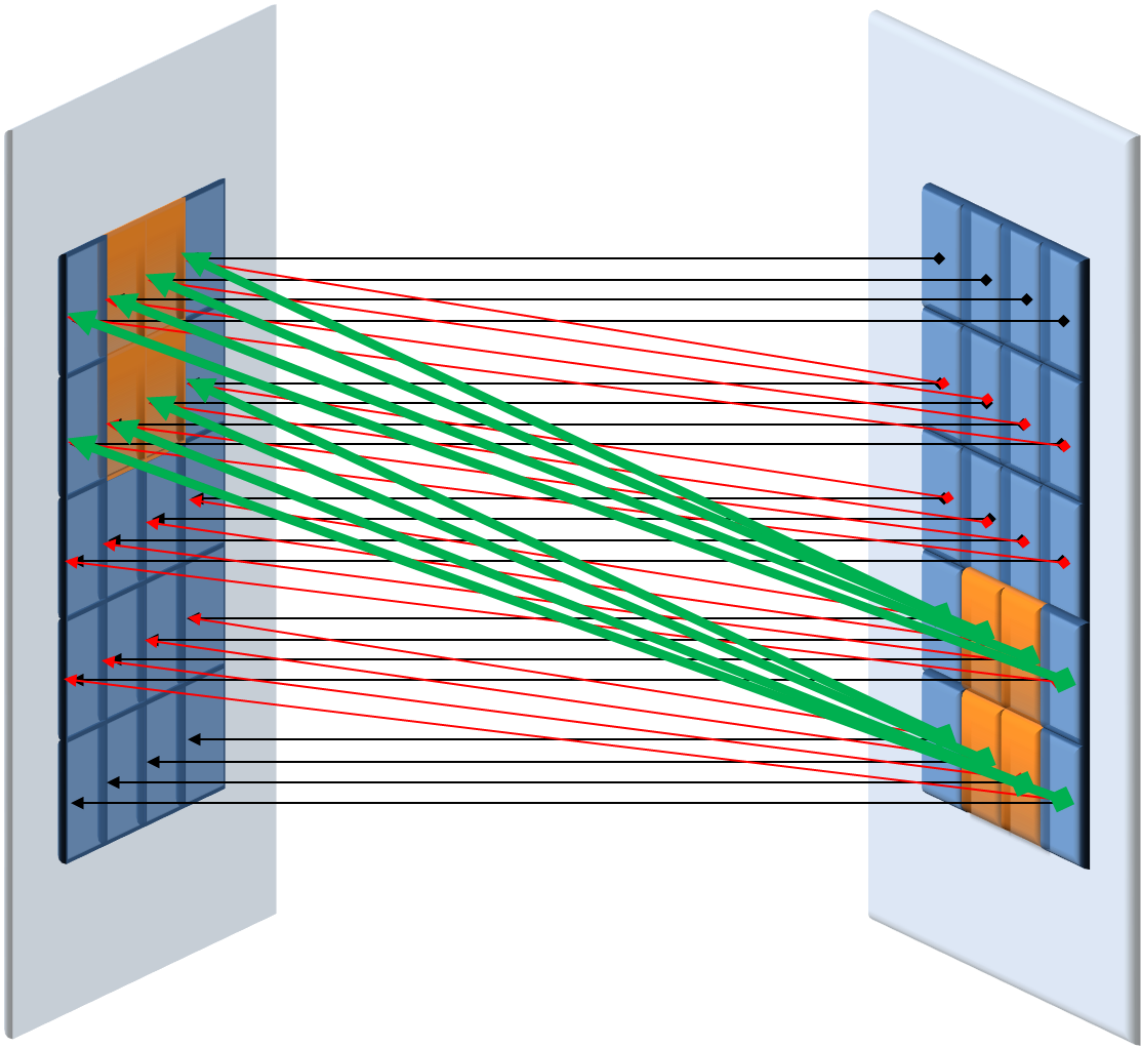

Figure 1.6. Depiction of synapse mappings for +3 translational transformation.

What follows is a mathematical interpretation of the above conceptual model. In order to test the effectiveness of the gradient descent algorithm—ultimately with the intent to test the backpropagation algorithm itself—an experiment was devised to incrementally adjust the parameters governing the translation, rotation, and dilation of simple, perceived 2D shapes such that the difference between the input perceived shape

and an "ideal" shape stored in memory was minimized. In other words, the gradient descent was used to find the parameter values that caused two shapes to be matched.

What follows is the theory formulated to set up the experiment, succeeded by the results of a program implemented to test the theory. Coordinate frames were set to describe the translation, rotation, and dilation leading to the 2D equation specifying the mapping from the original shape to the modified shape. Finally, the parameters—for the x and y coordinates—were adjusted such that the 2D mean squared error function was minimized.

1.2.1 2D Mapping Equation

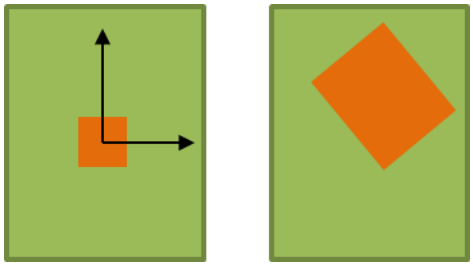

Figure 1.7. A transformed image- Left: an example of a known ideal, I and Right: the scene, C, to be matched.

The algorithm for mapping one shape, or "scene," to another shape, or "ideal," begins with the transformation equations in Table I-1 below. The parameters α and β determine the extent and direction of dilation of the known ideal shape, I, for the x-

coordinates and y-coordinates, respectively. After dilation, the subsequent equations first rotate the x and y values then translate them as shown below, leading to the transformed output, O. The output, O, is then compared with the scene, C, the error is calculated to enable the adjustment of the parameters to minimize the error.

Dilation	$x_1 = \alpha x$	$y_1 = \beta y$
Rotation	$x_2 = x_1 cos\theta + y_1 sin\theta$	$y_2 = y_1 cos\theta - x_1 sin\theta$
Translation	$x_3 = x_2 + d_x$	$y_3 = y_2 + d_y$

Table 1.1. Equations for the dilation, rotation, and translation of the ideal image stored in memory which will be transformed to match the perceived scene.

The synapse mappings from the ideal image to the final image are worked upon by the transformation equations to create the desired change. The figure below demonstrates dilation, rotation, and translation.

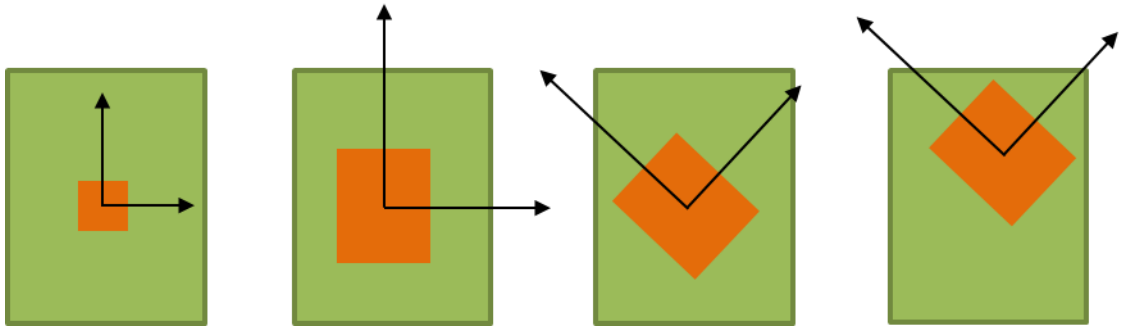

Figure 1.8. Steps to transform and image- Left to right: Ideal, dilation, rotation, and translation of the original ideal.

The equations in Table 1.1 may be written as a matrix function,

$$\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \begin{bmatrix} \alpha\cos(\theta) & \beta\sin(\theta) \\ -\alpha\sin(\theta) & \beta\cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{1.3}$$

Similarly, we can define S as the transformation function used to take the ideal to the output, O, yielding,

$$O = S\big(d_x + \alpha\cos(\theta)\,x + \beta\sin(\theta)\,y, d_y - \alpha\sin(\theta)\,x + \beta\cos(\theta)\,y\big). \tag{1.4}$$

Taking the mean squared error,

$$J = \frac{1}{2}(O - I)^2 \tag{1.5}$$

as the error function to be minimized, where I is the ideal. After taking partial derivatives of the error function with respect to the parameters, and discretizing the results we are left with the evolution of those parameters over time as shown.

$$\frac{d}{dt}\begin{bmatrix} d_x \\ d_y \\ \alpha \\ \beta \\ \theta \end{bmatrix} = -\mu \sum_x \sum_y (O - I) \begin{bmatrix} \dfrac{\partial O}{\partial x_3} \\ \dfrac{\partial O}{\partial y_3} \\ \begin{bmatrix} \dfrac{\partial O}{\partial x_3} & \dfrac{\partial O}{\partial y_3} \end{bmatrix} \begin{bmatrix} \cos(\theta)x \\ -\sin(\theta)x \end{bmatrix} \\ \begin{bmatrix} \dfrac{\partial O}{\partial x_3} & \dfrac{\partial O}{\partial y_3} \end{bmatrix} \begin{bmatrix} \sin(\theta)\,y \\ \cos(\theta)\,y \end{bmatrix} \\ \begin{bmatrix} \dfrac{\partial O}{\partial x_3} & \dfrac{\partial O}{\partial y_3} \end{bmatrix} \begin{bmatrix} y_3 - d_y \\ -x_3 - d_x \end{bmatrix} \end{bmatrix} \tag{1.6}$$

In the above equation, μ adjusts the fineness, or rate, of the increment of the shape characterization parameters.

1.2.2 Implementation

A discrete form of these equations was programmed in MatLab. The first step was to minimize the error between each of the three ideal shapes and the scene, respectively. This was done by incrementing the above transformation parameters until the output converged. To further minimize the error, a composite of the three estimated shapes at the point of convergence were respectively compared to the scene. Error minimization was accomplished by adjusting the μ values controlling the participation coefficients—weights that determine how much each shape estimate contributes to the composite shape. Each participation coefficient's μ value determines the rate of change in the weighting of each estimate shape. The option to add a specified amount of noise to the scene was included in the program.

1.2.3 Results

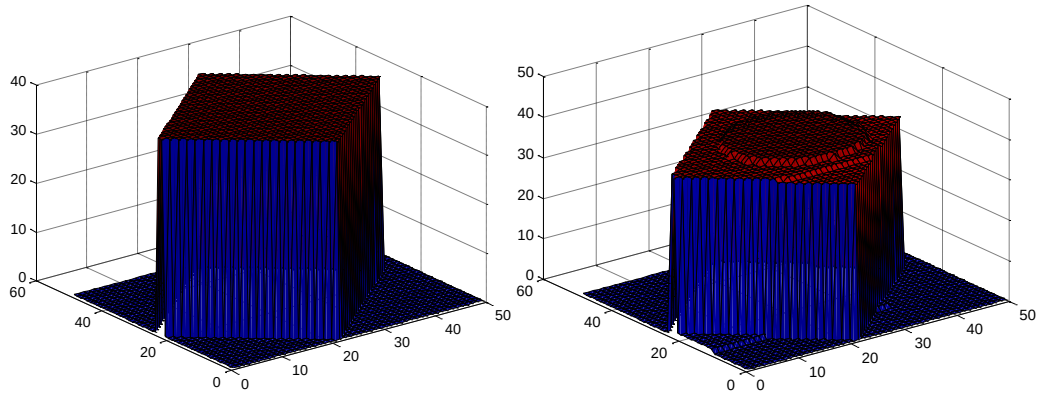

Figure 1.9. Input (Left) and Output Scenes (Right) of the gradient descent method for simple shape matching.
Note: Although these images appear 3D, all computations were performed on 2D shapes. MatLab's 3D plot function generated these 3D images from 2D input to enable detection of miniscule differences not apparent using 2D plot.

Without noise, the error between an input scene and its ideal could be as low as 6%. However, during the experiments, it was found that a distinct set of participation constant μ values was required to avoid a local minima and acquire the global minimum for the error between a given scene and ideal. In other words, minimizing the error between a scene containing a square and an ideal would require discovering a different set of μ values then would be needed to minimize the error between a scene containing a circle and its ideal. Furthermore, these μ values were often mutually exclusive. The conclusion: the abundance of local minima for even simple shapes with noise makes a normal gradient descent method of reaching a global minimum impractical. See the

appendix for the results of additional shapes. These results illustrate how the firing rate model with backpropagation is unsuited for fast, reliable pattern recognition or image processing.

1.3 Spiking Neuron Models

One of the strongest arguments for spiking neural networks as opposed to average firing networks as better neuron models is human reaction time. In experiments, reaction time is too fast to allow for averaging pulses [17]. In an experiment to classify images into two groups, participants were able to complete the task within 400-500 ms—200-300 ms of which are required for the act of pressing a button—leaving 200-300 ms for the act of classification. Additionally, EEG signals demonstrated that the classification was performed in less than 200 ms. These experiments strongly indicate that the mean firing assumption, while it may be useful, is not an accurate description of brain activity. Not only are the neurons a more fitting model, as indicated above, but are a means for "representing time frequency, phase and other features" [18].

Because the spiking neural network model stores information in pulse/spike arrival times as opposed to being dependent purely on the mean rate at which pulses arrive, it is a more accurate model. In SNNs, the spike arrival times affect the downstream neuron's chemical levels, and if a sufficient number of pulses arrive within a given timeframe, the downstream neuron's response is to generate a pulse of its own.

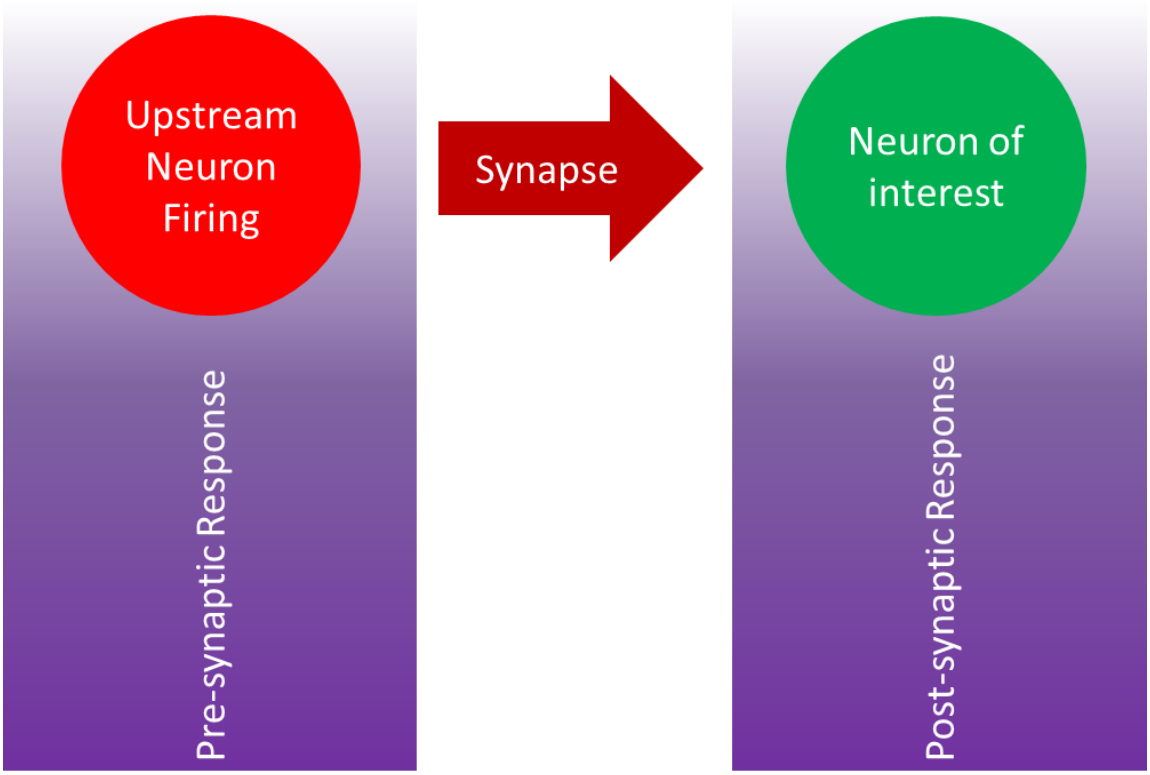

Figure 1.10. Definition of pre-synaptic and post-synaptic response (PSR).

Generally, the "downstream neuron's response" is referred to simply as the post-synaptic response, PSR, and the upstream neuron's response is referred to as the pre-synaptic response. The prevailing SNN theories often describe the PSR as finite sloped—usually an exponential function [19]. For example, in presenting the Spike Response Model, Gerstner and Kistler, [19], choose the PSR to be an exponential function, ε, as shown in the following figure.

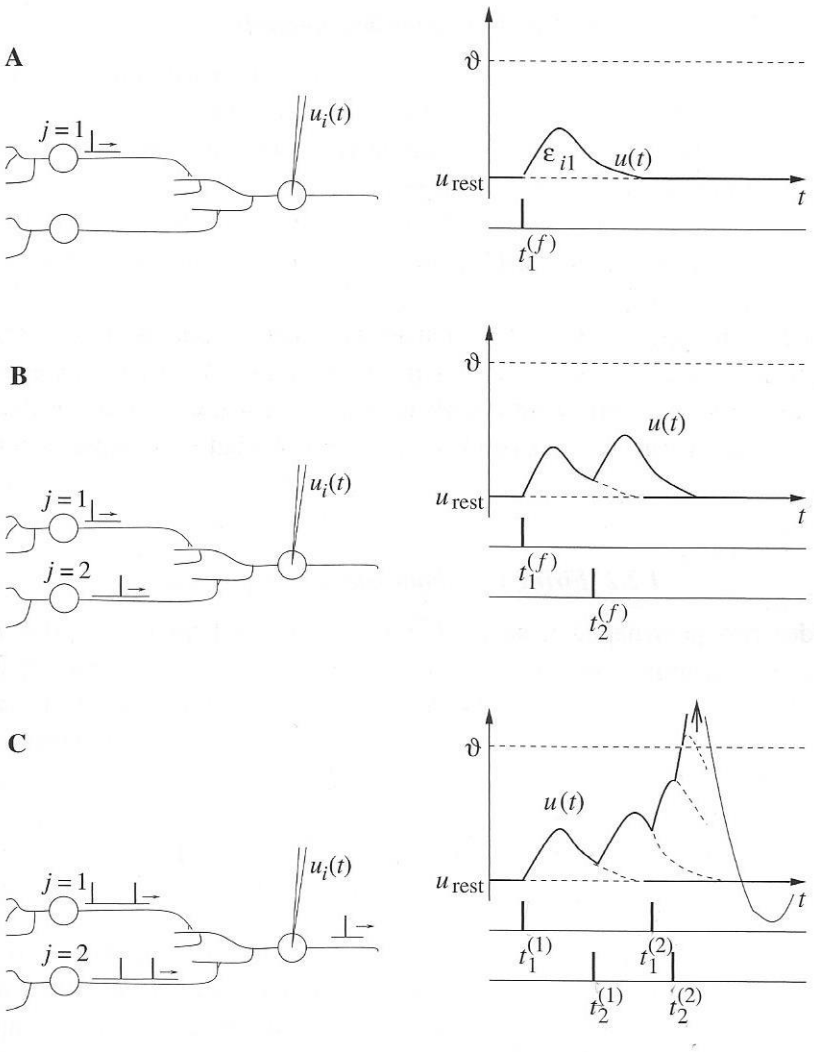

Figure 1.11. Gerstner and Kistler's [19] portrayal of the PSR due to incoming pulses.

In (A), only one of the two "upstream neurons," or pre-synaptic neurons, generates a pulse, resulting in a single corresponding response in the "downstream neuron's" potential, or post-synaptic response of the action potential [19]. In the SRM, the action potentials are mathematically modelled by Dirac deltas. Hence, in (B), both

presynaptic neurons generate APs leading to multiple post-synaptic responses to those arriving APs [19]. As the pre-synaptic neurons fire, the change induced in the post-synaptic neuron potential is modelled by superposing the representative function of the PSP to incoming APs—an exponential function—until the total potential crosses a threshold value, causing the neuron to fire (C) [19]. An equation, again exponential, describing the evolution of the PSR over time is given in Maass and Bishop [20].

Maass and Bishop go on to describe the same kind of PSR for other threshold models such as the Integrate and Fire Model. This model characterizes the ion flow through the neuron membrane as a circuit composed of a resistor in parallel with a capacitor as shown below [21]. Going a step further, Burkitt and Clark [22] stipulate that the slope should have a "max magnitude of order one," but no reason or support is given for this assumption.

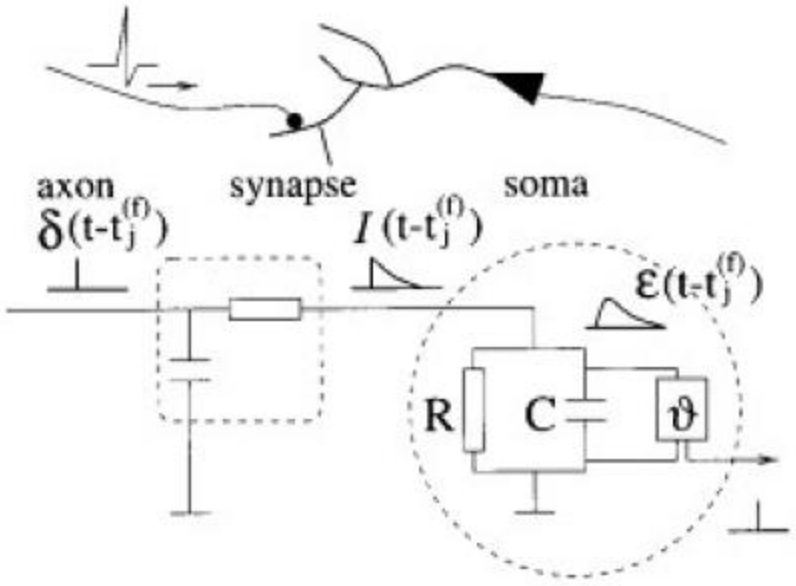

Figure 1.12. Mass and Bishop's portrayal of the integrate and fire model [20].

Finally, Maas and Bishop [20] validate the SRM model and its assumptions, i.e.: finite sloped PSR feature of the model, by showing that it may be used to approximate the Hodgkin Huxley model—the main empirical model that was created based on data from a giant squid axon.

In the each of these cases, no reason is given for the finite slope assumption. After more searching, it's possible to discover that the finite slope of the SRM is based on a generalization of the experimental potential response of a neuron [23], [24].

Given the incredible complexity of neurons, care should be taken to determine which features are crucial for integration into neuron models. A thorough screening of the features incorporated into neuron models will increase our understanding of the neuron itself as well as our ability to explore further with confidence in the firm

foundational understanding of the required assumptions and implications. As the finite slope assumption of these models has not been subjected to a simple thought experiment to test its concurrence with physics or logic, we will do so in the following pages.

1.4 Spike Response Model

Many models describing the biological function of neurons have been formulated [25]. Though the models vary, there are fundamental portions that recur in the various mathematical models. As seen in the Hodgkin-Huxley model [26], ion channel model [19], FitzHugh-Nagumo model [27], Izhikevich model [28], Integrate and Fire model [29], etc., our knowledge of neural networks is complex and detailed, but it remains to be determined which details are crucial to the performance of learning in pattern recognition.

Below, is a very simple model presented by Gerstner and Kistler: the Spike Response Model, SRM_0 [19] and [21]. This section contains a brief explanation of the SRM_0, featuring a finite sloped PSR. Following that is a pulse response equation created to meet the simple needs of the Associator neurons of the Shift Determiner framework, which will be introduced later.

According to the SRM_0, at any given time, t, the following equation describes the voltage potential, $u_i(t)$, of a single neuron, i, with presynaptic neurons, j,

$$u_i(t) = \eta(t - \hat{t}_i) + \sum_j w_{ij} \sum_{t_j^{(f)}} \epsilon_0\left(t - t_j^{(f)}\right) + \int_0^\infty \kappa_0(s) I^{ext} ds \,. \qquad \textbf{(1.7)}$$

The model dictates that neuron i will fire when the following conditions are met

$$u_i(t) = \vartheta \text{ and } \frac{du_i(t)}{dt} > 0. \quad \textbf{(1.8)}$$

When the potential of the neuron has been raised by a sufficient number of presynaptic firings so that it is equal to the constant threshold value, ϑ, and the slope of the potential is increasing, the neuron will fire. This time instant is referred to as $t = t_i^{(f)}$. The latest firing time of a neuron is indicated by $\hat{t}$.

In the voltage potential equation, the kernel η describes the shape of the action potential, or spike, and the refractory period of neuron i. It is dependent on the last firing time of neuron i. ϵ_0 describes the shape of the response of neuron i to pulses generated by upstream neurons and is dependent upon the most recent firing times, $t_j^{(f)}$, of presynaptic neurons, j. κ_0 describes the shape of those presynaptic pulses entering neuron i during the refractory period. It depends on s, where s is defined as $t - t_j^{(f)}$. I^{ext} refers to an externally applied stimulus, or input. Finally, w_{ij} is the coefficient denoting the strength of the connection between neuron i and its presynaptic neurons.

The SRM_0 is a neuron model with the following assumptions distinguishing it from other models: that the entering pulses, are dependent solely on the firing time of the presynaptic neuron and not additionally dependent on the last firing time of neuron i. Below are possible explicit equations for the kernels η, ϵ_0 and κ_0:

$$\eta(t - \hat{t}) = \delta(t - \hat{t}) - \eta_0 \exp\left(-\frac{t - \hat{t}}{\tau_{recov}}\right), \quad \textbf{(1.9)}$$

$$\epsilon_0\left(t - t_j^{(f)}\right) = t \exp\left(-\frac{t - t_j^{(f)}}{\tau_E}\right) \quad \textbf{(1.10)}$$

$$\kappa_0(s) = t \exp\left(-\frac{t - t_j^{(f)}}{\tau_k}\right) \quad \textbf{(1.11)}$$

All above information is modified from Gerstner and Kistler. In Equations 1.10-1.12, ϵ_0 and κ_0 were modified by replacing the constant before the exponential with the variable t. δ is the dirac delta.

While the above kernels make up the SRM_0, the complexity of these equations is not necessary to accomplish the specific goal to create a spike response model to show that a relevant feature of these models is the finite slope of the PSR. This will be demonstrated through the Associator framework which will be presented subsequently.

1.5 Post-Synaptic Response

As previously mentioned, the basic unit of the Competitive Classifying Unit is the Shift Determiner. Associators are composed of blocks of Shift Determiners that employ this competitive structure such that the higher the correlation between an ideal pattern and a stimulus the faster that correlation is identified as the correct match.

The minimum components required to for the above described functionality of Shift Determiner neurons are: 1) an initially finite, nonzero, positively sloped ramp modeling the change due to incoming pulses, and 2) a refractory period, both of which may be modeled by triangular pulses. In particular, the linear behavior of the rising potential is based on the superposition of the triangular pulses representing the spikes, as shown in the following figure. These requirements are the final components that must be explained before the Shift Determiner can be explained in detail.

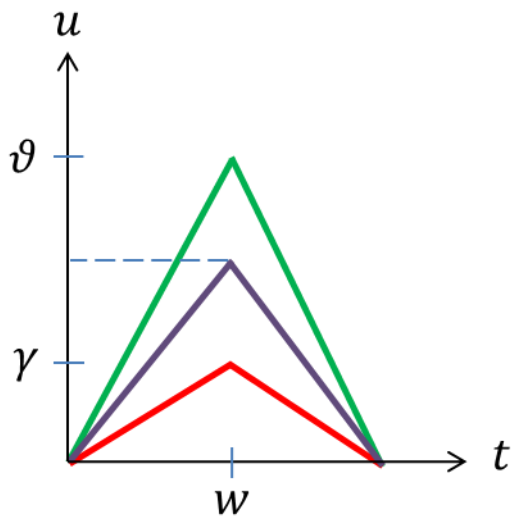

Figure 1.13. Associator version of the PSR and superposition to reach the threshold.

Assuming that each post-synaptic response to an incoming AP has a max height of γ (the red curve in the above figure), a pulse half-width of w ms, then the simple equation for a single post-synaptic response to an incoming spike is:

$$response(t) \stackrel{\text{def}}{=} \Lambda(t) = \gamma \begin{cases} 1 - \frac{1}{w}|t - w|, & 0 \le t \le 2w \\ 0, & otherwise \end{cases} \quad \textbf{(1.12)}$$

When multiple post-synaptic responses are generated within the neuron of interest at the same time, they superpose, generating a higher neuron potential (the purple curve). Finally, if there are a sufficient number of incoming pulses, the responses will superpose so that the neuron's threshold potential is reached (green curve).

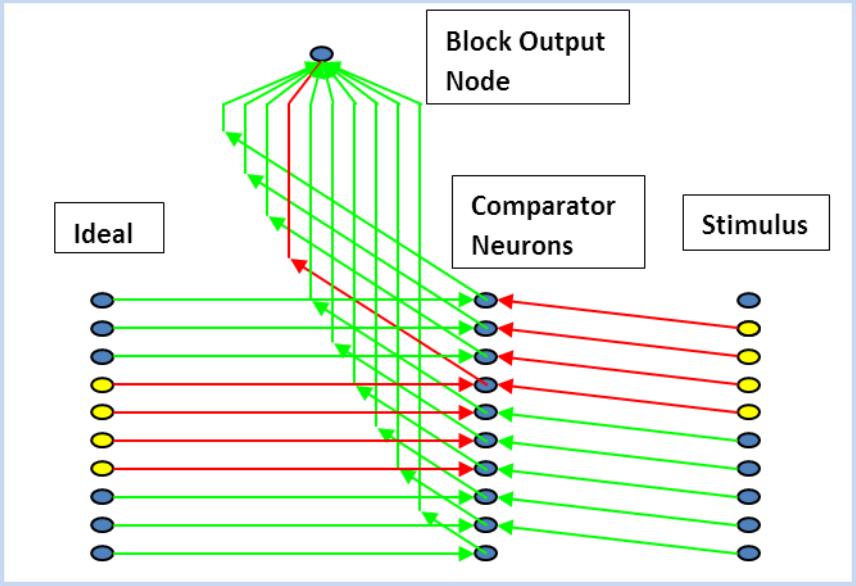

Figure 2.1. Depiction of the +1 Shift Determiner.

The goal of this analysis is a multilayered design which will find the best correlation between a presented stimulus and a remembered ideal pattern using the finite slope feature presented in the previous section. Because the neural architecture for recognizing 2D images is very complex, this explanation is confined to 1D strings of light and dark pixels. Similarly, only one remembered ideal pattern will be assumed. We describe the situation where the perceived string is only translated relative to the correct pattern in memory. Once this simple example is understood, the additions needed to extend to multiple remembered patterns and to 2D can be implemented.

2.1 The Shift Determiner Framework

The +1 Shift Determiner, represented in the box above, is a framework for detecting the translation, or shift, of a stimulus relative to an "ideal" pattern in memory. The stimuli and remembered ideal are binary patterns. In Figure 2.1, the light ovals represent 1s and the dark ovals represent 0s. Thus, the external stimulus is composed of a zero and four contiguous ones followed by five contiguous zeroes. The ideal contains the same pattern of ones, but centered.

At $t = 0$, the stimulus data and the remembered ideal are sent, as spikes, to comparator neurons. These "neurons" are similar to spiking neural network neurons in that they have a firing threshold, however, they do not have the weight coefficients or algorithms that mimic learning.

As shown in the figure above, each comparator neuron is wired to accept a pulse from the remembered ideal and from the external stimulus. In order to fire, the threshold of each comparator neuron must be reached. If a pulse has a max value of γ, then the threshold value, ϑ, for a comparator neuron is set to 2γ to ensure that firing occurs when the highest correlation is found, as seen in the figure below.

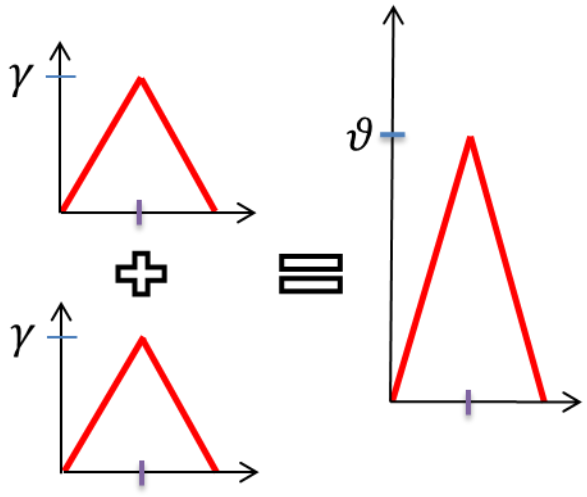

Figure 2.2. Demonstration of the superposition of postsynaptic responses required to reach threshold.

Thus, when the comparator neuron receives two pulses, the threshold will be reached and the neuron will fire. In the $+1$ shift example, only one comparator neuron receives a pulse from both the ideal and the stimulus, consequently, only that comparator neuron fires a pulse. The spikes from all comparator neurons are routed to the output node and summed as presynaptic input. The size of the total postsynaptic response of the output neuron is proportional to the correlation between the external stimulus and the remembered ideal.

2.2 Shift Determiner Blocks

Now, as we can see, the correlation between the stimulus and ideal can be increased by shifting the external stimulus down. If we look at the -1 shift in synapse

connections between the stimulus and the comparator neurons, we get the following representation: the Shift Determiner Block.

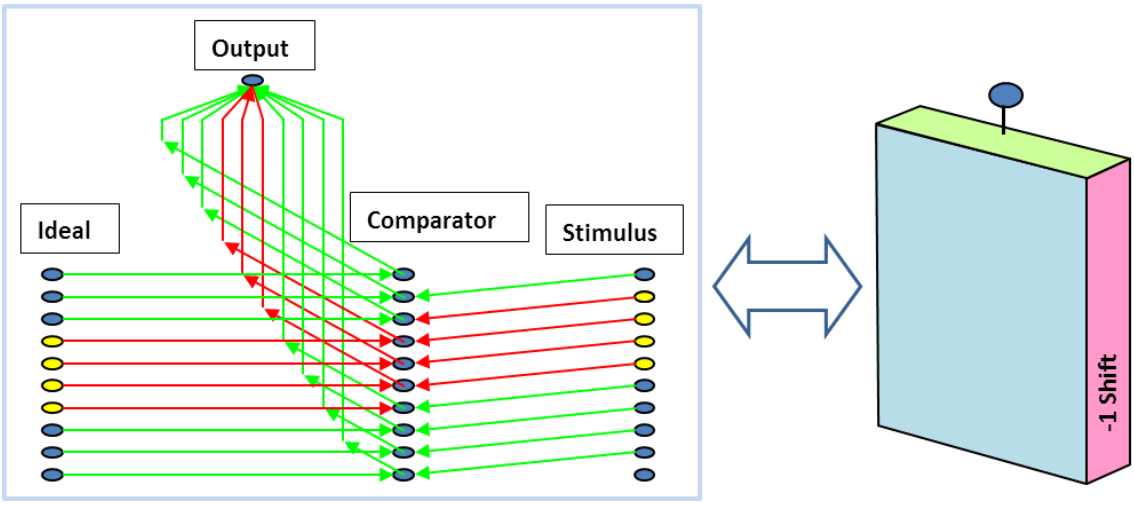

Figure 2.3. Equating the -1 Shift Determiner setup to the visual representation of a shift block.

The synapses between the stimulus and comparator neurons have been shifted down by two units, leading to a -1 positioning of the Shift Determiner. In the coming pages, we will symbolize the $\pm n$ Shift Determiner architecture by the block on the right.

Continuing with the -1 shift example, we can easily see that while that shift has a high correlation, a -2 shift will yield the highest correlation. In this simple example, it is easy to find highest correlation shift, however as the complexity increases, we need to know how many shifts can be investigated in order to find the shift leading to the highest

correlation. The question is: how many Shift Determiner Blocks are needed to account for all possible translations in a given stimulus?

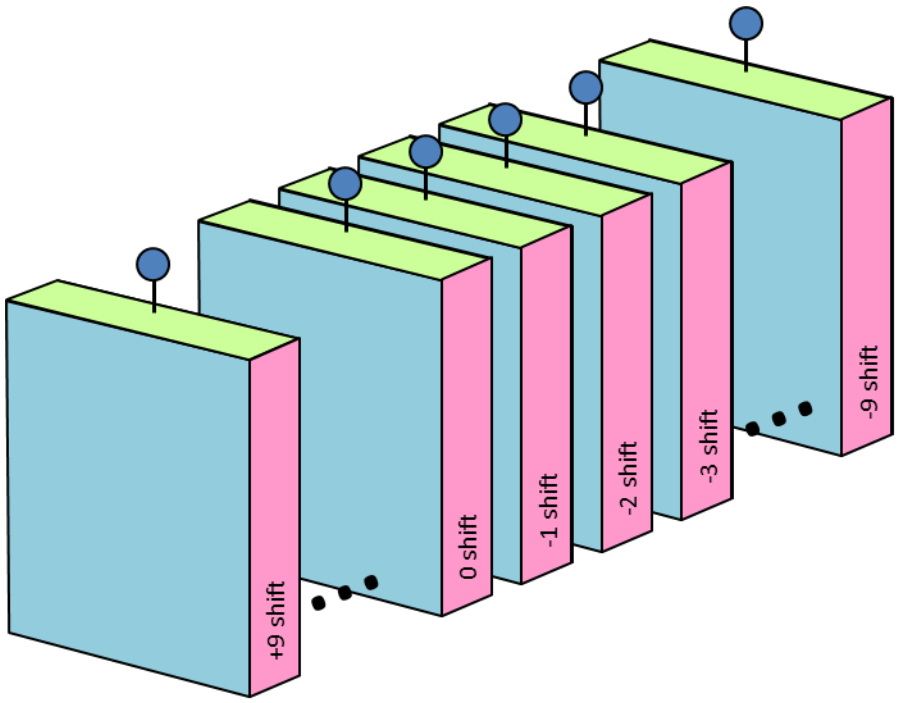

Figure 2.4. Depiction of the Shift Blocks required for the -1 shift example with 10 stimulus data points.

If we have an external stimulus and an ideal both composed of k values, then $2k-1$ Shift Determiner Blocks will be enough to find the translation between the remembered ideal and the external stimulus. On the other hand, if the ideal is composed of m units, then $k+m-1$ will be enough to find the translation between the remembered ideal and the external stimulus.

If we look at the -1 Shift example, there are 10 stimulus data points. Thus, $k = 10$, meaning 19 Shift Determiner Blocks would be able to account for any shift between the ideal pattern and the stimulus.

2.3 Shift Determiner Firing Condition

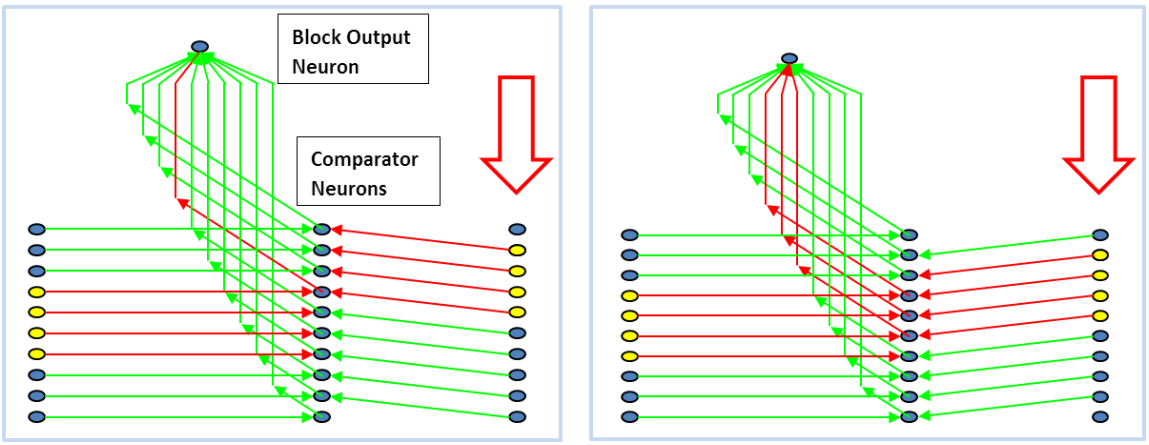

Figure 2.5. Comparison of the shifts +1 Shift Determiner (Left) and the -1 Shift Determiner (Right).

The final element of the $\pm n$ Shift Determiner Block that needs to be discussed is the firing condition of the block output neuron. Looking at the $+1$ and -1 Shift Determiner blocks, it would appear that setting a threshold for this neuron would presuppose something about the correlation between the ideal and the stimulus.

For example, if the threshold is set to 2γ as it is for the comparator neurons, then the underlying assumption would seem to be that there only need to be two points of

correlation between the ideal and the stimulus for firing to occur. However, due to the fact that the PSR has a finite, nonzero, positive slope, the more spikes that are superposed, the steeper the slope becomes, and the faster the threshold is reached. This concept is demonstrated in the two cases below.

Assuming γ is a neuron's potential response to a single pulse and 2γ is the neuron's threshold, then for case 1 shown in Figure 2.6, when the single post-synaptic responses are summed, the height of the resulting potential is 2γ. Note that in this case 2γ is reached at a time interval equal to half the width of the post-synaptic response.

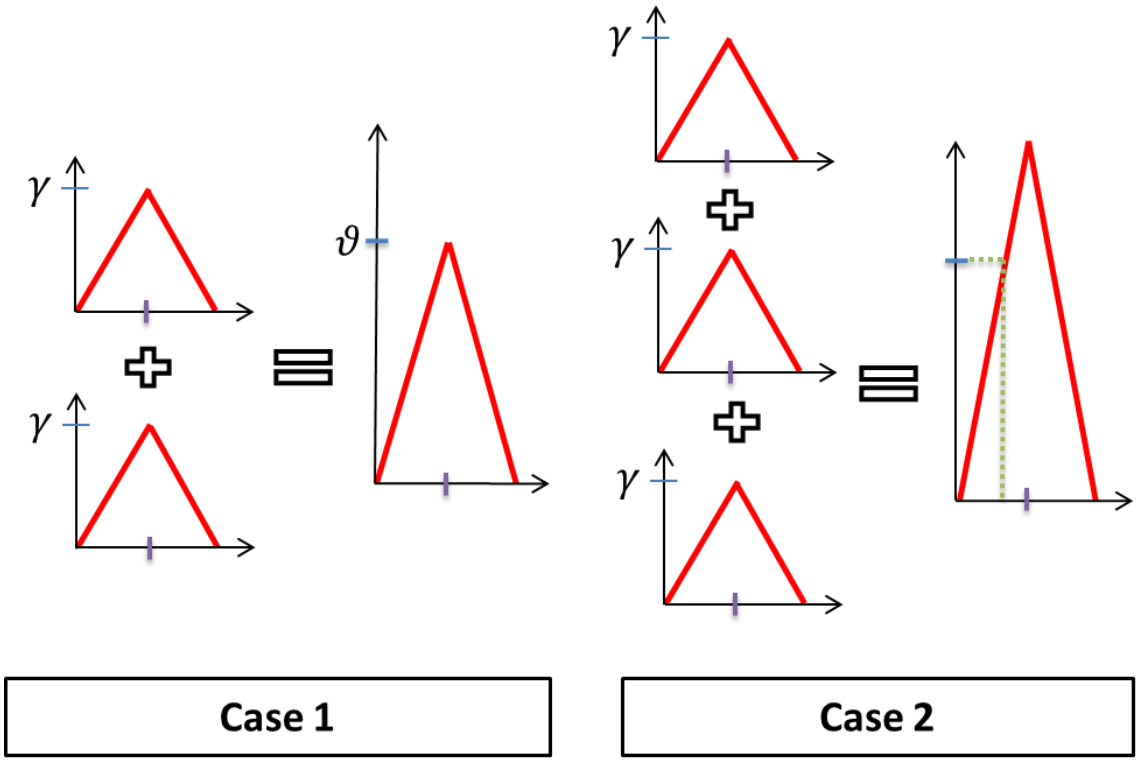

Figure 2.6. Visual example: more incoming pulses lead to faster firing.

In the second case, three post-synaptic responses are added, creating a potential that reaches the 2γ threshold value at a time interval that is less than half the pulse width. So, the second case brought the neuron to the threshold value at an earlier time than in the first case. This means that the Shift Block that generates the most comparator neuron pulses will cause its block output neuron to fire before the output neurons of all the other Shift Blocks. Thus, the "winner" (most correlated shift block) is singled out through competition between shift blocks, and the block with the most correlations fires before all the other block output neurons.

2.4 The Associator Structure

Finally, we note that all the block output neurons are interconnected with inhibitory synaptic weights. When the "winner" block output neuron fires, its pulse inhibits the firing of all other output block neurons. Going back to our previous example to see how this applies, the first Shift Determiner Block to reach the threshold would be the -2 Shift Block. The block output neurons of all blocks are "wired," for two-way communication, to the block output neurons of all other Shift Blocks, as shown in the next figure.

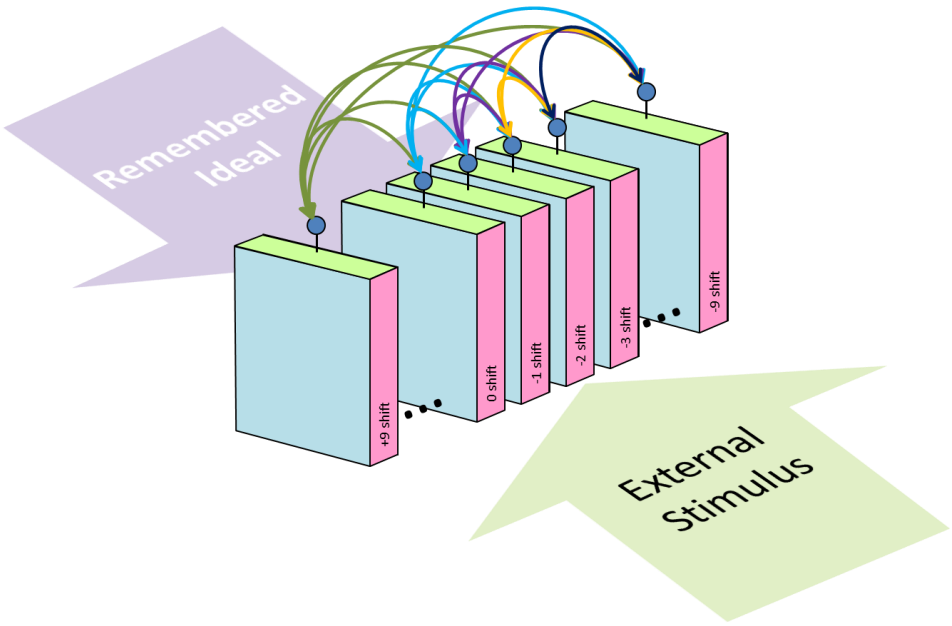

Figure 2.7. Depiction of inhibition wiring among the Shift Block output neurons.

Thus, a pulse from the -2 Shift Block would reach all the other output neurons and inhibit them, so they do not fire. The full framework is depicted in the figure below by replacing the representations of the inter-block connections in the figure above with the double headed arrows and adding a final output node. The final output node receives the pulse of the block with highest correlation.

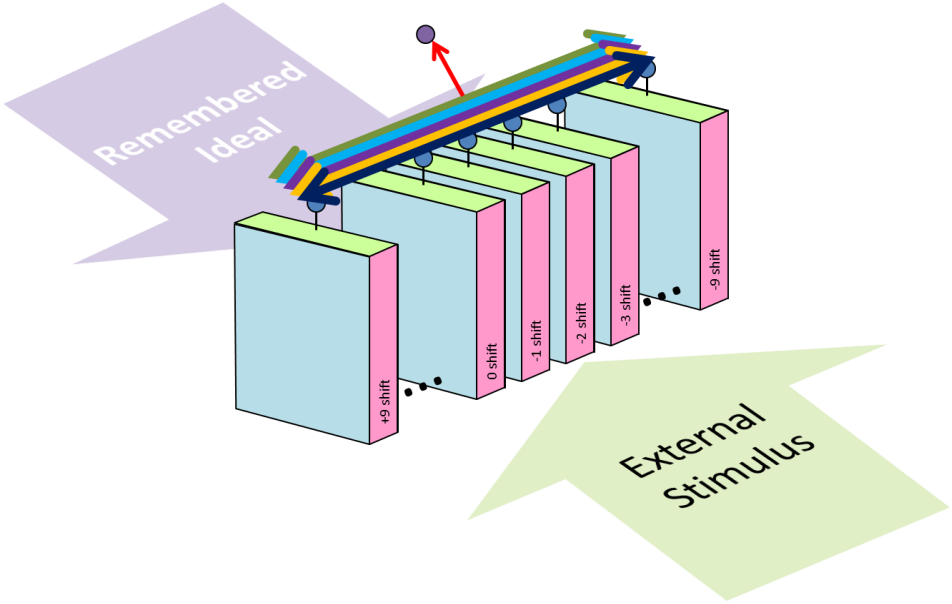

Figure 2.8. Compact depiction of the Shift Blocks, inhibitory connections, and final output.

In order to facilitate the explanation of the next section, the above image will first be depicted compactly. Thus, the following figure shows the new representation of all the Shift Blocks, the inhibiting connections, and the final output node as a single module hereafter referred to as an Associator.

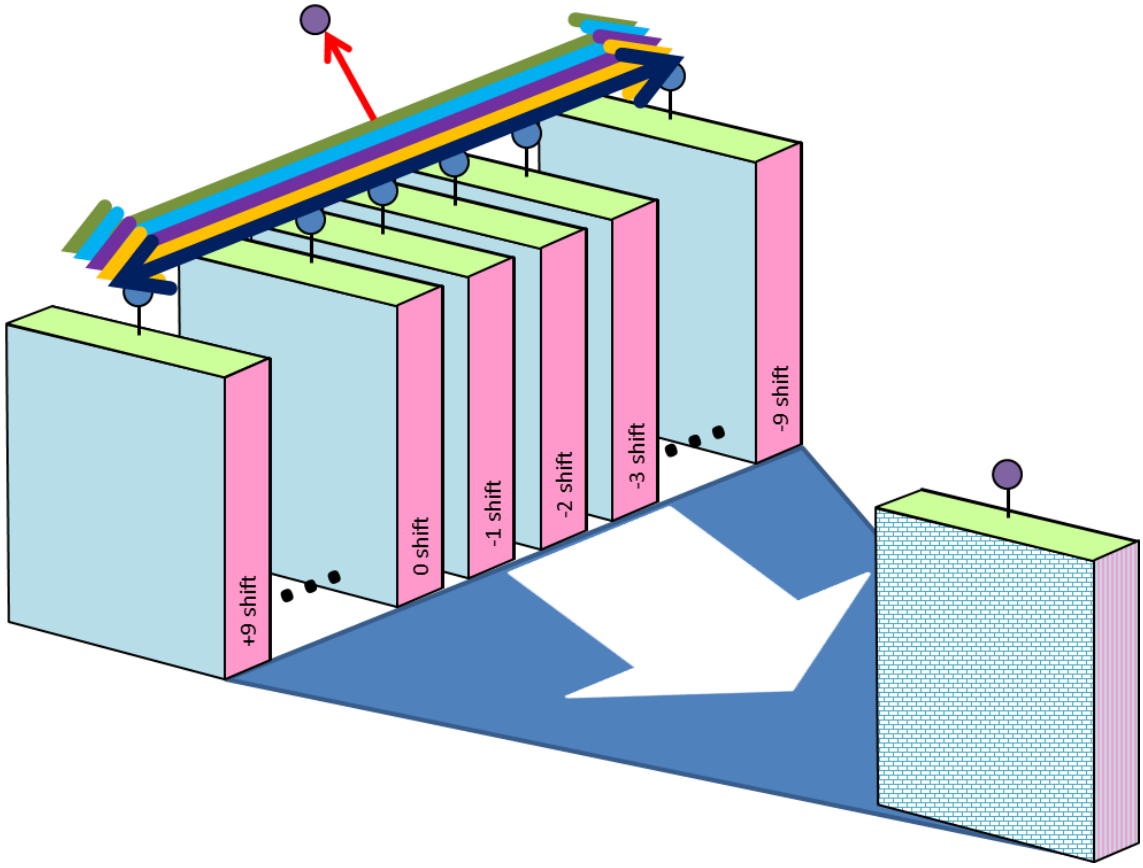

Figure 2.9. Depiction of the Shift Blocks that make up the Associator.

The function of the Associator, starting from the basic unit, the Shift Determiner, has been established: competition between shift blocks, inhibitory connections between shift blocks, and the final output node receiving the "winning" pulse.

The final piece that the Associator addresses is the assumption of a single remembered ideal. For simplification, the initial assumption was that only one ideal would be available. However, in order for this design to be of use, it needs to be capable of "remembering" various ideals and distinguishing which remembered ideal has the

highest correlation with the external stimulus. We will now address multiple ideals, or the Competitive Classifying Unit (CCU).

Each Associator in the CCU takes in unique remembered ideal as well as the external stimulus. The Associators are inhibitorily connected to each other to extend the competition to include competition between the remembered ideal patterns.

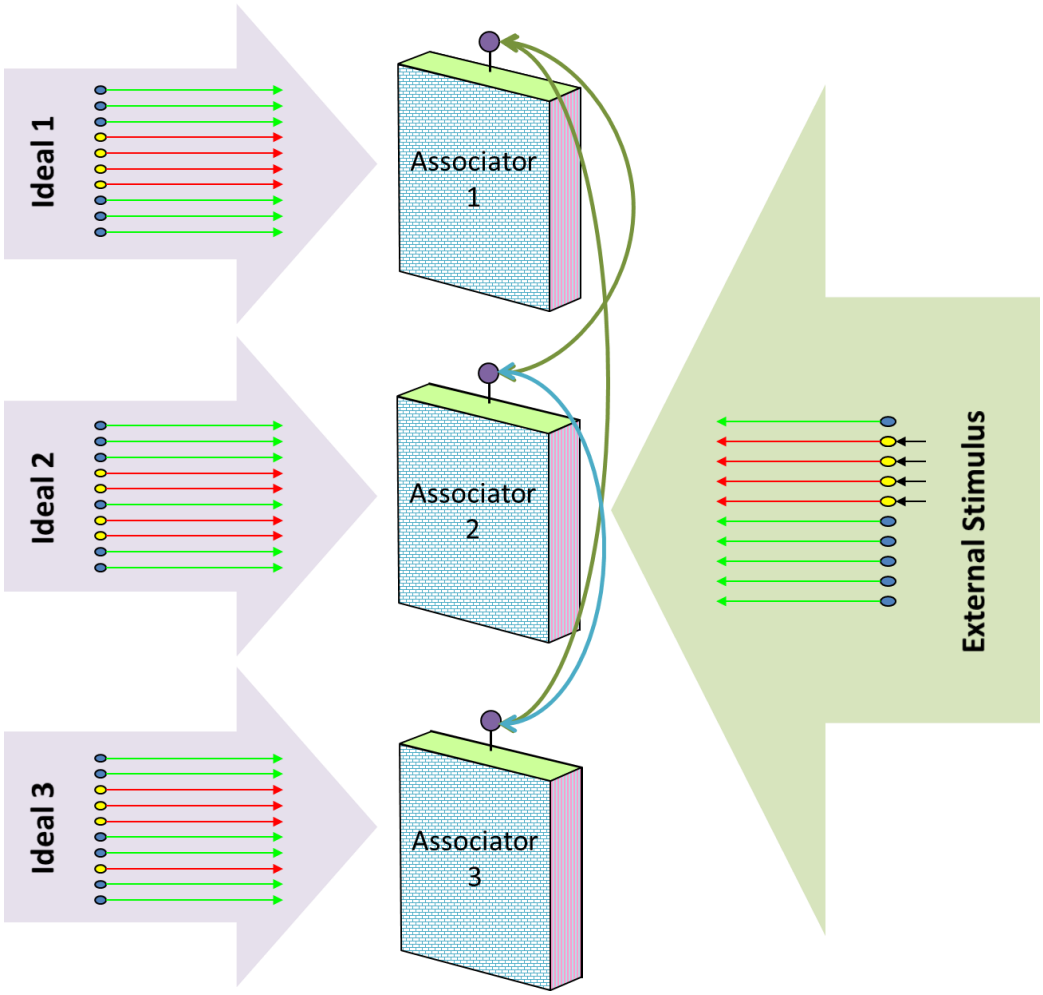

Figure 2.10. Depiction of the Competitive Classifying Unit matching an external stimulus to the correct ideal.

For example, we see that Associator 1 will fire before the other Associators because, as seen before, Ideal 1 will have the highest correlation with the -2 Shift Block. Thus, the CCU will be able to classify more than just a single stimulus.

2.5 Case Study: Sensitivity to Non-concurrently Arriving Impulses

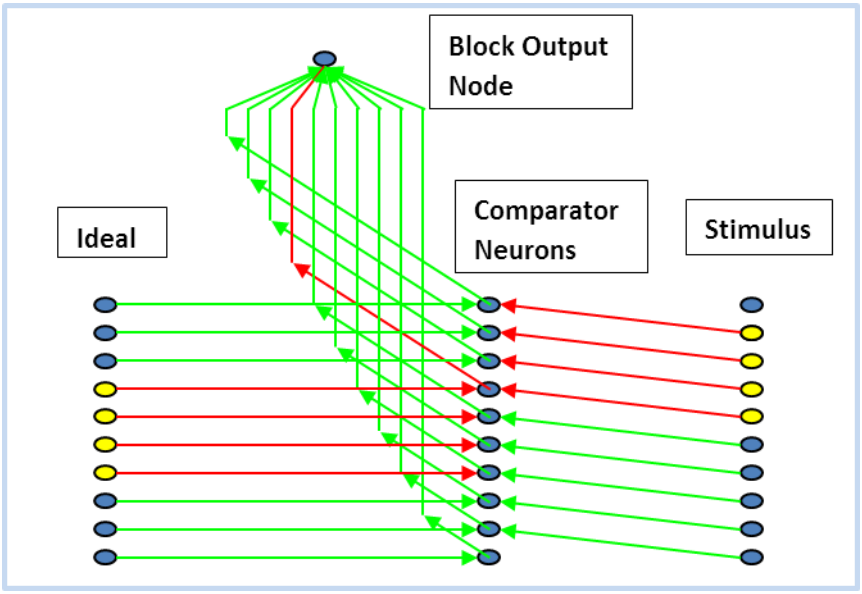

Figure 2.11. The Shift Determiner.

The simplified case of the Shift Determiner has been investigated holding to the following assumptions:

1) The remembered ideal and the external stimulus are input to the Shift Determiner at the same time instant.

2) Both the remembered ideal and external stimulus are 1D.

3) Only a single remembered ideal pattern is available.

At the end of the section entitled the Associator Structure, assumption 3 was addressed and expanded so as to account for the remembrance of multiple ideal patterns. In this section, we will address assumption 1.

We start with the simplest case, the case in which there is a single comparator neuron receiving a 1 from both the remembered ideal and the external stimulus. However, in this instance, one of the two inputs arrives at the comparator neuron later than the other input. What we will determine is how sensitive the Shift Determiner is to variation in the input arrival time of the ideal and stimulus.

Figure 2.12. Depiction of a single comparator neuron receiving ideal and stimulus inputs.

If the two inputs arrive at the same instant, the comparator neuron's potential response was previously determined to be as in the figure below.

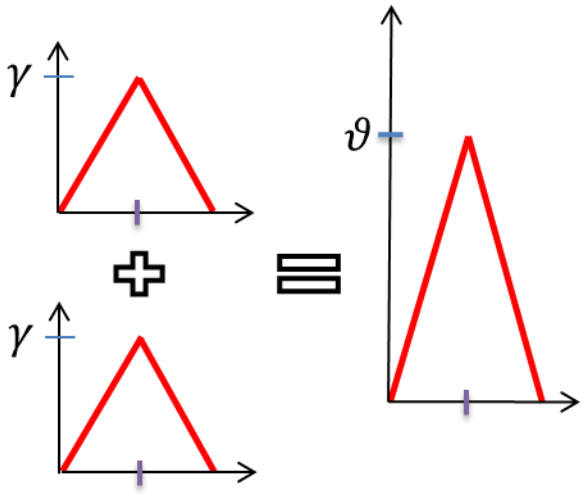

Figure 2.13. Depiction of the response of a comparator neuron to two concurrently arriving pulses.

The two responses superpose, resulting in a pulse height that is equal to 2γ. However, as pulse arrival times are varied, becoming farther apart, the superposition yields a smaller and smaller max potential. The following figures assume the standard value of a single postsynaptic response, γ, is 1, the abscissa is time, and the ordinate is the measured response.

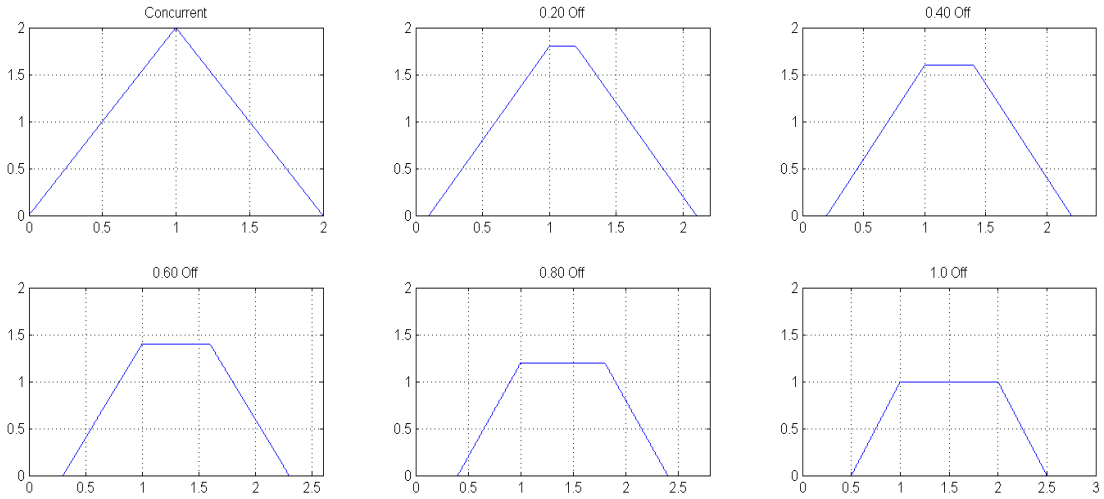

Figure 2.14. The effect of increasingly non-concurrent pulse arrival times on neuron potential.

It is apparent that if the threshold for the comparator neuron is 2, as previously assumed, then the neuron will only fire in the case in which the responses occur concurrently. If it is not possible for the responses to be triggered concurrently, then the comparator neuron threshold may be lowered according to the following equation,

$$(n - \Delta t_d)\gamma > \vartheta \text{ where } \Delta t_d < 1 \qquad \textbf{(2.1)}$$

where n is the number of incoming responses and $\Delta t_d$is the time delay between responses. It should be noted that to avoid false positives, Δt_d needs to be less than 1. Otherwise, as can be seen in the example response graphs above (1.0 Off), the neuron would fire with only one incoming response, giving a false result.

This brief analysis of non-concurrent arrival times raises the question of how much delay time the design can withstand before the system will no longer match the

stimulus to the correct ideal. If the threshold is set just above $n-1$, then the maximum allowable delay time before any false values occur is $\Delta t_d = 1 - \epsilon$ (where ϵ is a number much less than 1) coupled with the threshold set according to the equation above. Thus, the system will function correctly as long as the arriving pulses are just short of a pulse width apart.

3. SENSITIVITY ANALYSIS OF RANDOMLY VARYING THRESHOLD VALUES

This section contains the simulation and performance of the Shift Determiner framework with 1D inputs. The analysis inputs a stimulus and a single ideal pattern without noise into the comparator neurons of a shift determiner. The same ideal pattern is also introduced to the comparator neurons separately. Here we study the sensitivity of the system to random variations in the firing threshold.

Reviewing the Shift Determiner process, if the input value from the ideal pattern and the corresponding input from the stimulus are both 1, the comparator neuron will fire, inducing a postsynaptic response of γ in its block output neuron. There is no randomness in the comparator neuron threshold.

Thus, each shift block takes in a version of the stimulus that has been shifted by a single unit to provide alternatives for comparison with the ideal. The block output neuron of each shift block has a random threshold taken from a uniform distribution. This threshold is chosen from an interval to determine the design's sensitivity to randomness in the threshold.

Finally, the block output neuron that receives the most inputs will have a potential that rises more steeply than the other blocks. As soon as the potential of the block output neuron reaches the threshold value, the block output neuron will fire a pulse inducing a PSR of γ in the Associator neuron. Whichever block output neuron pulse

that arrives at the Associator neuron first will correspond to the correct shift that causes the greatest equality between the ideal and the stimulus.

3.1 Experiment with Randomly Varying Thresholds without Noise

The following stimulus and ideal pattern were used for all experiments.

stimIn = [0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0]

idealIn = [0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

The correct shift for all experiments is 7 due to the zeroes the program pads both sides of the stimulus and ideal with in order to facilitate comparison over the various shifts. Six runs of 100 trials each were performed to determine the relationship between the framework's accuracy and the amount of randomness in the threshold. The accuracy of the 'winning' shift was found by comparing the true shift value of the stimulus to the shift selected as 'winner' by the Shift Determiner framework. The results, presented graphically in Figure 3.1, show that as the standard deviation of the random threshold decreases, the accuracy of the program increases. From the graph, with a standard deviation in the firing threshold of ~0.8, 90% of the selected winners will be accurate.

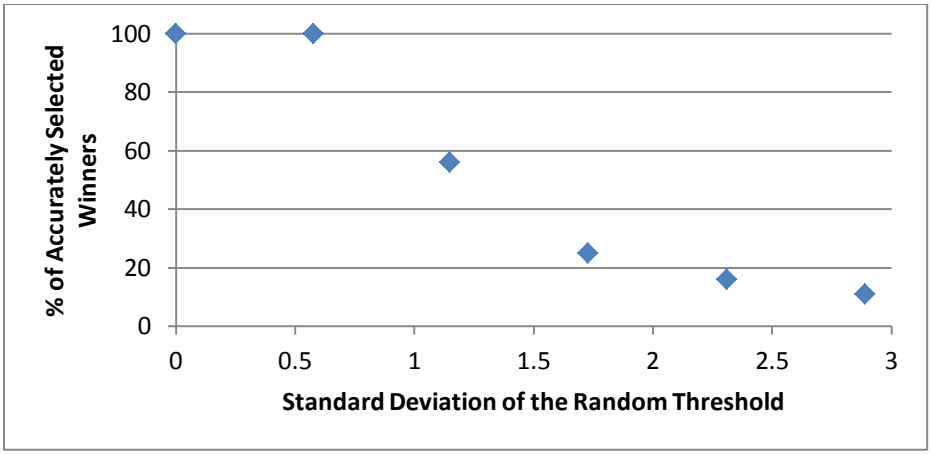

Figure 3.1. The percentage of accurately selected winners v standard deviation in randomly varying firing threshold values where each data point is 100 trials.

The standard deviation, σ, for a uniform distribution is

$$\sigma = \frac{b - a}{\sqrt{12}} \tag{3.1}$$

where b and a are the upper and lower bounds of the uniform distribution. In this case, the max possible potential (seen by the Block Output neuron) due to matching of the stimulus to the ideal is 5γ and the minimum is 0. Thus the standard deviation for this uniform distribution is 1.4γ.

In order to vary the randomness of the threshold for the different trials, one of the bounds must be changed so that the threshold is chosen from a different interval. For ease in finding the changes due to randomness in the threshold, a varying percentage, P, of the lower bound, a, was used to calculate the threshold and then used in the calculation of the standard deviation. In the equation below, a and P are directly related,

so as P is increased toward 100%, the threshold is selected from a narrower interval that contains higher values.

$$a = bP \tag{3.2}$$

Thus leading to the standard deviation equation

$$\sigma = \frac{b(1-P)}{\sqrt{12}}. \tag{3.3}$$

The correlation between the standard deviation and the average 'winning' threshold for the previously mentioned example is shown below. The range around each value shows the range from with the average firing threshold could be chosen for those 100 trials.

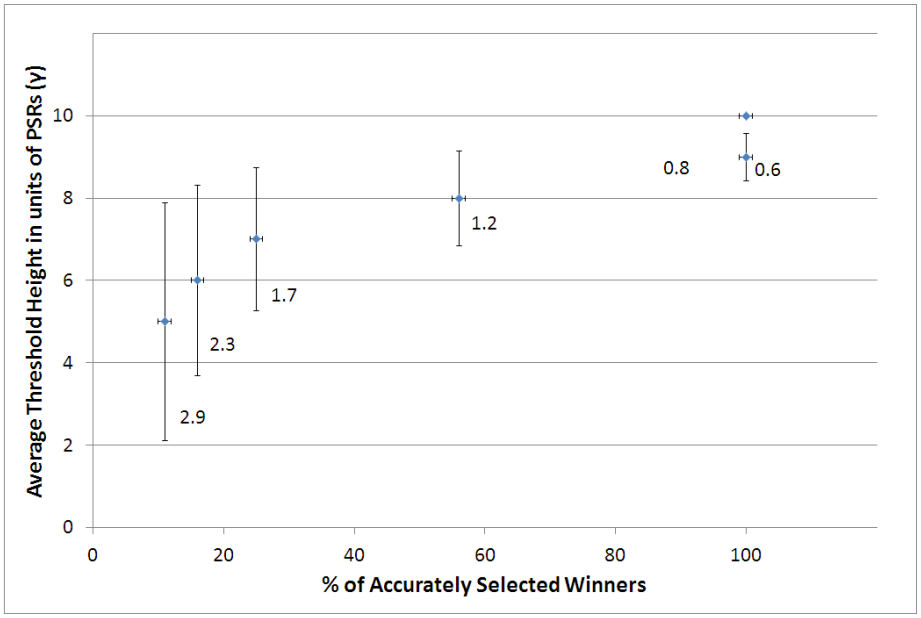

Figure 3.2. Average winning threshold versus standard deviation in average firing threshold.

As expected, when the threshold has no randomness, the average 'winning' threshold is equal to the true value, 5γ. But, as the interval in which the threshold may be chosen increases, the average 'winning' value gets further from the true value.

3.2 Experiment with Randomly Varying Thresholds with Noise

In this test, noise was added to the previously used stimulus by a bit-flipping rule and subsequently introduced to the program for analysis. The noise injection method is as follows: each element, i of a random vector, v, composed of elements from the uniform distribution is compared to a user specified value, p. If $v_i \leq p$, the i^{th}element of the stimulus vector, s, is flipped to the alternative value. In this method, a high value for p indicates a greater amount of noise in the stimulus vector.

The accuracy of the 'winning' shift was determined by comparing the true shift value of the stimulus to the shift selected as 'winner' by the Shift Determiner framework. For each of the nine values of p between 0 and 1, the noisy input stimulus and the shifted ideal were run through the Shift Determiner. The following tables give example inputs and outputs for the nine values of p.

The red boxes outline the program-selected shift that finds the ideal within the stimulus. Note that the first three selections share the same shift value and also find the ideal within the noise. Examining the other selections shows that while the original shift location of the pattern within the image is not maintained due to the noise, the Shift Determiner does find the ideal pattern that, in these cases, has been created by the noise.

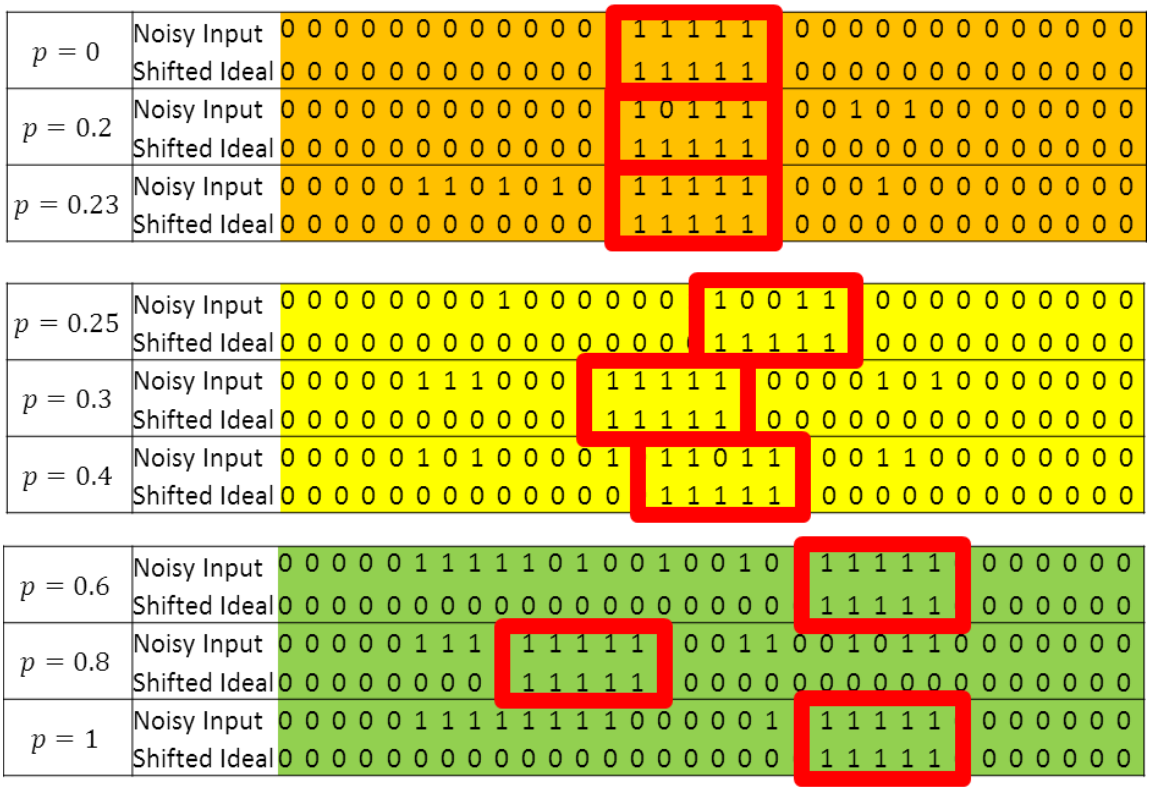

Figure 3.3. Presentation of stimulus and ideal patterns selected by the Shift Determiner with increasing noise in the input.

As previously stated, the accuracy of the program is determined by comparing the original shift value to the winning shift value selected by the Shift Determiner. As noted, only the trials for $p \leq 0.23$ find the ideal pattern at the location of the original shift. Thus, only these values are indicated as accurately selected winners in the graph below. The graph does show how the percentage of accurate "winners" drastically decreases as the noise value increases. While the program does find the ideal pattern within the greater p values, that is only possible because the simplicity of the ideal pattern means that the pattern will likely be generated by the noise itself.

4. NETWORK THEORY

In this section, the theory of homeostasis and periodic spike timing is investigated with a view to expand the area of inquiry beyond matching and into the learning/memory portion. As explained, the CCU matches a given stimulus to an ideal stored in memory. The following section outlines the theoretical requirements of a network of neurons with the dedicated purpose of storing data.

Firstly, the theory of homeostasis—the biological term describing the constraint on total neural chemicals required for firing and the natural tendency for an constant state to be maintained—is carried through to its logical conclusions with respect to a neural network. Explicitly, homeostasis implies periodic, synchronous firing within a network or cluster of neurons, as aperiodic firing will result in the shutdown of the network (catatonia). This is due to the fact that multiple firings are required to induce a downstream neuron to fire. The spiking neural network model is discretized in terms of the pulse times, the differential equations for firing are provided, and the mathematical proofs for synchronicity and periodicity are shown. Finally, the issue of network learning is addressed with particular attention to the order of training a network of neurons and the spike time window.

4.1 Periodic Spike Timing

The starting point is the phenomenological spike response model based on Gerstner and Kistler but which differs slightly from the SRM_0 presented in the first section. The state of neuron i is described by the neuron potential, $u_i^{tot}(t)$, which has a resting value of zero. If the neuron potential reaches a threshold, θ, an output spike is triggered. Assuming that the last spike is fired at time $\hat{t}_i$, the evolution of the total potential is given by:

$$u_i^{tot}(t) = \sigma(t - \hat{t}_i) + u_i(t) \tag{4.1}$$

$$u_i(t) = \eta(t - \hat{t}_i) + \sum_j w_{ij} \sum_{t_j^{(f)}} \varepsilon_{ij}\left(t - \hat{t}_i, t - t_j^{(f)}\right) + \int_0^\infty \kappa(t - \hat{t}_i s) I(t - s) ds$$

Here, $\sigma(t - \hat{t}_i)$ describes the action potential or spike, while $u_i(t)$ is the relatively slowly varying response after the spike. The function η represents the afterpotential, or negative overshoot which typically follows a spike. This gives rise to refractoriness, i.e. the observation that for a period of time after the action potential, it is much more difficult to trigger a second spike. The function ε_{ij} represents the time response of neuron i to an incoming spike from presynaptic neuron j. w_{ij} is the synaptic efficacy of the synapse receiving input from neuron j and transferring it to neuron i. Unless otherwise stated, w_{ij} is assumed non-negative for all i and j, and is allowed to be time-varying The two sums run over all presynaptic neurons, j, and all firing times $t_j^{(f)} < t$ of neuron j.

Firing occurs whenever the (post-spike) membrane potential reaches the threshold θ:

$$t = t_i^{(f)} \Leftrightarrow u_i(t) = \theta, \text{ and } \frac{du_i(t)}{dt} > 0 \qquad \textbf{(4.2)}$$

$$\hat{t}_i = \max\left\{t_i^{(f)} < t\right\}$$

θ may be time-varying, but for this exposition we assume it to be constant.

As stated previously, the spike shape, $\sigma(t - \hat{t}_i)$, is essentially the same for all neurons and all spikes. Moreover, the learning rule for adapting the synaptic efficacies is also independent of the spike shape. Hence, $\sigma(t - \hat{t}_i)$ carries no essential information and we henceforth model it by a delta function. Thus we characterize the neural response by $u_i(t)$ and the above threshold crossing firing rule.

Next, we simplify this model in order to render the analysis of the dynamics of learning more intelligible and reasonably tractable. Analysis of the more complex model given by (4.1) is deferred.

First, in accordance with the simplified spike response model SRM_0 given by Gerstner and Kistler in [30], the function $\varepsilon_{ij}\left(t - \hat{t}_i, t - t_j^{(f)}\right)$ is assumed independent of $\hat{t}_i$. We ignore the external input term, since training inputs will be achieved by constraining the firing times of a subset of "output" neurons. Further, we choose to make $\varepsilon_{ij}\left(t - t_j^{(f)}\right)$ uniform for all neurons, i.e. $\varepsilon_{ij}\left(t - t_j^{(f)}\right)$=$\varepsilon\left(t - t_j^{(f)}\right)$. At this point, our model is:

$$u_i(t) = \eta(t - \hat{t}_i) + \sum_j w_{ij}\left(t_j^{(f)}\right) \sum_f \varepsilon_{ij}\left(t - t_j^{(f)}\right) \tag{4.3}$$

$$t = \hat{t}_i \Leftrightarrow u_i(t) = \theta, \text{ and } \frac{du_i(t)}{dt} > 0$$

$$\hat{t}_i = \max\left\{t_i^{(f)} < t\right\}$$

In the above equation, we allow w_{ij}, $j = i$ to be nonzero, representing a self-stimulating capability. This allows autonomous, periodic firing of an individual neuron. In addition, we impose the following conditions on the functions $\eta(t)$ and $\varepsilon(t)$:

C.1: Both $\eta(t)$ and $\varepsilon(t)$ vanish for non-positive arguments.

C.2: For non-negative arguments, $\eta(t)$ and $\varepsilon(t)$ are continuous and differentiable up to second order.

C.3: $\eta(0) = 0, \quad \eta(t) \leq 0 \; \forall \; t \geq 0, \eta(t)$ increases monotonically from the minimum value $-\hat{\eta}(\hat{\eta} > 0)$ at $t = 0$.

C.4: $\varepsilon(0) = 0, \varepsilon(t) \geq 0 \; \forall \; t \geq 0, \varepsilon(t)$ has the globally maximum value $\hat{\varepsilon}(> 0)$ at $t = \tau_\varepsilon$.

C.5: $\hat{\varepsilon} < \hat{\eta}, \tau_\varepsilon < \tau_\eta$

Simple specific choices for these functions might be:

$$\eta(t) = \begin{cases} -\hat{\eta} exp\left(-t/t_\eta\right), t > 0 \\ 0, t \leq 0 \end{cases} \tag{4.4}$$

$$\varepsilon(t) = \begin{cases} \hat{\varepsilon}\dfrac{t}{t_\varepsilon}\exp(1 - t/t_\varepsilon), t > 0 \\ 0, t \leq 0 \end{cases}$$

Typically, $\hat{\eta} \approx 10\hat{\varepsilon}$, $\theta/\hat{\varepsilon} \approx 10 - 30$, and $\tau_\eta \approx 3\tau_\varepsilon$. Thus it takes some tens of post-synaptic responses to trigger a spike (assuming the efficacies to be of order unity on the average).

4.2 Dynamic Model of Spike Times

It is generally agreed that it is the spatiotemporal pattern of the spiking times that encodes information within the network. Therefore, instead of phrasing the neural dynamics in terms of the neural potentials, we use (4.3) to obtain the equations determining the spiking times themselves.

First, let us order the $t_i^{(f)}$ as follows:

$$t_i^{(1)} \leq t_i^{(2)} \leq \cdots \leq t_i^{(n)} \tag{4.5}$$

where $t_i^{(1)}$ is the first spike of neuron i following some initial time and $\hat{t}_i = t_i^{(n)}$. Next, define the time interval between successive spikes of neuron k:

$$C_k(n) = t_k^{(n)} - t_k^{(n-1)} \tag{4.6}$$

and let:

$$T_k(n) = t_k^{(n)} = \sum_{n=1}^{n} C_k(n) \tag{4.7}$$

denote the n^{th} spike time of neuron k.

Now consider Equation (4.3a) at time $t_k^{(n+1)}$. Clearly, $(t_k^{n+1}) = \theta$, so that (4.3a) yields:

$$\theta = \eta\big(C_i(n+1)\big) + \sum_j \sum_m w_{ij}(m) \begin{Bmatrix} \varepsilon\left(T_i(n+1) - T_j(m)\right), if\ T_j(m) \geq T_i(n) \\ 0, otherwise \end{Bmatrix} \quad \textbf{(4.8)}$$

where $w_{ij}(m)$ denotes the value of the efficacy at time $t_j^{(m)}$. To streamline notation, we introduce the right-continuous Heaviside function:

$$H(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \quad \textbf{(4.9)}$$

Then we write the expression $\begin{Bmatrix} \varepsilon\left(T_i(n+1) - T_j(m)\right), if\ T_j(m) \geq T_i(n) \\ 0, otherwise \end{Bmatrix}$ as

$\varepsilon\left(T_i(n+1) - T_j(m)\right) H\left(T_j(m) - T_i(n)\right)$ to obtain:

$$\theta = \eta\big(C_i(n+1)\big) + \sum_j \sum_m w_{ij}(m) \varepsilon\left(T_i(n+1) - T_j(m)\right) H\left(T_j(m) - T_i(n)\right) \quad \textbf{(4.10)}$$

In view of (4.3c), $C_i(n+1)$ is the minimum positive quantity that satisfies the above relation. Hence, noting that $T_i(n+1) = T_i(n) + C_i(n+1)$ we obtain, for a group of N neurons with all-to-all coupling:

$\forall i = 1, \ldots, N$, and $n = 0,1, \ldots$:

$$C_i(n+1) = min\left\{ \begin{matrix} \Lambda > 0: \\ \theta = \eta(\Lambda) + \sum_j \sum_{m \geq 0} w_{ij}(m) \varepsilon\left(\Lambda + T_i(n) - T_j(m)\right) H\left(T_j(m) - T_i(n)\right) \end{matrix} \right\} \quad \textbf{(4.11)}$$

$$T_i(n+1) = C_i(n+1) + T_i(n)$$

Note that since the term on the right-hand side within the brackets begins at zero at $\Lambda = 0$, it can only reach θ if its slope is positive. Hence the condition $t = \hat{t}_i \Leftrightarrow \frac{du_i(t)}{dt} > 0$ in (4.3b) is already implied. We conclude that (4.11) determines the spiking times implied by (4.3a-c).

4.3 Synchronous and Periodic Spike Times

Operation of the network (4.11), including synaptic plasticity evidently requires a level of vigorous neural activity in the form of frequent spiking. Yet, in many instances it takes tens of pre-synaptic spikes to cause a post-synaptic neuron to fire. One could imagine a situation where the pre-synaptic spikes arrive incoherently and the resulting post-synaptic response is insufficient to attain the threshold potentiation. Were this to happen for a sufficient fraction of neurons, the spiking activity of system (4.11) could decline and eventually cease altogether. Experimental evidence for synchronous firing can be found in [31]. Here we consider how to improve the efficiency with which the spiking frequency can be stimulated.

First note that it is the accumulation of post-synaptic responses represented by the term $\sum_j \sum_{m\geq 0} w_{ij}(m)\varepsilon\left(\Lambda + T_i(n) - T_j(m)\right) H\left(T_j(m) - T_i(n)\right)$ that builds up to exceed the threshold and cause a spike. In view of C.4, the postsynaptic response can never be larger than when the presynaptic spikes all occur at the same time. To be precise, define:

D.1: The spiking of system (4.11) is *synchronous* when

$T_j(n) = T_i(n), \forall i, j = 1, \dots, N, and\ \forall n = 0,1, \dots$

D.2: The spiking of the system in (4.11) is *periodic* when $C_i(n)$ is a constant

$\forall i, j = 1, \dots, N, and\ \forall n = 0,1, \dots$

In connection with these definitions we have:

Theorem 1

For system (4.11) and the related assumptions:

(a) A necessary condition for synchronous spiking is that the sum of the weights from firing neurons leading to a given neuron at a given time step, $\sum_j w_{ij}^f(m)$, equals a constant that is independent of i. In other words:

$$\sum_j w_{ij}^f(m) = N\bar{w}(m)\,, \forall i \tag{4.12}$$

where $\bar{w}(m)$ is a non-negative function of the spike index m.

(b) Under condition (4.12) and D.1, the spike intervals are given by:

$$\begin{gathered} \forall i = 1, \dots, N; and\ n = 0,1, \dots: \\ C_i(n+1) = \bar{C}(n+1) \\ \bar{C}(n+1) = \min\{\Lambda > 0 : \theta = \eta(\Lambda) + N\bar{w}(n)\varepsilon(\Lambda)\} \end{gathered} \tag{4.13}$$

when a solution to (4.13) exists.

(c) A necessary condition for spiking to be both synchronous and periodic is that:

$$\sum_j w_{ij}^f(m) = N\bar{w}\,, \forall i, \forall m = 0,1, \dots \tag{4.14}$$

where $\bar{w}$ is a positive, real number.

(d) Under condition (4.14), and D.1 and D.2, the spike intervals are all constant and equal to $\bar{C}$, given by:

$$\begin{gathered} C_i(n) = \bar{C}, \forall i = 1, \dots, N; \text{and } n = 0,1, \dots \\ \bar{C} = min\{\Lambda > 0 : \theta = \eta(\Lambda) + N\bar{w}\varepsilon(\Lambda)\} \end{gathered} \tag{4.15}$$

when a solution to (4.15) exists.

Proof:

(a) In view of C.1, $\varepsilon\left(\Lambda + T_i(n) - T_j(m)\right)$ vanishes if $T_j(m) > \Lambda + T_i(n)$, and because of D.1, the number of spikes occurring up to any time is the same for all neurons. Therefore the summation over m in the term $\sum_j \sum_{m\geq 0} w_{ij}^f(m)\varepsilon\left(\Lambda + T_i(n) - T_j(m)\right) H\left(T_j(m) - T_i(n)\right)$ is restricted to $m = n$. Hence (4.11b) becomes:

$$C_i(n+1) = min\left\{\Lambda > 0 : \theta = \eta(\Lambda) + \sum_j w_{ij}^f(n)\varepsilon\left(\Lambda + T_i(n) - T_j(n)\right) H\left(T_j(n) - T_i(n)\right)\right\} \quad \textbf{(4.16)}$$

Moreover, $T_i(n) = T_j(n)$, so that the Heaviside function assumes the value unity and (4.16) becomes:

$$C_i(n+1) = min\left\{\Lambda > 0 : \theta = \eta(\Lambda) + \varepsilon(\Lambda) \sum_j w_{ij}^f(n)\right\} \quad \textbf{(4.17)}$$

Finally, D.1 implies that all the increments, $C_i(n)$, are the same function of n, denoted by $\bar{C}(n)$. Then (4.17) gives:

$$\bar{C}(n+1) = min\left\{\Lambda > 0 : \theta = \eta(\Lambda) + \varepsilon(\Lambda) \sum_j w_{ij}^f(n)\right\} \quad \textbf{(4.18)}$$

Since this must hold for all i, it is necessary that $\sum_j w_{ij}^f(n)$ be independent of i. This proves part (a).

(b) Using (4.12), (4.18) yields (4.13).

(c) Under D.1 and D.2 the $C_i(n)$ are all the same function of n, denoted by $\bar{C}(n)$, and $\bar{C}(n)$ is the same positive constant, $\bar{C}$, for all n. Then (4.18) holds for $\bar{C}(n+1) = \bar{C}$ and (4.15) follows.

Regarding existence of solutions to (4.13) and (4.15), we have the following:

Theorem 2

Under C.3 and C.4, if

$$N\hat{\varepsilon}\overline{w}(n) > \theta + \hat{\eta}\,, \forall n \tag{4.19}$$

then a unique solution to (4.13) exists. Likewise, if C.3, C.4, and (4.19) hold and $\overline{w}(n)$ is a constant, then a unique solution to (4.15) exists.

Proof:

By virtue of (4.19), and C.4:

$$\eta(\Lambda = \tau_\varepsilon) + N\varepsilon(\Lambda = \tau_\varepsilon)\overline{w}(n) \tag{4.20}$$

$$= \eta(\Lambda = \tau_\varepsilon) + N\hat{\varepsilon}\overline{w}(n) > \eta(\Lambda = \tau_\varepsilon) + \hat{\eta} + \theta\,, \forall n$$

But C.3 states that $(\Lambda) \geq -\hat{\eta}$, therefore $\eta(\Lambda = \tau_\varepsilon) + \hat{\eta} \geq 0$, and (4.20) yields:

$$[\eta(\Lambda) + N\varepsilon(\Lambda)\overline{w}(n)]_{\Lambda=\tau_\varepsilon} > \theta\,, \forall n \tag{4.21}$$

By C.2 $\eta(\mathrm{t})$ and $\varepsilon(\mathrm{t})$ are continuous, and by C.3 and C.4, $\eta(0) + N\varepsilon(0)\overline{w}(n) = 0$. Therefore $\eta(\Lambda) + N\varepsilon(\Lambda)\overline{w}(n)$ must cross θ from below at least once. Thus there exists a minimum value of Λ for which this occurs, and by definition this is the solution to (4.13). Obviously a similar argument shows the existence of a unique solution to (4.15) in the case that $\overline{w}(n)$ is a constant.

Comment: The existence of synchronous and periodic firing is widely observed in spiking neural nets. The above results are only the simplest manifestation of the phenomenon. For example, one could have separate groups of neurons, each having a different value of $\sum_j w_{ij}^f$ or $N\overline{w}$, and thus firing at different frequencies. Relatively sparse connections among such groups can result in all manner of interesting interference effects, but we do not pursue this subject further here.

Comment: (4.12) and (4.14) are referred to as *homeostasis* conditions in the literature. They express the idea that the chemical resources available to transmit signals to a given neuron through all the incoming synapses are limited. Such conditions are well known to stabilize synaptic plasticity and prevent runaway growth in the synaptic efficacies. Note that the above result shows it is the 1-norm of the vector of incoming synaptic efficacies that should be used. In addition, as will be explored later, homeostasis provides a competitive mechanism that concentrates efficacy in relatively few synapses and speeds learning. The importance of the above two theorems is that synchrony and periodicity, phenomena that are ubiquitously observed in spiking nets, ***imply*** homeostasis.

4.4 Implications of Homeostasis

The next question is: Does homeostasis in the form (4.14) imply the stable convergence of the system to synchrony and periodicity? Until further notice, the simpler condition (4.14) is assumed, along with relation (4.15).

Suppose that the w_{ij}^{f} are constant and system (4.11) is initially constrained (perhaps by external stimulation) so that its neurons fire at slightly different times. That is, for $n = 0$:

$$T_i(0) = \gamma_i(0)\,, i = 1, \dots, N$$
$$\max_i\{|\gamma_i(0)|\} \leq \bar{C} \tag{4.22}$$

where no two firing times are equal. Also, assume that $\bar{w}$ is large enough that all neurons will fire due to the post-synaptic response to the initial firing times given by (4.22).

Let us trace the evolution of the system. First, define the firing time at the initial time, $t = 0$, to be $\gamma_N(0)$ and arrange the $\gamma_i(0)$ in order of increasing size:

$$\gamma_1(0) < \gamma_2(0) < \dots < \gamma_N(0) = 0 \tag{4.23}$$

Note that we thus have:

$$|\gamma_1(0)| > |\gamma_2(0)| > \cdots > |\gamma_N(0)| = 0 \tag{4.24}$$

Because all neurons fire in response to the initial firing times, then for $n = 0$, equation (4.11) becomes:

$$T_i(1) - T_i(0) = min\left\{\Lambda > 0 : \theta = \eta(\Lambda) + \sum_{i \geq i}^{N} w_{ij}^{f}\varepsilon\left(\Lambda + \gamma_i(0) - \gamma_j(0)\right)\right\} \tag{4.25}$$

In view of $\max_i\{|\gamma_i(0)|\} \leq \bar{C}$, let us define:

$$\Lambda = \Lambda_0 + \Lambda_1$$
$$\Lambda_0 \cong O(\bar{C}) \tag{4.26}$$

$$\Lambda_1 \cong O\left(\max_i\{|\gamma_i(0)|\}\right) \leq \bar{C}$$

then substitute this into the condition $\theta = \eta(\Lambda) + \sum_{j\geq i}^{N} w_{ij}^f \varepsilon\left(\Lambda + \gamma_i(0) - \gamma_j(0)\right)$, expand in Taylor series, and neglect terms of order $(\max_i\{|\gamma_i(0)|\})^2$ or smaller, to get:

$$\theta = \eta(\Lambda_0) + \sum_{j\geq i}^{N} w_{ij}^f \varepsilon(\Lambda_0) \Longrightarrow \Lambda_0 = \bar{C} \tag{4.27}$$

$$\Lambda_1 = \frac{\sum_{j>i}^{N} w_{ij}^f \left(\gamma_j(0) - \gamma_i(0)\right)}{\sum_{j\geq i}^{N} w_{ij}^f + {\eta'(\bar{C})}\Big/{\varepsilon'(\bar{C})}}$$

Since $(\max_i\{|\gamma_i(0)|\}) \leq \bar{C}$, $\Lambda = \Lambda_0 + \Lambda_1$ with Λ_0 and Λ_1 given by (4.27) is still the minimum time needed to reach the threshold. Hence (4.25) becomes:

$$T_i(1) = \bar{C} + \gamma_i(1) \tag{4.28}$$

$$\gamma_i(1) = \frac{\sum_{j>i}^{N} w_{ij}^f \gamma_j(0) + \left(w_{ii}^f + {\eta'(\bar{C})}\Big/{\varepsilon'(\bar{C})}\right)\gamma_i(0)}{\sum_{j\geq i}^{N} w_{ij}^f + {\eta'(\bar{C})}\Big/{\varepsilon'(\bar{C})}}$$

To progress further, we need some elementary results concerning non-negative and positive matrices. A vector or matrix is non-negative (positive) if all its elements are non-negative (positive). If $\boldsymbol{M} \in \mathbb{R}^{N\times L}$ then the non-negative matrix formed by replacing each element by its absolute magnitude is denoted by $|\boldsymbol{M}|$. $\mathbb{R}_+^{N\times L}$ denotes the field of $N \times L$ positive or non-negative matrices. If $\boldsymbol{S}$ and $\boldsymbol{R}$ are two non-negative or positive matrices having the same row and column dimensions then the double inequality notation $\boldsymbol{S} \ll \boldsymbol{R}$, or $\boldsymbol{S} \leq\leq \boldsymbol{R}$ indicates element-by-element inequality. Denote by $\boldsymbol{u} \in \mathbb{R}_+^N$

the positive vector all of whose elements are unity. This is a vector of unit norm in the infinity norm. If $v \in \mathbb{R}^N$ is some vector, then:

$$\|\boldsymbol{v}\|_\infty = max\{|v_1|, |v_2|, \dots, |v_N|\} \tag{4.29}$$

Obviously, $\|\boldsymbol{u}\|_\infty = 1$. We have the following simple result:

Lemma 1:

Suppose $\boldsymbol{v} \in \mathbb{R}^N$, $\widehat{\boldsymbol{v}} \in \mathbb{R}^N$ and $\boldsymbol{x} \in \mathbb{R}_+^N$:

$$|\boldsymbol{v}| \leq\leq \widehat{\boldsymbol{v}} \tag{4.30}$$

$$\widehat{\boldsymbol{v}} = \boldsymbol{M}\boldsymbol{x}$$

where $\boldsymbol{M} \in \mathbb{R}_+^{N\times N}$ is upper triangular, has no zero upper triangular elements, and has all of its row sums equal to unity. Further, the elements of $\boldsymbol{x} \in \mathbb{R}_+^N$ are arranged in order of decreasing magnitude with $x_N = 0$:

$$x_1 \geq x_2 \geq \cdots \geq x_N = 0 \tag{4.31}$$

Then:

$$\|\boldsymbol{v}\|_\infty < \|\boldsymbol{x}\|_\infty \tag{4.32}$$

Proof:

From (4.30b) the first element through $N-2$ elements of $\widehat{\boldsymbol{v}}$ are given:

$$\hat{v}_i \leq \sum_{m=i}^{N-1} M_{im}x_m \,, i = 1, \dots, N-2 \tag{4.33}$$

Since all elements of $\boldsymbol{x}$ are less than or equal to $x_1 = \|\boldsymbol{x}\|_\infty$ and the row sums of $\boldsymbol{M}$ are unity, (4.33) gives:

$$\hat{v}_i \leq \sum_{m=i}^{N-1} M_{im} x_1 < x_1 = \|\boldsymbol{x}\|_\infty , i = 1, \ldots, N-2 \qquad \textbf{(4.34)}$$

where the positivity of all upper triangular elements of $\boldsymbol{M}$ has been used. For the $(N-1)^{th}$ and N^{th}elements of $\hat{\boldsymbol{v}}$, we have that $\hat{v}_N = 0$, and:

$$\hat{v}_{N-1} = M_{N-1,N-1} x_{N-1} < x_{N-1} < \|\boldsymbol{x}\|_\infty \qquad \textbf{(4.35)}$$

Thus all elements of $\hat{\boldsymbol{v}}$ are less than $\|\boldsymbol{x}\|_\infty$. Therefore $\|\boldsymbol{v}\|_\infty < \|\hat{\boldsymbol{v}}\| < \|\boldsymbol{x}\|_\infty$ and (4.32) follows. Collecting together the above results, we can prove at least a limited statement regarding the asymptotic stability of the synchronous, periodic firing condition.

Theorem 3:

Assume condition (4.14) and that the w_{ij}^f are constant. Suppose system (4.11) is initially constrained so that its neurons fire at slightly different times, i.e.:

$$T_i(0) = \gamma_i(0) , i = 1, \ldots, N \qquad \textbf{(4.36)}$$
$$\max_i\{|\gamma_i(0)|\} \leq \bar{C}$$

where the $\gamma_i(0)$ are ordered as in (4.23) and (4.24). Let $\bar{w}$ be large enough that all neurons will fire due to the post-synaptic response to the initial firing times given above. Assume also, that $\max_i\{|\gamma_i(0)|\}$ is sufficiently small that (4.29) yields an approximate estimate of the next set of firing times of acceptable accuracy. Then:

(a) $\max_i\{|\gamma_i(1)|\} < \max_i\{|\gamma_i(0)|\}$ and to the same or better accuracy of approximation and for all $n = 0,1, \ldots$:

$$T_i(n+1) = (n+1)\bar{C} + \gamma_i(n+1) \tag{4.37}$$

$$\gamma_i(n+1) = \frac{\sum_{j>i}^{N} w_{ij}^f \gamma_j(n) + \left(w_{ii}^f + {\eta'(\bar{C})}\Big/{\varepsilon'(\bar{C})} \right) \gamma_i(n)}{\sum_{j\geq i}^{N} w_{ij}^f + {\eta'(\bar{C})}\Big/{\varepsilon'(\bar{C})}}$$

where for each n the $\gamma_i(n), i = 1, \ldots, N$ are first ordered as In (4.23).

(b) Let $\gamma(n) \in \mathbb{R}^N = [\gamma_1(n), \gamma_2(n), \ldots \gamma_n(n)]^T$. For all $n = 0,1, \ldots$:

$$\|\gamma(n+1)\|_\infty < \|\gamma(n)\|_\infty \tag{4.38}$$

(c) As $n \to \infty, \gamma(n)$ converges to 0. Hence the firing is ultimately synchronous and periodic.

Proof:

(a) The w_{ij}^f are all positive, and by C.3, $\eta(t)$ is monotonically increasing so the term $\eta'(\bar{C})/\varepsilon'(\bar{C})$ is positive. Then (4.37b) yields:

$$|\gamma_i(1)| \leq \frac{\sum_{j>i}^{N} w_{ij}^f |\gamma_j(0)| + \left(w_{ii}^f + {\eta'(\bar{C})}\Big/{\varepsilon'(\bar{C})} \right) |\gamma_i(n)|}{\sum_{j\geq i}^{N} w_{ij}^f + {\eta'(\bar{C})}\Big/{\varepsilon'(\bar{C})}} \tag{4.39}$$

Letting $\gamma(n) \in \mathbb{R}^N = [\gamma_1(n), \gamma_2(n), \ldots \gamma_n(n)]^T$, this has the matrix form:

$$|\gamma(1)| \leq \boldsymbol{\Psi} |\gamma(0)| \tag{4.40}$$

$$\boldsymbol{\Psi}_{ij} = \begin{cases} \dfrac{\left[\left(\eta'(\bar{C})\Big/_{\varepsilon'(\bar{C})}\right)\delta_{ij} + w_{ij}^{f}\right]}{\sum_{k\geq i}^{N} w_{ik}^{f} + \eta'(\bar{C})\Big/_{\varepsilon'(\bar{C})}}, j \geq i \\ 0, j < i \end{cases}$$

Thus $\boldsymbol{\Psi}$ is upper triangular, all its upper triangular elements are positive and its row sums are unity. Moreover, since the elements of $|\gamma(0)| \in \mathbb{R}^N = [|\gamma_1(0)|, |\gamma_2(0)|, \ldots, |\gamma_n(0)|]^T$ are ordered as are the elements of $\boldsymbol{x} \in \mathbb{R}_+^N$ in (4.31), all the conditions of Lemma 1 are met. Therefore

$\max_i\{|\gamma_i(1)|\} < \max_i\{|\gamma_i(0)|\}$, or $\|\gamma(1)\|_\infty < \|\gamma(0)\|_\infty$.

Since the bounds on the elements of $\gamma(1)$ are tighter than those imposed on $\gamma(0)$, to the same or better degree of approximation as in the formulation of (4.29), we can obtain the expression for $\gamma(2)$ in terms of $\gamma(1)$ by ordering $\gamma(1)$ as in (4.23), substituting $\gamma(1)$ for $\gamma(0)$ and repeating the arguments leading from (4.23) to (4.37). In like manner, this process can be repeated *ad infinitum* to arrive at (4.37).

(b) Equation (4.37b) yields:

$$|\gamma_i(n+1)| \leq \frac{\sum_{j>i}^{N} w_{ij}|\gamma_j(n)| + \left(w_{ii} + \eta'(\bar{C})\Big/_{\varepsilon'(\bar{C})}\right)|\gamma_i(n)|}{\sum_{j\geq i}^{N} w_{ij} + \eta'(\bar{C})\Big/_{\varepsilon'(\bar{C})}} \qquad \textbf{(4.41)}$$

Again, using the notation $(n) \in \mathbb{R}^N = [\gamma_1(n), \gamma_2(n), \ldots \gamma_n(n)]^T$, this has the matrix form:

$$|\gamma(n+1)| \leq \boldsymbol{\Psi}|\gamma(n)| \tag{4.42}$$

where $\boldsymbol{\Psi}$ is given in (4.40b). As in part (a), all the conditions of Lemma 1 are met. Therefore, (4.38) follows.

(c) Via the above results, we have created an infinite sequence, $\{\|\gamma(n)\|_\infty, n = 0, \ldots, \infty\}$, for which $\|\gamma(n+1)\|_\infty < \|\gamma(n)\|_\infty$ for every $n \geq 0$. Furthermore, since $\gamma(n) \in \mathbb{R}_+^N \; \forall \; n \geq 0$, the sequence is bounded by zero. Then by the monotone convergence theorem, the sequence converges to $\inf_n\{\|\gamma(n)\|_\infty\} = 0$, so that the Theorem 3 is proved, as shown by Bibby in [32].

4.5 Learning Binary Sequences

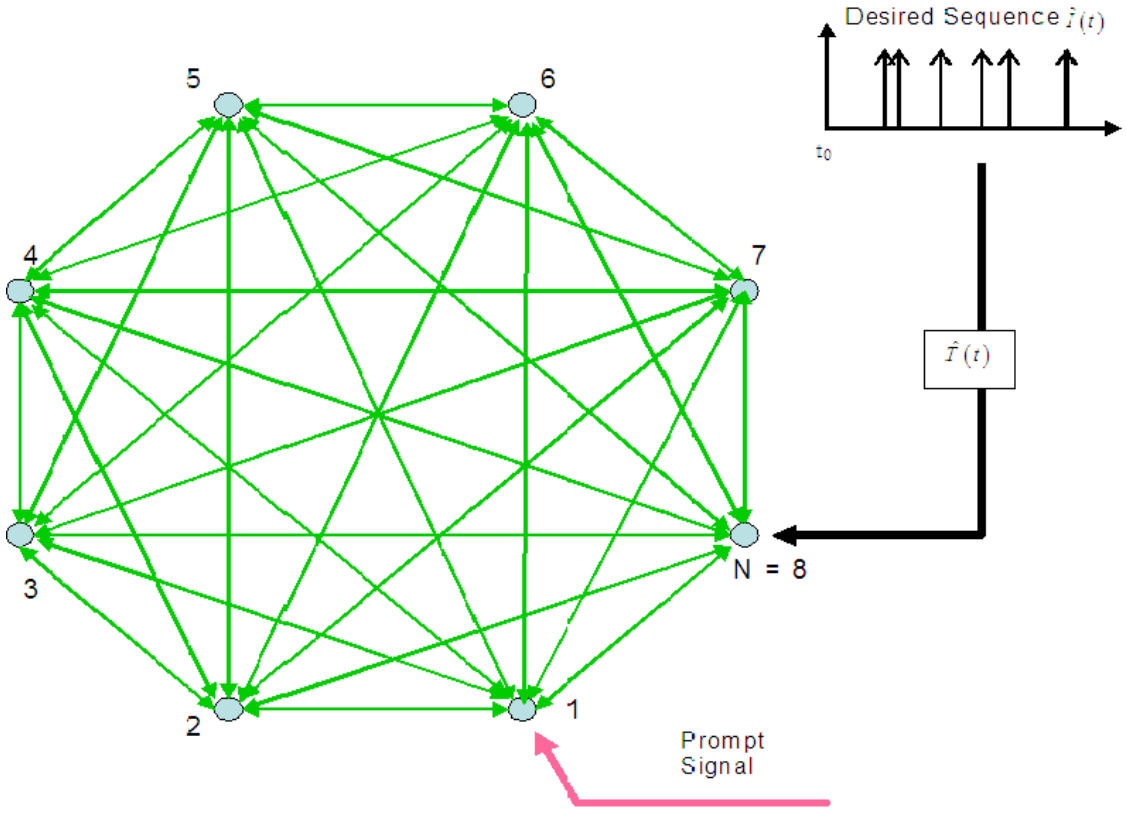

Figure 4.1. Depiction of neuron cluster with all-to-all connections, a prompt signal input to an arbitrary neuron (in this case neuron 1), and a training signal input to an arbitrary neuron (in this case neuron 8).

Suppose we have an N neuron network with all-to-all coupling (see Figure 4.1) whose spiking times are governed by (4.11), with the homeostasis constraint (4.14) and

the bound (4.19) with $\bar{w}$ a constant such that the network spikes periodically with period $\bar{C}$ satisfying (4.15).

It is assumed here that prior to the onset of training, all synaptic efficacies are positive. It is desired to train the network to be able to duplicate a binary sequence when prompted by an initiation or prompting signal applied to neuron 1, and starting the network from a quiescent state. Let the prompt signal be an impulse at time $t = 0$ and of strength $\bar{W} = N\bar{w}$, sufficiently large that it causes neuron 1 to fire at $t = \bar{C}$.

The desired binary sequence is a spike train produced by neuron 'N' having amplitudes either zero or unity. We assume that the sequence to be learned has spikes that are periodic and in synchrony with the periodic spiking of the network. In other words, if $t = 0$ is the time of the prompt signal, then the spiking sequence that is desired from neuron N has the form:

$$\hat{I}(t) = \sum_{m=1}^{M} I_m \delta(t - m\bar{C}), \qquad I_m = 0 \text{ or } 1 \text{ for } m = 1, \dots, M \tag{4.43}$$

where M is the total length of the binary sequence and the spike train begins at $t = \bar{C}$. However, the training signal has the form:

$$\hat{T}(t) = \bar{W} \sum_{m=1}^{M} T(m)\delta(t - m\bar{C})$$
$$T(m) = 2(I_m - \tfrac{1}{2}) \tag{4.44}$$

where the reason for the form of $\hat{T}(m)$ will become apparent below.

One first has to discover how the synaptic weights can evolve while satisfying (4.14). Consider the post synaptic neuron i. Define the non-negative matrix $\boldsymbol{W}_i \in \mathbb{R}_+^N$

having $w_{i,j}^{f}; j = 1, \dots, N-1$ as its j[th] element. Defining a vector of $\boldsymbol{u} \in \mathbb{R}_{+}^{N}$, composed of ones, conditions (4.14) take the form:

$$\boldsymbol{W}_i^T(n)\boldsymbol{u} = N\bar{w} \tag{4.45}$$

where as indicated, we assume that the synaptic weight is constant over a given time period between spikes but is a function of the number of periods, n, that have elapsed since the first prompt signal. Let us assume that $\boldsymbol{W}_i(n+1)$ is fully determined by its previous value, (n) , and by the past spiking times. Then the time evolution is characterized by a relation for its increment over two successive periods. (4.45) immediately implies:

$$\boldsymbol{u}^T[\boldsymbol{W}_i(n+1) - \boldsymbol{W}_i(n)] = 0 \tag{4.46}$$

Further. (4.45) yields:

Lemma 2:

$\boldsymbol{W}(t)$ satisfying (4.46) has the form:

$$\boldsymbol{W}_i(n) = N\bar{w}\boldsymbol{D}(n)\boldsymbol{f}(n) \tag{4.47}$$

$$\boldsymbol{D}(n) = {}^{1}/_{\sum_{l=1}^{N} f_l}$$

for any $\boldsymbol{f}(t) \in \mathbb{R}^{NxN}$ such that no row of $\boldsymbol{f}(t)$ is orthogonal to $\boldsymbol{u}$.

Proof:

Obviously, from (4.45), a necessary condition is that $\boldsymbol{W}(t)$ cannot be orthogonal to $\boldsymbol{u}$, so the same must be true for $\boldsymbol{f}(n)$. The rest is pure tautology since:

$$W_i^T \boldsymbol{u} = N\bar{w}\boldsymbol{D}(t)\boldsymbol{f}^T(t)\boldsymbol{u} = N\bar{w}\boldsymbol{D}(t)\left[\sum_{l=1}^{N} f_l\right] = N\bar{w} \tag{4.48}$$

Thus we obtain the stated result.

Now there is the question concerning the choice of $\boldsymbol{f}(n)$. $\boldsymbol{f}(n)$ can be *any* vector with rows not orthogonal to $\boldsymbol{u}$; including $\boldsymbol{W}_i(n)$ itself. Indeed, the only quantities entering into the problem are $\boldsymbol{u}$, $\boldsymbol{W}_i(n)$, and, possibly, the spike time-dependent plasticity associated with each individual synapse, so $\boldsymbol{f}(n)$ must be a function of one of these three or a combination. If $\boldsymbol{f}(n)$ were $\boldsymbol{u}$ then $\boldsymbol{W}_i(n+1) = \bar{w}$, which is untenable, since there would be no synaptic plasticity. Then, $\boldsymbol{f}(n)$ must be $\boldsymbol{W}_i(n)$, so that (4.47) becomes:

$$\boldsymbol{W}_i(n+1) = N\bar{w}\boldsymbol{W}_i(n)/\boldsymbol{u}^T\boldsymbol{W}_i(n) \tag{4.49}$$

Note that this expression would preclude any synaptic plasticity in the event that only one presynaptic neuron fires, for in that case $\boldsymbol{W}_i(n)$ has only a single element so that $\boldsymbol{W}_i(n+1) = N\bar{w}$. Can (4.49) be altered to be consistent with spike time-dependent plasticity? By the latter (see Figure 4.2) we mean the behavior noted from experimental observations of single pairs of neurons and their connecting synapse [33]-[36]. If the presynaptic neuron fires a short time (approximately 40 ms) before the postsynaptic neuron then the efficacy is strengthened. If the postsynaptic neuron fires shortly *before* the presynaptic neuron, the efficacy is reduced. Finally if the firings are simultaneous or widely separated, there is no change. This behavior is summarized in the so-called *Learning Window*, $\hat{F}(\hat{t}_j - \hat{t}_i)$.

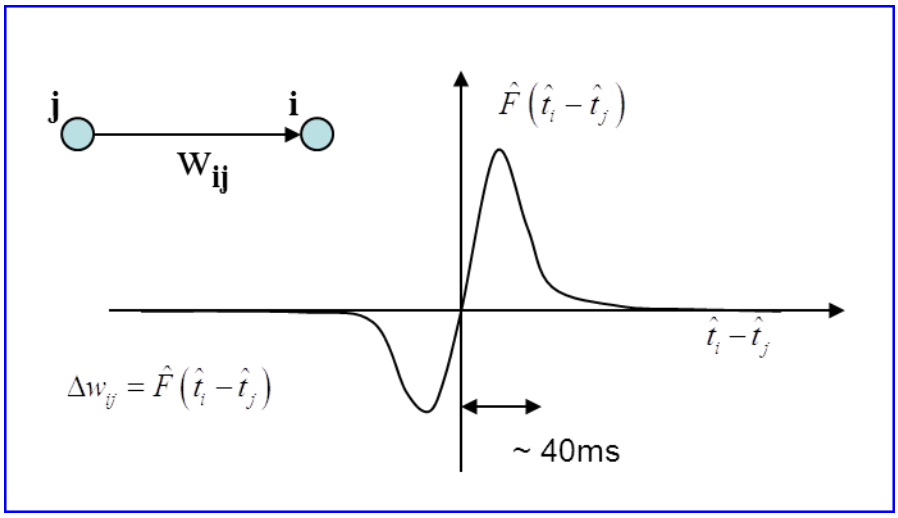

Figure 4.2. Learning Window, $\hat{F}(\hat{t}_j - \hat{t}_i)$, presents experimentally obtained information concerning neuron pair interaction.

This might take the form:

$$\hat{F}(\hat{t}_j - \hat{t}_i) = \frac{1}{\hat{\tau}}(\hat{t}_i - \hat{t}_j)\hat{h}\exp\left[1 - \left|\hat{t}_i - \hat{t}_j\right|/\hat{\tau}\right] \qquad \textbf{(4.50)}$$

where $\hat{h}$ is the maximum magnitude of $\hat{F}(\hat{t}_j - \hat{t}_i)$, and $\hat{\boldsymbol{\tau}}$ is the value of $\left|\hat{t}_i - \hat{t}_j\right|$ at which this occurs.

It is clear that if the above plasticity model is correct for single synapses, then (4.49) must be modified so that the total efficacy, $N\overline{w}$, is no longer constant. In particular, the model in Figure 4.2 must be recovered whenever any one of the presynaptic neurons fires alone. Letting $\boldsymbol{W}_{i,j}(n)$ denote the jth element of $\boldsymbol{W}_i(n)$, the aforementioned condition implies:

$$\Psi_{ij}(n) = \Psi_{ij}(n-1) + \hat{F}(\hat{t}_j - \hat{t}_i) \qquad \textbf{(4.51)}$$

$$\boldsymbol{W}_{ij}(n) = \Psi_{ij}(n)\boldsymbol{W}_{ij}(n-1)/\boldsymbol{u}^T\boldsymbol{W}_i(n-1)$$

But now (4.45) is not satisfied, and moreover, if the total efficacy is time-varying, then the network will not spike periodically. It follows that the plasticity rule, (4.50) or Figure 4.2, cannot be used.

Under the assumption that the network fires periodically, we conclude that *the total pre-synaptic efficacy feeding any one neuron is either* $\overline{W} = N\overline{w}$ *or some level that is too small for the postsynaptic neuron to fire*.

The above observation prompts the following hypothesis. Although the spike-time-dependent plasticity illustrated in Figure 4.2 is based upon much carefully collected data, it might be that the standard interpretation of this data confuses cause and effect, or rather confuses correlation with causation. The standard interpretation is that the firing of neuron i after neuron j *causes* the synaptic weight to increase; and, likewise the opposite situation causes the weight to decrease. Let us entertain the opposite interpretation. **The firing of neuron j stimulates a higher concentration of neural-transmitters (represented by the weight, w_{ij}), at the gap of neuron i, inducing it to fire. Likewise the previous firing of neuron i depletes the neural transmitter concentration (w_{ij} decreases) linking it to the pre-synaptic neuron.**

Under the above interpretation, learning occurs as a consequence of the stimulation of neural transmitter supply by the forced firing of neurons under training, together with the depletion of neural transmitter to related, but not directly stimulated neurons, such that the homeostasis condition required for periodic firing is maintained at

every step. This picture permits us to confine attention to the simplest possible synaptic weight adjustment, i.e. each weight is changed to either zero or $\bar{W}$. In the following section, we present a learning algorithm of this form, by which the network of Figure 4.1 can organize itself to learn and store binary temporal sequences.

5. SYNCHRONOUS AND PERIODIC FIRING NEURON CLUSTER

Continuing from the previous section's theoretical setup of an all-to-all connected network of neurons, a computer model was developed using MatLab to simulate a simplified neuron cluster which, after training, will output the learned binary sequence when prompted. The major simplification in this simulation is that while in the previous section's example, the neuron cluster was designed for all-to-all coupling, in this experiment, connections converge toward a single neuron. For this introduction, the central neuron is Neuron 8, as shown in the figure below.

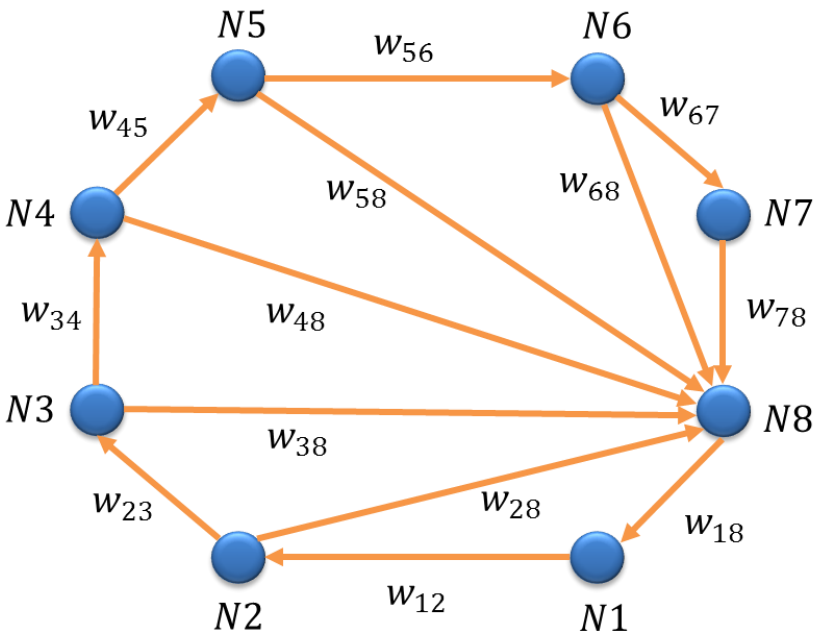

Figure 5.1. Neuron 8 cluster connection structure.

The simulation is composed of two sections: the learning portion and the testing portion. During the learning portion, the weights are incremented according to the learning rule until they have learned the given binary sequence. After the weights are randomly initialized to values between 0 and 1, the binary sequence is input to Neuron 1 (N1) during the first period then to Neuron 8 (N8) two periods later. See Figure 5.1. The offset in the timing between the inputs to N1 and N8 when combined with the learning rule creates the learning capacity of the cluster.

As stated in the previous section the constraint of N8 to fire at the specified time then induces the weight changes in the neuron connections. The learning rule is as follows: a weight is increased when a downstream neuron, N8, fires in the subsequent time step and is decreased if a downstream neuron does not fire in the next time step. Over the training time the weights within the outer ring are either decreased or increased according to that rule, until those weights correspond to the binary sequence being input to N8. Because, as was shown in the previous section, the sum of the weights of the upstream firing neurons, $\sum_j w_{ij}^f$, must equal a constant, and as there is only a single upstream neuron leading into N8 at a given time step, the weight is set to the constant value, 1. Similarly, if N8 doesn't fire, then the particular presynaptic weight feeding into N8 is "pruned" away, or becomes 0.

5.1 Simulation Specifications and Constraints

The design specifications and constraints upon which the model was based are:

1. Limited coupling as shown in Fig. 5.1, as opposed to the all-to-all coupling, shown in IV-1.
2. Synchronous and periodic neuron firing with a constant spiking period,
3. Homeostasis, or the sum total weight value of all presynaptic connections leading to a given downstream neuron is conserved at each time step.

$$\sum_j \mathrm{w}_{ij}^f(n) = N\overline{w}^f(n) \tag{6.1}$$

4. Learning Rule: during training, a weight is increased to 1 when a downstream neuron fires in the time step after the upstream neuron, $n+1$, and is decreased to zero if a downstream neuron does not fire a time step later.

$$w_{8j}(n) = \begin{cases} 1, & if \begin{cases} N_j(n-1) = 1 \\ N_8(n) = 1 \end{cases} \\ 0, else \end{cases} \tag{6.2}$$

5. During training, a binary element from the training sequence will be input to the final neuron, N8, two time step after that element is input to the first neuron, N1.

$$\begin{aligned} input &= [\,1\,0\,0\,0\,0\,0\,] \\ training &= [\,1\,0\,0\,0\,0\,0\,] \end{aligned} \tag{6.3}$$

6. During testing, the weight values from training will be loaded, a single prompt will be input to the first neuron, and the final neuron will output the learned binary sequence.

$$prompt = [1] \tag{6.4}$$

7. During training, all weights will be randomly initialized to a value greater than 0 and less than or equal to 1.

5.2 Training Simulation 1

As stated, the simulation is composed of two parts, the training portion and the testing portion. In the training portion, all weights for the first two training epochs are initialized to random values between 0 and 1. An epoch is eight time steps: the time it takes for the binary sequence's effects to propagate through the neuron cluster. The first binary sequence to be input is

[1 0 0 0 0 0].

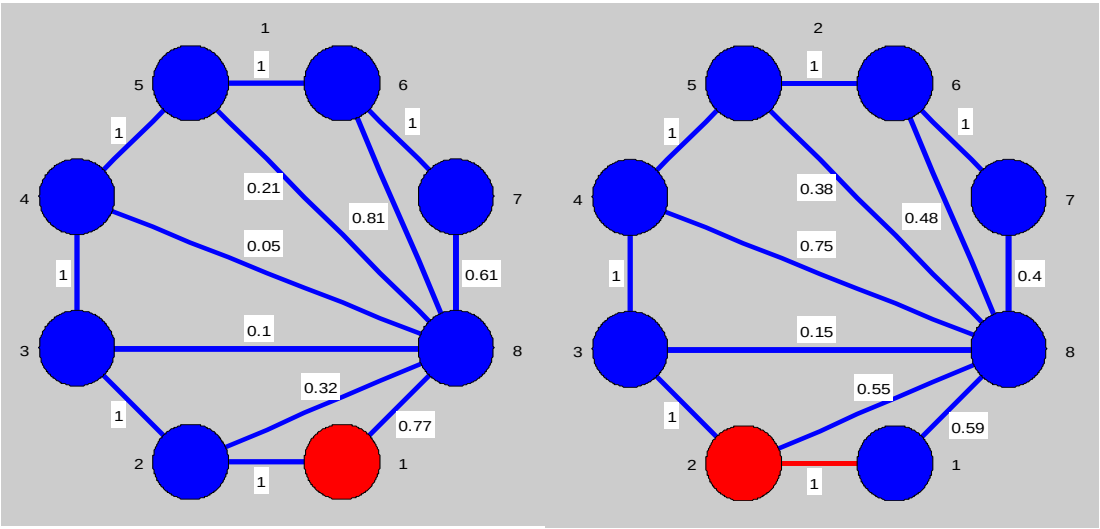

Figure 5.2. Left: Initial state of the N8 cluster at the first time step, and Right: N8 cluster at the second time step.

The following figures, starting with Fig. 5.2, show the response of the cluster to the binary training sequence at each time step. The weight values for each connection are displayed in the white boxes. At the first time step, the first element from the binary

sequence is entered into N1, causing it to fire, as indicated by the N1's red color. The synapse connecting N1 and N2 is shown as red to indicate firing has occurred. At time step two, N2 fires, induced by the firing of the first neuron.

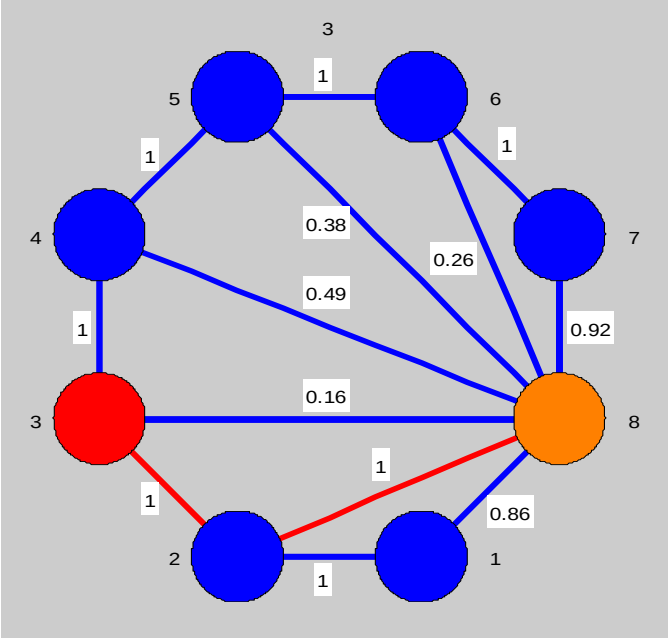

Figure 5.3. N8 cluster at time step three where N2 induced the firing of N8.

In the third time step, N3 and N8 fire, induced by the firing of N2 during the previous time step. Moreover, the first element of the binary sequence is entered into N8 which additionally causes N8 to fire. The fact that N8 fires due to the training input *and* due to an induced firing is depicted by N8 turning orange instead of red. This scenario

also means that the weight value for the connection from 1 to 8 is positively incremented for time step 3 during the next training epoch.

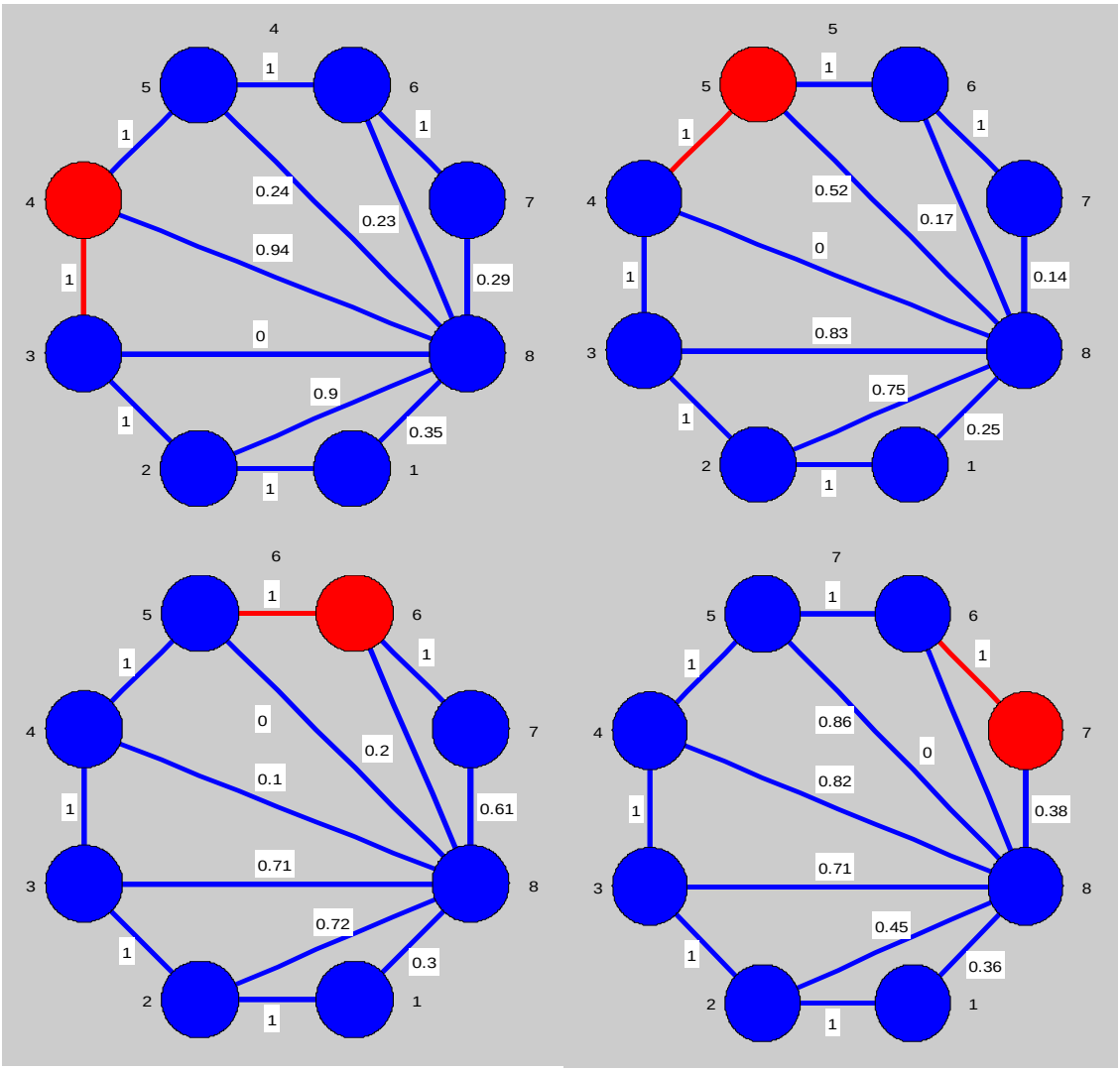

Figure 5.4. Final firings of the N8 cluster: firings of N4-N7.

Because all subsequent values of the binary sequence are zeros, there will be no other firings. Therefore, a cluster, when presented with this specific input, and after sufficient training, will only have a single connection to N8 remaining after training is complete. As shown in the above figures, the single spike propagates around the circumference of the cluster until N7. Due to the fact that the N7 to N8 connection is a data bearing connection and the input does not specify that there should be a nonzero connection there, N7 does not induce N8 to fire.

5.3 Test Simulation 1

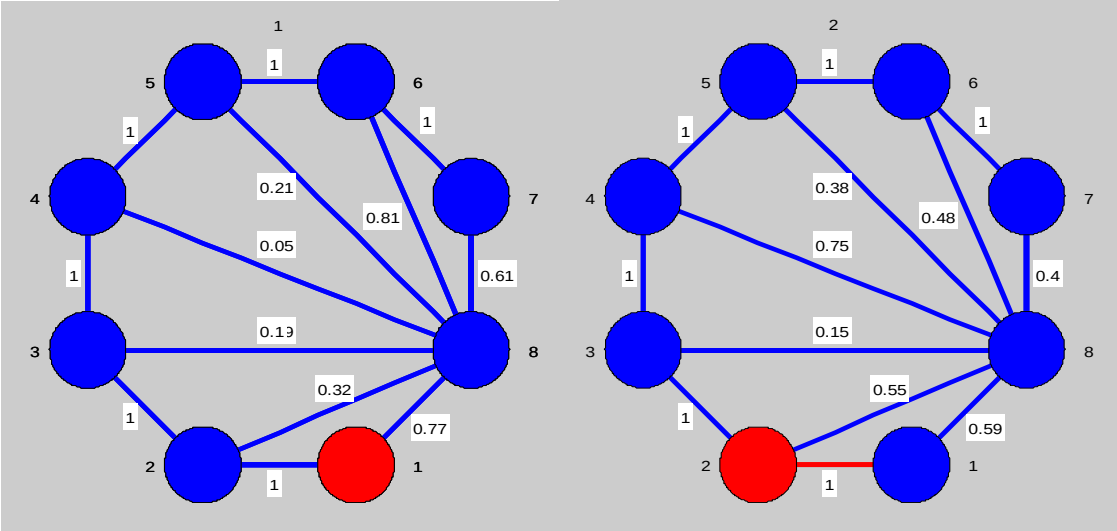

Figure 5.5. Initial state of the N8 cluster at the first time step (Left), and N8 cluster at the second time step (Right).

For the testing portion of the simulation, the weights for each connection at each time step of the final epoch (epoch 3 weights) of the training portion are loaded for use. The prompt, [1], is input to N1, causing it to fire as depicted in red above. In the second time step, N2 fires. In the third time step, N3 and N8 fire, induced by N2's firing during the previous time step. The remaining time steps in the testing portion follow the pattern of the training portion.

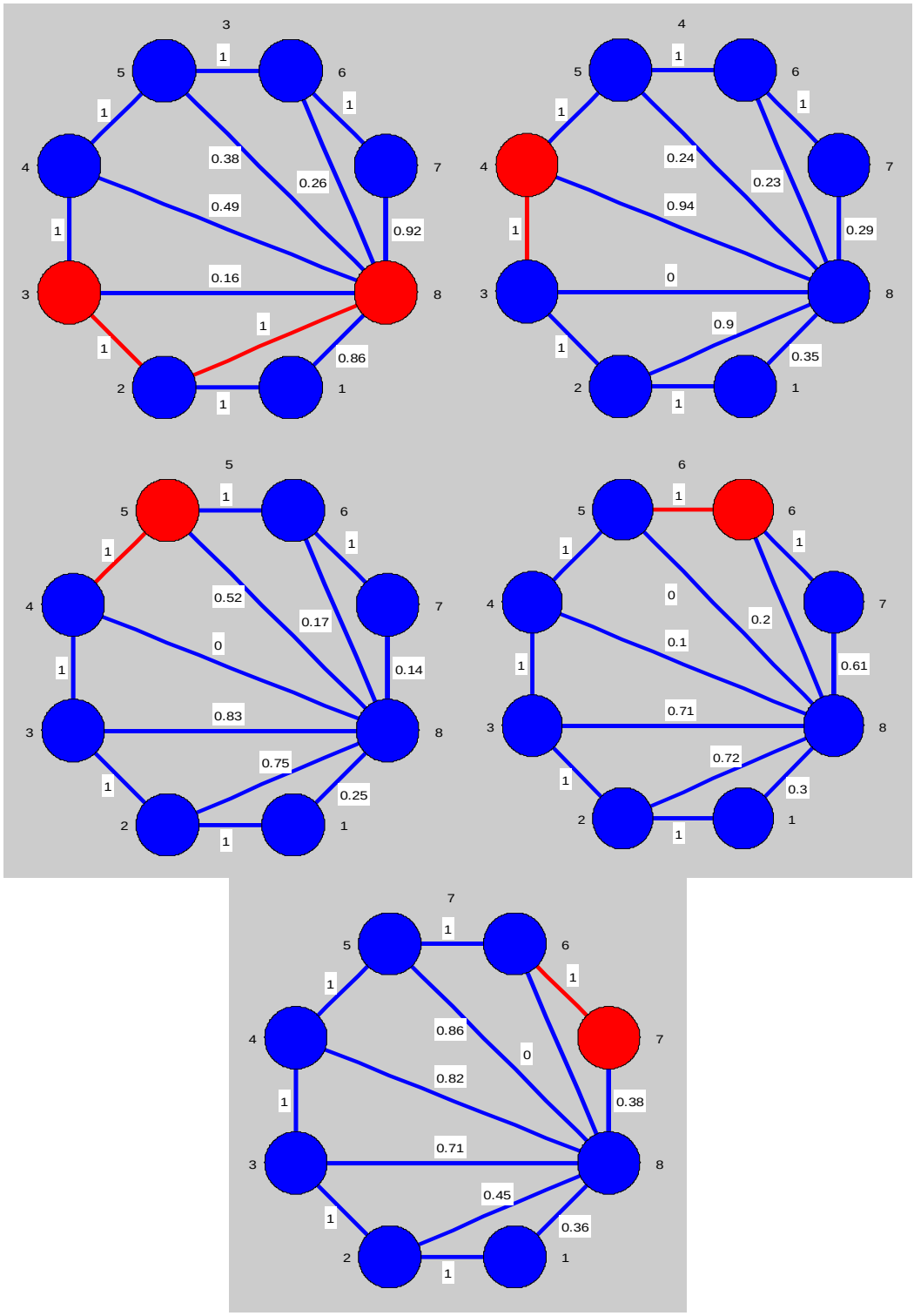

Figure 5.6. Final firings of the N8 cluster: firings of N3-N7, where N2 induced the firing of N8 during the third time step.

5.4 Analysis

Currently, after 3 epochs, the testing portion of the program returns a 0 for neurons that should not fire and 1 for the neurons that should fire over the eight time steps. In other words, for the binary sequence [1 0 0 0 0], the testing output is [1 0 0 0 0]. However, as the prompt is in actuality the same as the input sequence, an additional binary sequence, [1 0 1 1 0 1], was input to ensure the functionality of the framework, where the final bit is the termination signal.

5.5 Training Simulation 2

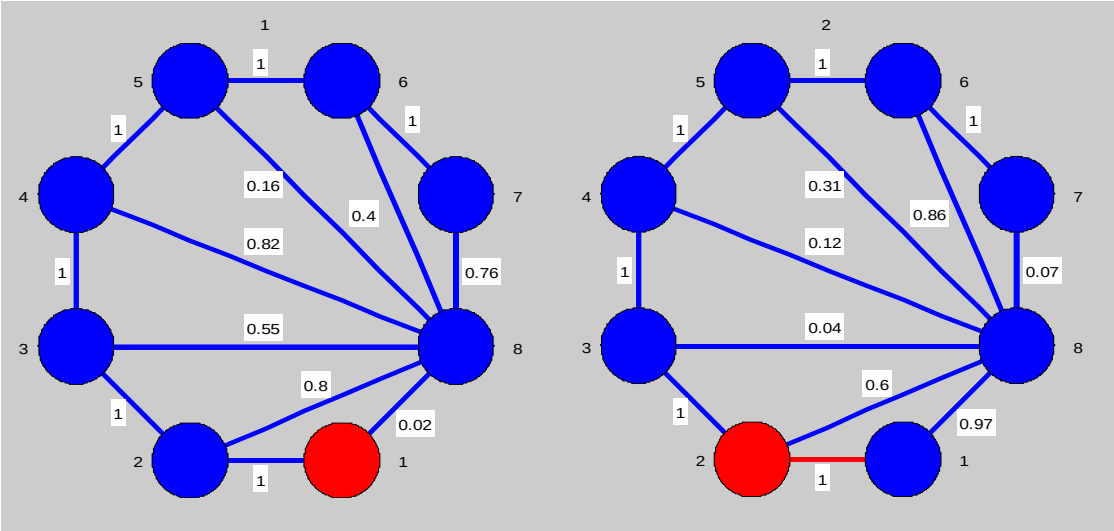

Figure 5.7. Initial state of the N8 cluster at the first time step (Left), and N8 cluster at the second time step (Right).

For the second simulation, the binary sequence to be learned is [1 0 1 1 0 1]. As seen in the figure above, the first input in the training sequence is entered into N1, inducing N2 to fire at the subsequent time interval.

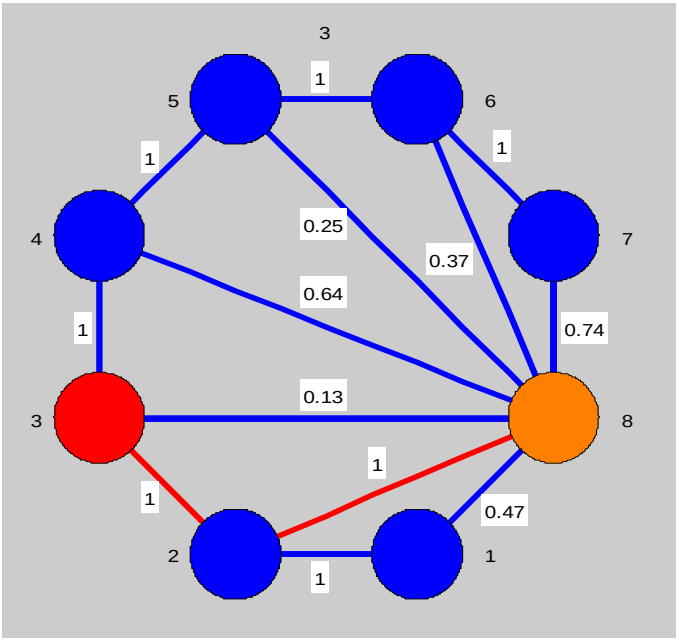

Figure 5.8. N3 firing and N8 induced by N2.

During the third time step, N3 and N8 fire, induced by N2. Again, N8 is depicted as orange because the training input induces N8 to fire during the same time step as one of the other seven neurons induces N8 to fire. In this case, that neuron was N2. The cumulative output at N8 is [1].

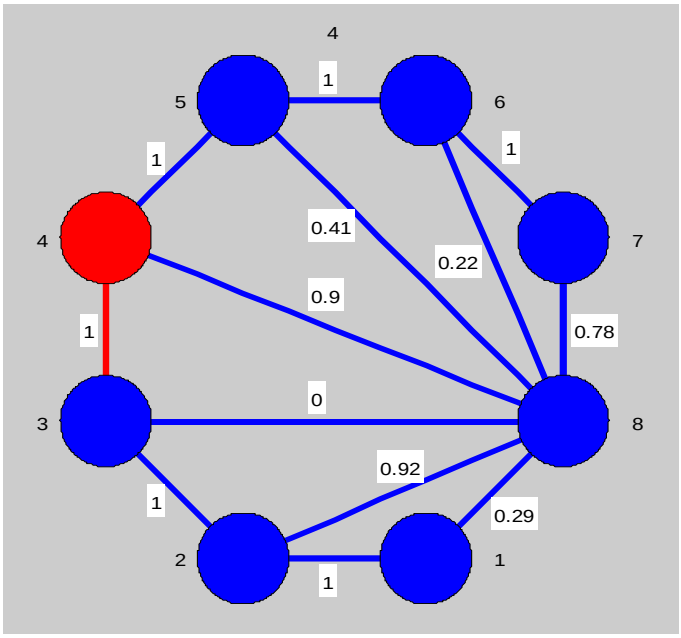

Figure 5.9. N4 firing.

In time step four, N4 fires due to N3 firing previously. Note the w_{83} connection has been "pruned," or decremented to zero, meaning N3 will not induce N8 to fire during the testing portion. The cumulative output at N8 is [1 0].

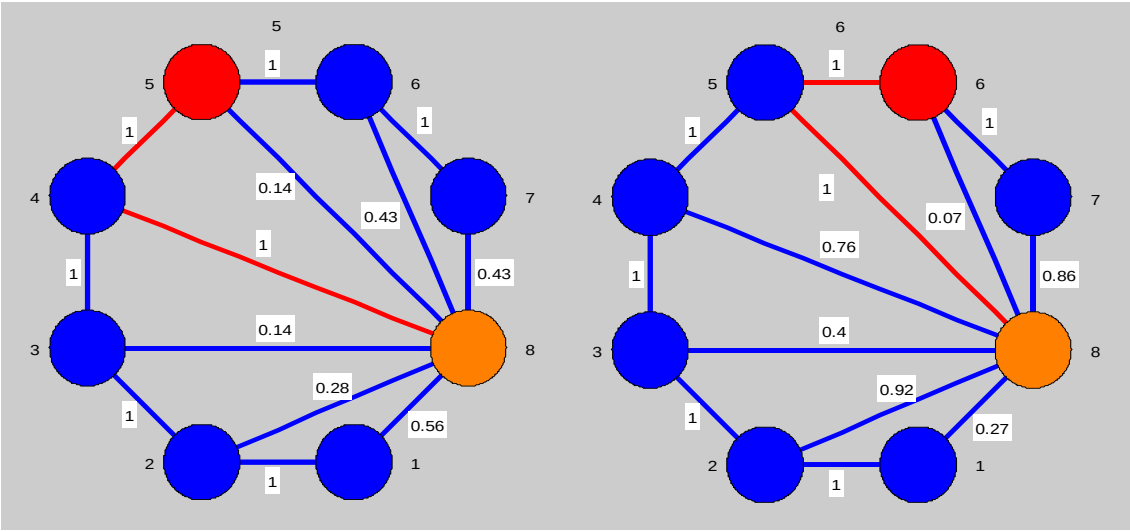

Figure 5.10. N5 firing and N8 firing due to N4 (Left), and N6 firing and N8 firing due to N5 (Right).

N4 has caused N5 and N8 to fire, and a training input also causes N8 to fire. Note w_{84} is stored as unity. The cumulative output at N8 is [1 0 1]. During time step six, N5 has caused N6 and N8 to fire, and a training input again causes N8 to fire. Note w_{85} is stored as unity. The cumulative output at N8 is [1 0 1 1].

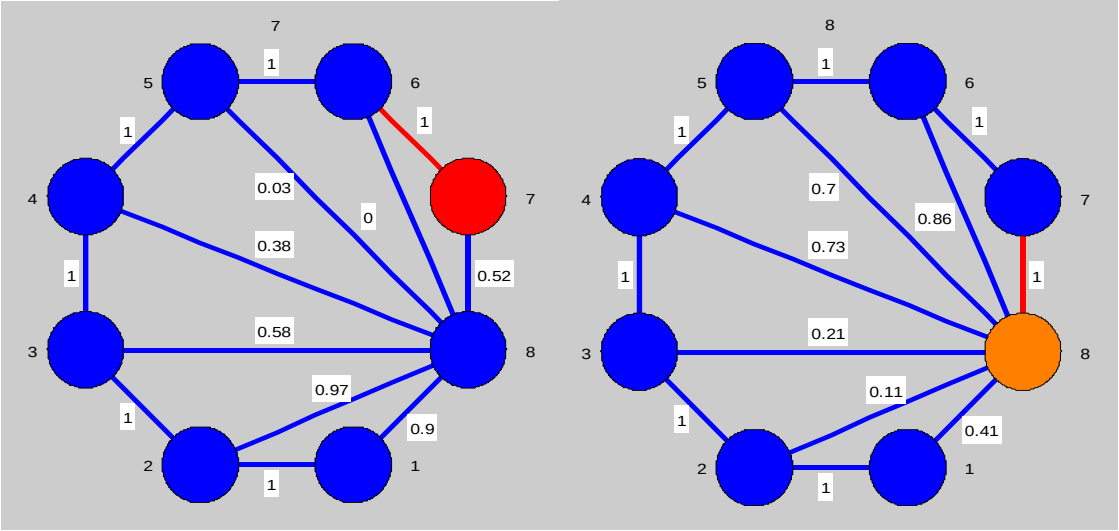

Figure 5.11. N7 firing (Left), and N8 firing induced by N7 (Right).

N6 induces N7 to fire, then in the next time step, N7 induces N8 to fire. In addition, another training input has been added to N8. The cumulative output at N8 is [1 0 1 1 0 1], where the final bit is the termination signal.

5.6 Testing Simulation

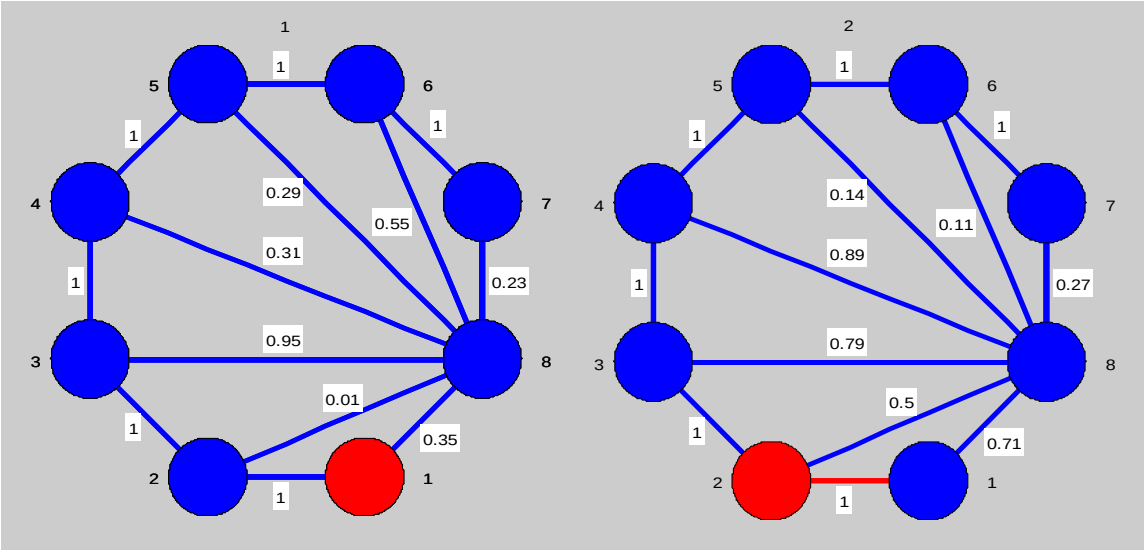

Figure 5.12. Initial state of the N8 cluster at the first time step (Left, and N8 cluster at the second time step (Right).

The time dependent weights from the training portion have been loaded. The prompt, [1], is introduced to N1 during the first time step, which causes N2 to fire in the subsequent time step.

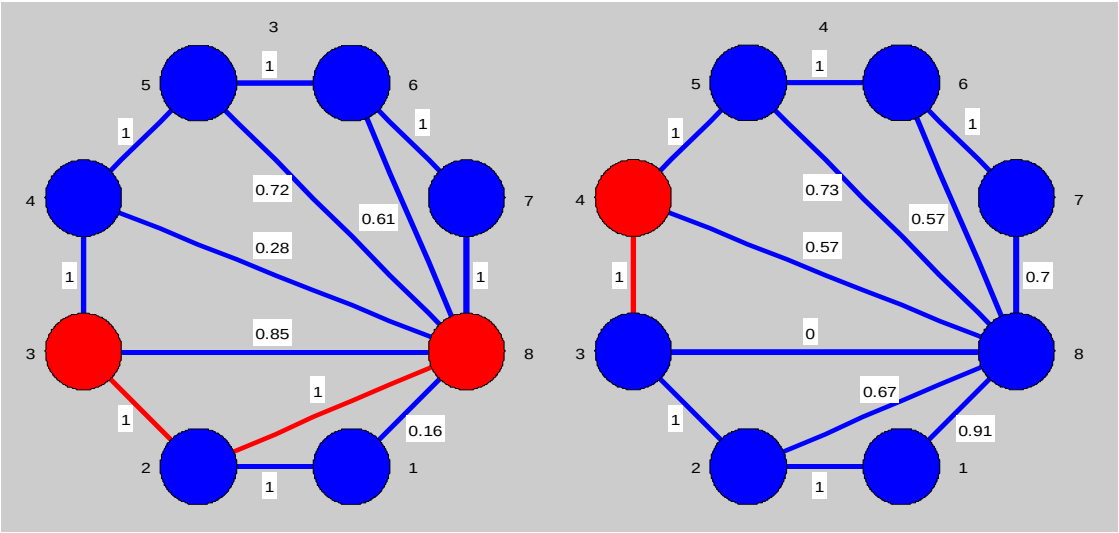

Figure 5.13. N3 firing and N8 firing due to N2 (Left), and N4 firing (Right).

For time step three, N2 induces N3 and N8 to fire because the training weight value for the w_{82} is 1. The cumulative output is [1]. During time step four, N3 induces the firing of N4. The cumulative output is [1 0].

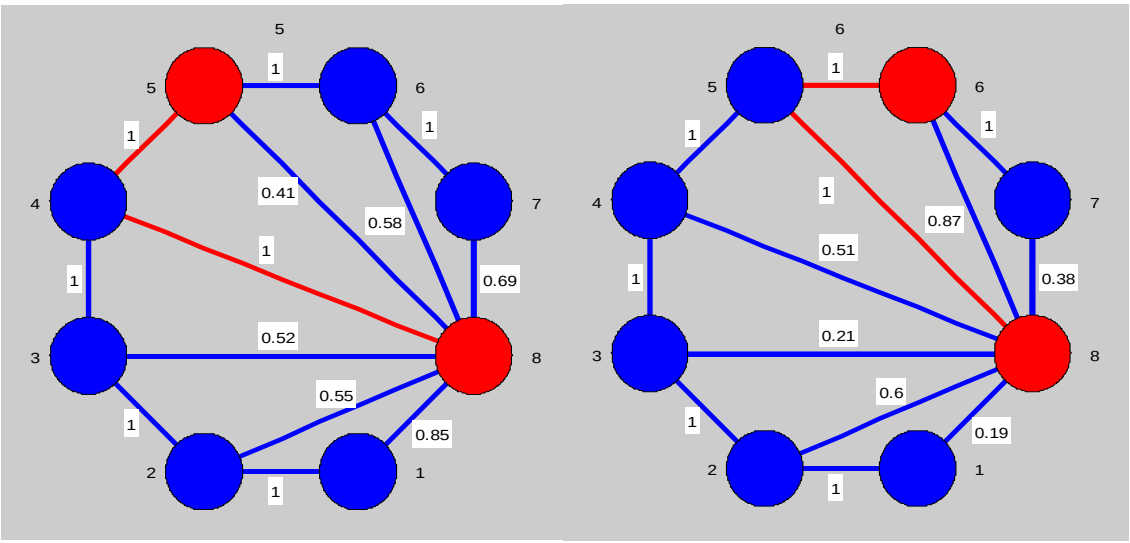

Figure 5.14. N5 firing and N8 firing due to N4 (Left), and N6 firing and N8 firing due to N5 (Right).

N4 induces the firing of N5 and N8 in time step 5. The cumulative output is [1 0 1]. Time step six shows the firing of N6 and N8. The cumulative output for both steps is [1 0 1 1].

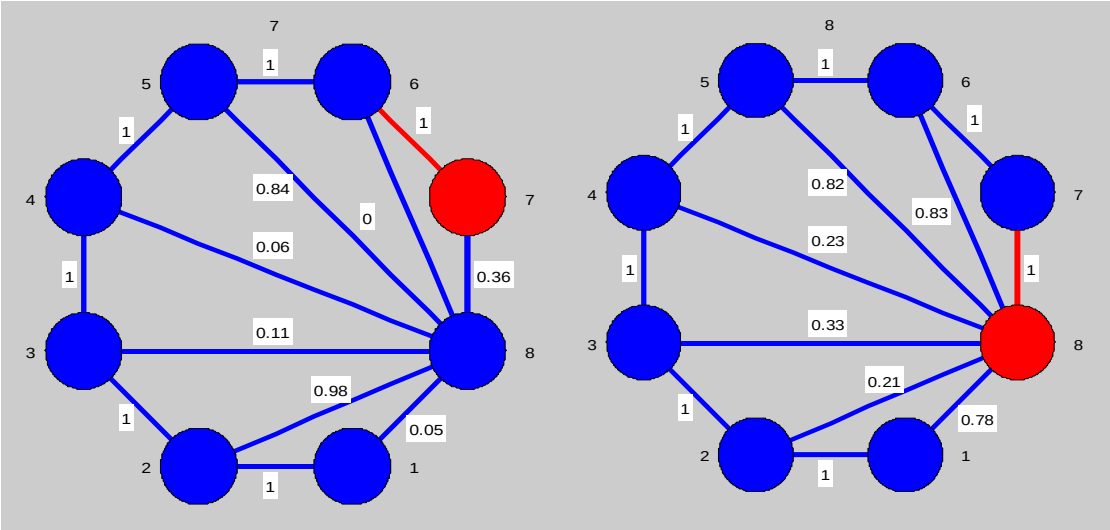

Figure 5.15. N7 firing (Left), and N8 firing (Right).

N6 induces the firing of N7 then N8 in the subsequent time step, yielding the cumulative output: [1 0 1 1 0 1].

5.7 Analysis 2

After 3 epochs of training, the testing portion of the program returns a 0 for neurons that should not fire and 1 for the neurons that should fire. For the input binary

sequence, the response from the prompt, [1], input at N1 is the output [1 0 1 1 0 1] at N8. The output is identical to the sequence the cluster was trained to learn.

6. FAST LEARNING NETWORKS

The previous section presented a specialized structure of the network theory from section four, where the peripheral connections, or "delay" lines, were "hardwired." On the other hand, this section presents an all-to-all network that starts with purely randomly weighted connections and constructs the previously demonstrated interconnection pattern seen in real time. This more general algorithm is capable of self-organizing in the manner that had been assumed from the start in the previous network.

6.1 Network Algorithm

Recall the all-to-all connected network featured in Figure 4.1. Suppose all the weights initially have non-negative, but arbitrary values less than $\bar{W}$. As mentioned above we start training with a prompt signal at neuron 1 at $t = 0$ causing it to fire at $t = \bar{C}$. The training signal, (4.44) begins at $t = \bar{C}$ and is input to neuron N. In the following, we represent the time by the number of the periods, n, elapsed. We postulate the following learning process, where steps A-C occur each firing period:

A. Consider the last neuron to fire (at the start it is neuron 1). Its stimulation builds a surge of neurotransmitter. The weight connecting it to neuron N is adjusted so that: it equals $\bar{W}$ if $T(n) = 1 (I_n = 1)$ (the training signal enhances the neural transmitter, thereby increasing the weight), or zero if $T(n) = -1 (I_n = 0)$ (the training signal is inhibitory, thus depleting the neural transmitter).

B. At the same time, the largest weight leading from the last neuron to fire (except the weight leading to neuron N) is set to $\bar{W}$ due to the pulse of neural transmitter, and a pulse of magnitude $\bar{W}$ is delivered to this postsynaptic neuron—so that it will fire after interval $\bar{C}$, thus to become the last-neuron-to-fire in the next cycle.

C. Finally, at the same instant as B: neural transmitter is depleted to all connections *to* and *from* the last-neuron-to-fire except the connections adjusted in steps A and B, connections already at maximum strength, and the connection to the *next-to-last-neuron* to fire. In essence, the strengthening of the weight in B depletes the competing weights. Further, propinquity in time sustains the strength of the weight leading from the *next-to-last-neuron* to fire.

D. Go back to A and repeat during the next cycle.

When the binary sequence ends, it is padded with zeros, and the above process continues until all firing ceases.

To fully explore this learning process, we use Figure 4.1 as an example. To simplify the explanation, neurons are numbered clockwise in such a way that the weights along the rim, starting at neuron 1 and moving in the clockwise direction, are initially the largest and are arranged in order of descending magnitude as one proceeds. Thus w_{12} is the largest, w_{23} the second largest, etc. The sequence to be learned is $I(n) = [1, 1, 0, 1]$.

It is padded with zeros so the training input is

$I(n) = [1,\ 1, -1,\ 1, -1, -1, -1, -1]$.

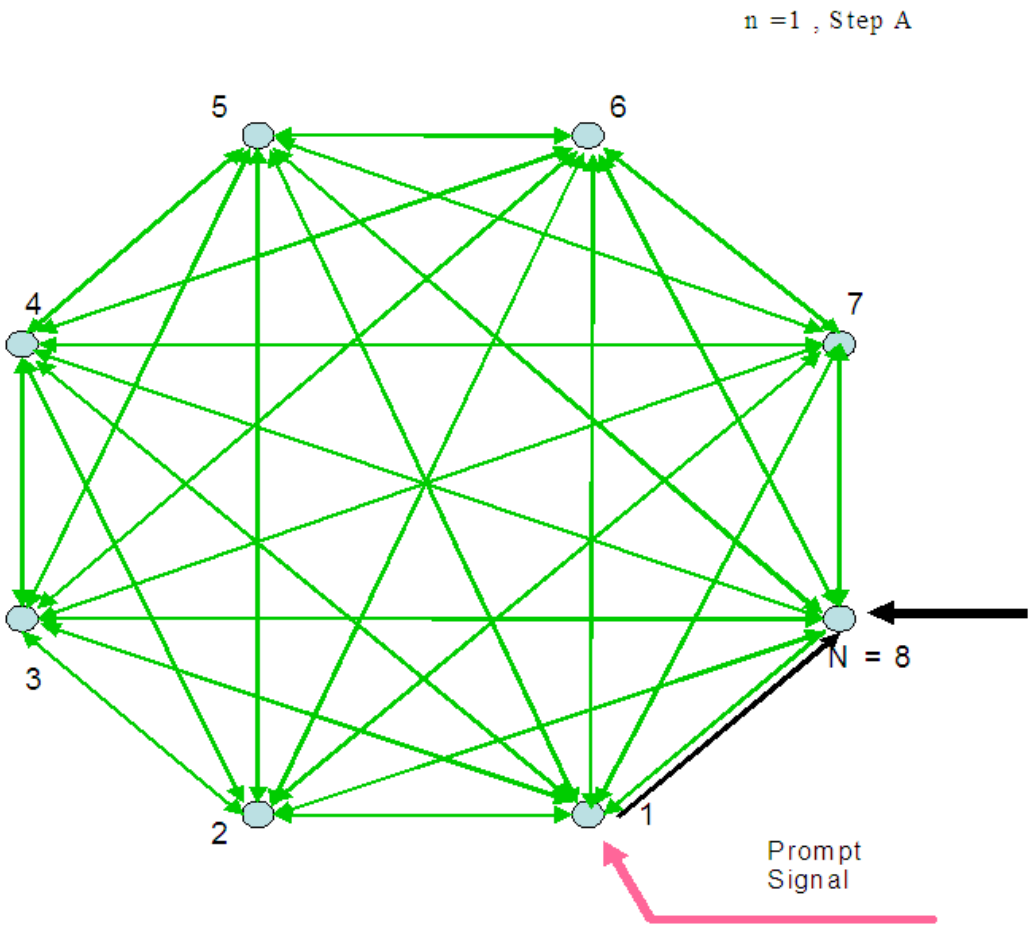

Figure 6.1. Depiction of cluster events during time step 1A: neuron 1 and neuron 8 fire.

Figure 6.1 shows the situation at $n = 1$, at completion of step A. Since both the prompt signal and the training signal are both reinforcing, the weight is increased to its maximum possible value. This is indicated by drawing the weight as a thick black arrow. At this stage, the last-neuron-to-fire is neuron 1. Figure 6.2 shows $n = 1$, at completion of step B.

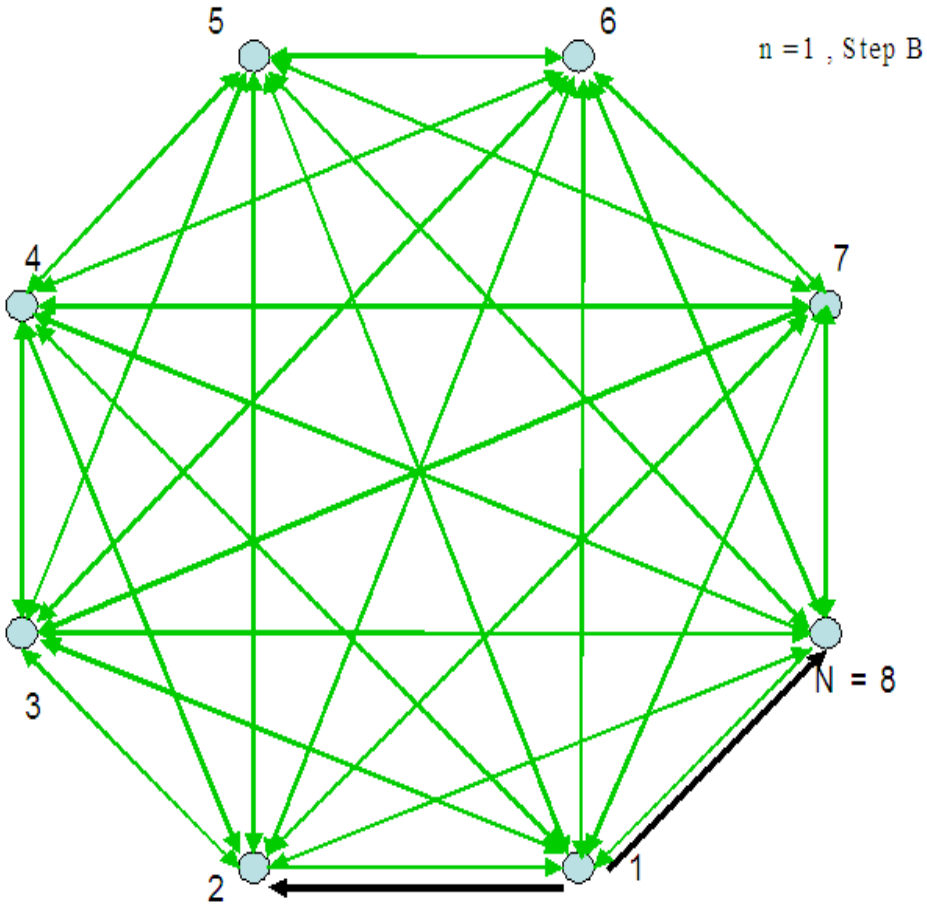

Figure 6.2. Depiction of cluster events during time step 1B: weight adjustment of the synapse between neuron 2 and 1.

In accordance with our numbering convention, w_{12} is the largest weight leading from the last-neuron-to-fire so it is reinforced to maximum value. Then a pulse of magnitude $\bar{W}$ is delivered to neuron 2, so that at time $n = 2$ it will become the last-neuron-to-fire.

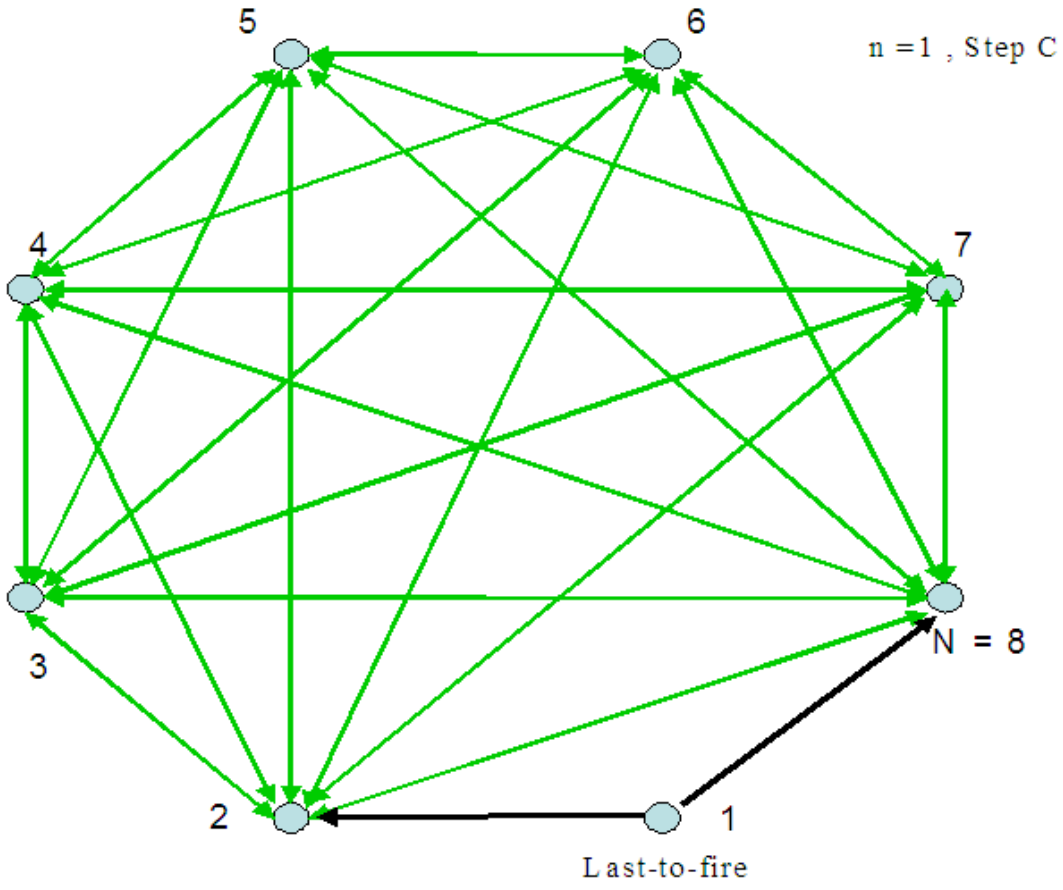

Figure 6.3. Depiction of cluster events during time step 1C: connections to neuron 1 are trimmed except for those along the perimeter of the cluster.

Finally, Figure 6.3 shows that all connections to and from neuron 1, except for those already reinforced are severed. In essence, there is a competition among all the weights linked to neuron one, and only the weights w_{18} and w_{12} survive. Note that because all the other links to neuron 1 are cut, it will not be possible to fire neuron 1 again during the training process. Neuron 1 cannot become a member of a closed loop.

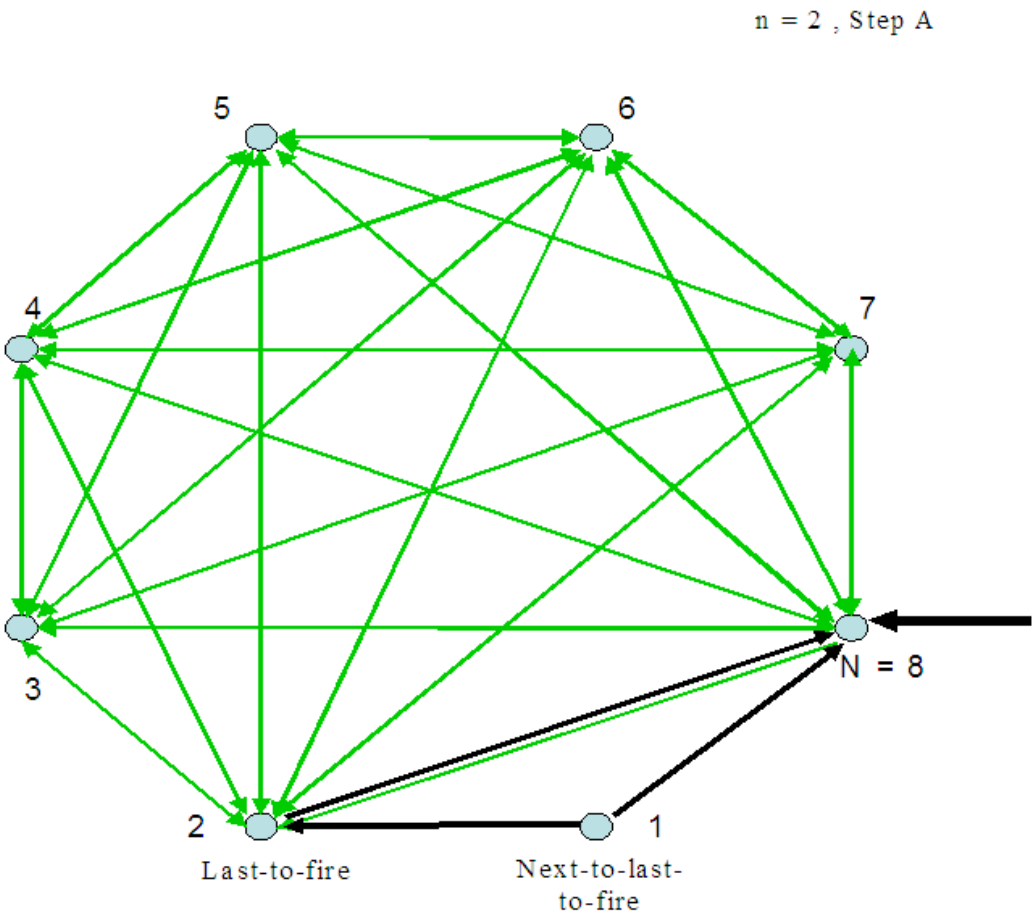

Figure 6.4. Depiction of cluster events during time step 2A: neuron 2 fires and the connection to neuron 8 is reinforced.

Next, at time $n = 2$, neuron 2 fires—thus becoming the last-neuron-to-fire. Since $T(2)$ is again positive the weight w_{28} is set to the maximum (See Figure 6.4).

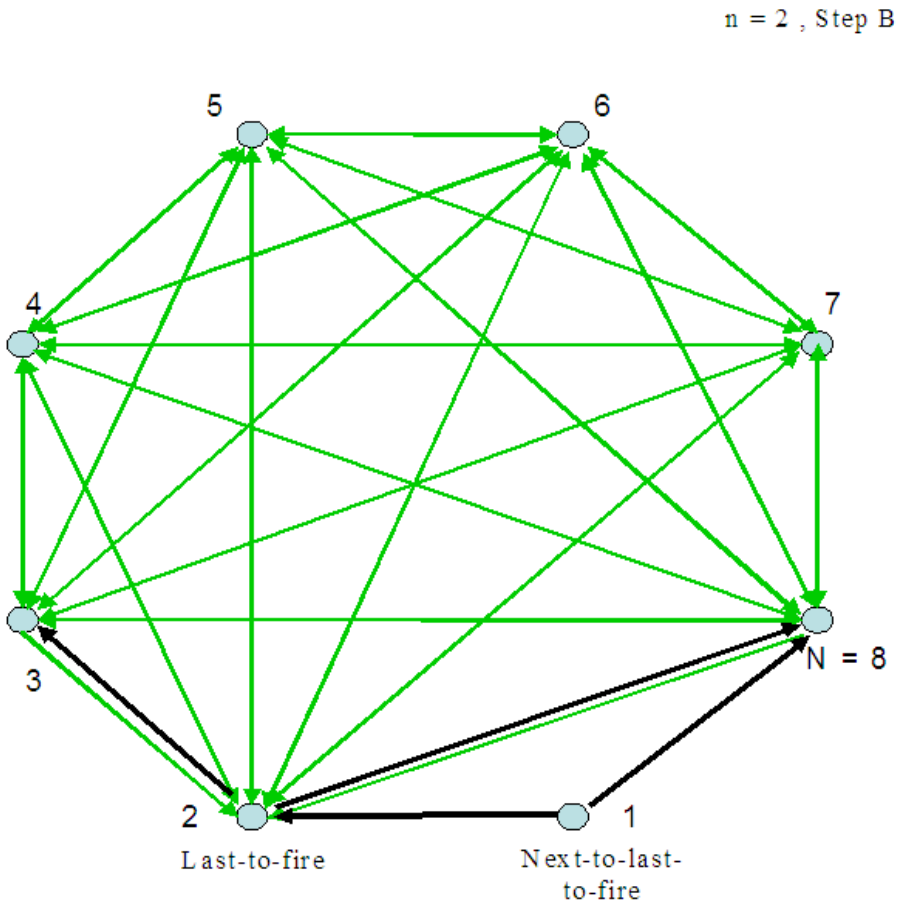

Figure 6.5. Depiction of cluster events during time step 2B: the connection between neuron 2 and 3 is reinforced.

On step B, as Figure 6.5 shows, aside from w_{28} , weight w_{23} is the largest weight issuing from neuron 2. This is reinforced, a firing pulse is delivered to neuron 3, and at time $n = 3$, it will become the last-neuron-to-fire.

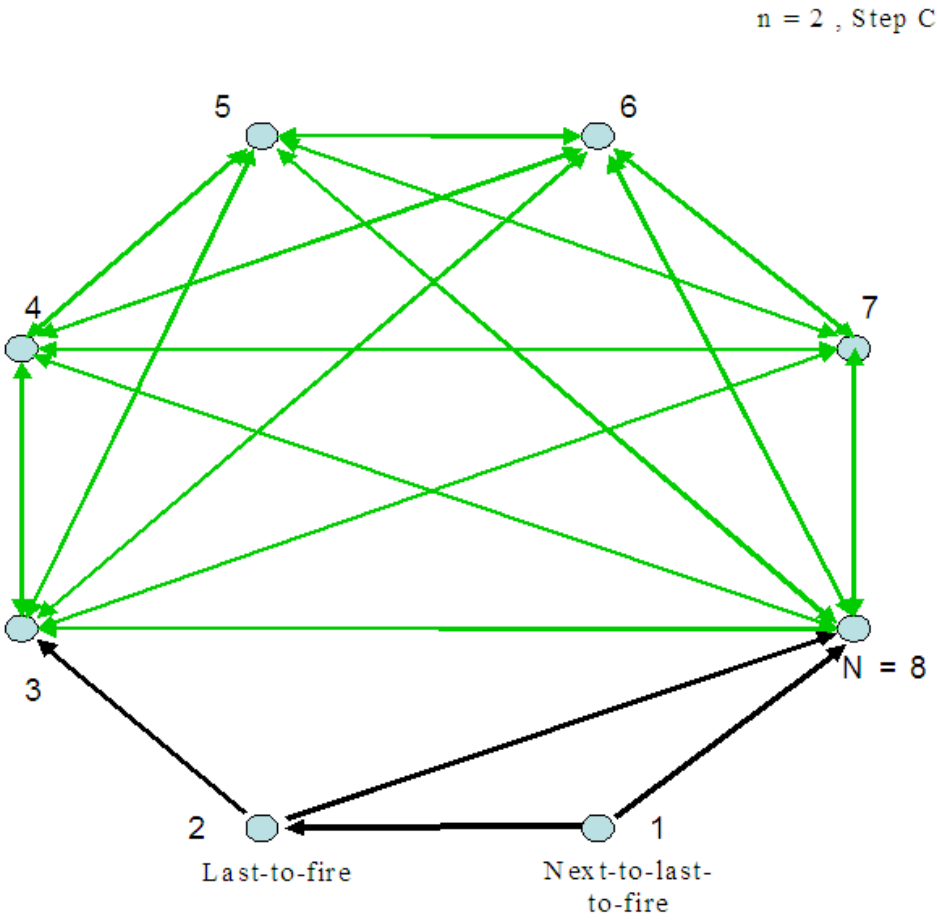

Figure 6.6. Depiction of cluster events during time step 2C: connections leading to neuron 2 are trimmed except for those between it and the last-to-fire and next-to-last-to-fire neurons.

Most importantly, as shown in Figure 6.6, all connections to and from neuron 2, except those already reinforced and *except the connection from the next-to-last-neuron-to-fire* are zeroed out. Now we can see that the system is building a chain of neurons that fire successively, and *without firing any one neuron more than once.*

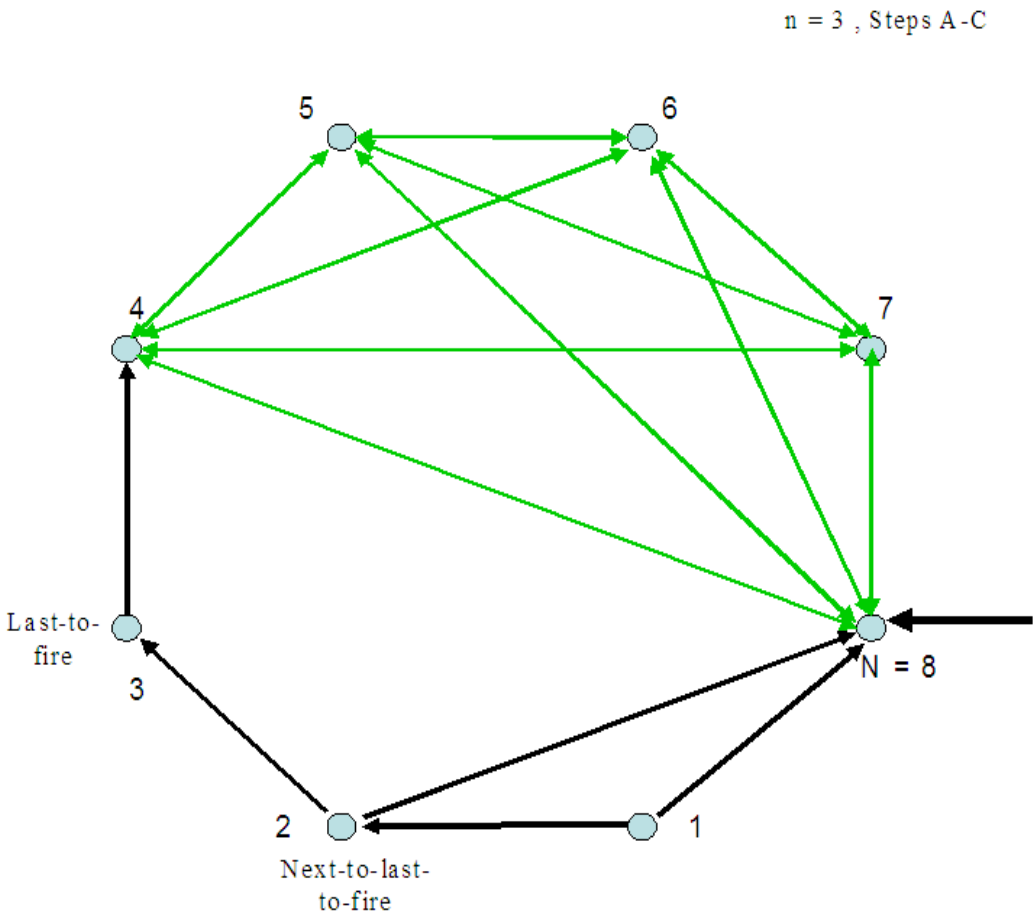

Figure 6.7. **Depiction of cluster events during time step 3A-C: the connection to neuron 8 is made ineffective according to the training pattern, the connection to neuron 4 is strengthened, and all other connections to neuron 3 are made ineffective.**

These points become even clearer in Figure 6.7 which shows the situation at time $n = 3$ when steps A-C are carried out. Because $T(3) = -1$, in this case the weight from neuron 3 to neuron N is zeroed out. Likewise, Figure 6.8 shows the configuration at $n = 4$.

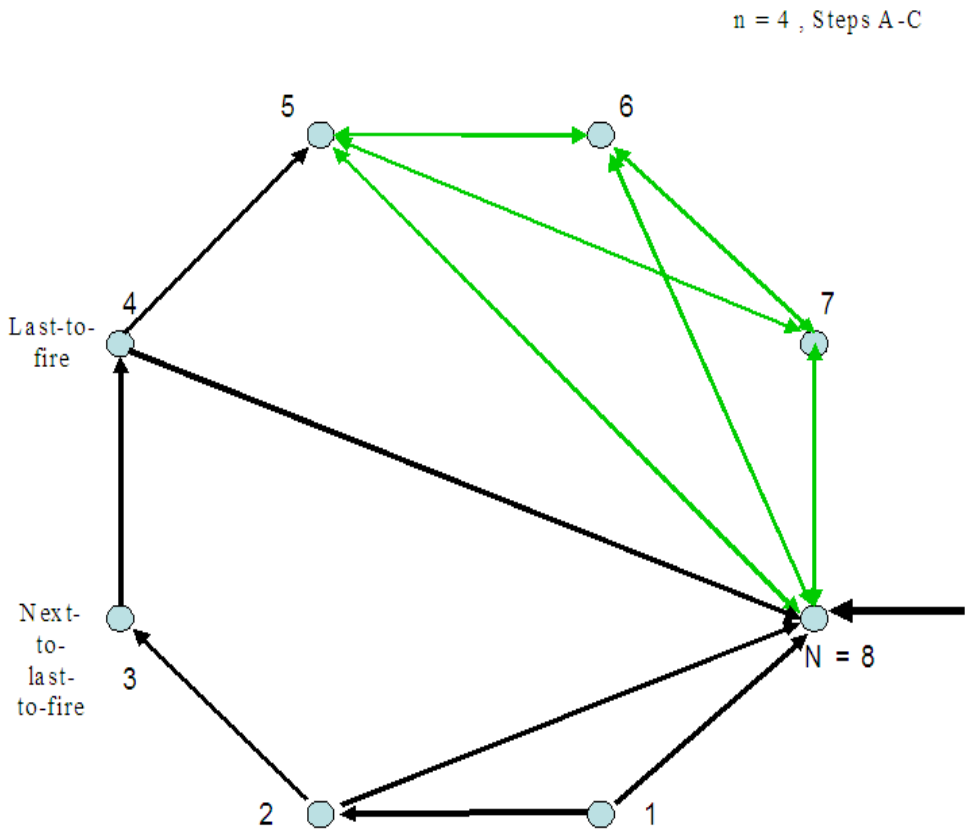

Figure 6.8. **Depiction of cluster events during time step 4AC: neuron 4's connection to neuron 8 is strengthened according to the training pattern, the connection to neuron 5 is strengthened, and all other connections to neuron 4 are made ineffective.**

Similar to the steps during neuron 2's firing, A-C entail the strengthening of the connection between neuron 4 and neuron 8, between neuron 4 and neuron 5, and lastly, the depletion of neurotransmitter to all other connections to or from neuron 4.

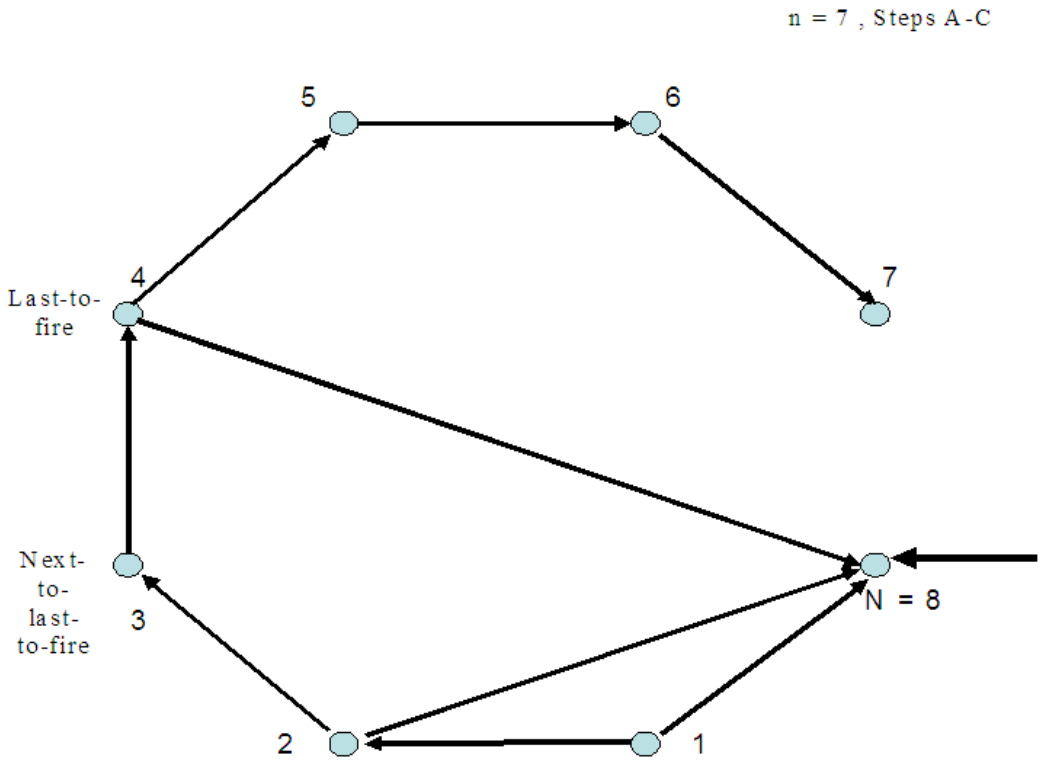

Figure 6.9. **Depiction of cluster events during time step 7A-C: neuron 7's connection to neuron 8 is made ineffective according to the training pattern, the respective connections to neuron 6 and 7 are strengthened, and all other connections to neuron 5-7 are made ineffective.**

Finally, Fig. 6.9 shows the result once all firing has ceased. Since the training signal is inhibitory after $n = 4$, there are no connections from the remainder of the rim to neuron N. However the chain of neurons is still completed as a result of step B.

In summary, after one training session, the system has organized itself into a delay line: a chain of neurons that fire one after the other, without repetition. The fan of weights leading from this chain to neuron N contains the information of the learned binary sequence. It is obvious that after a single training session, if the prompt signal is applied without training input, neuron N will produce the output:

$\hat{I}(t) = \sum_{m=1}^{M} I_m \delta(t - m\bar{C}),\ I_k = 0 \text{ or } 1 \text{ for } k = 1, \ldots, M$, as desired.

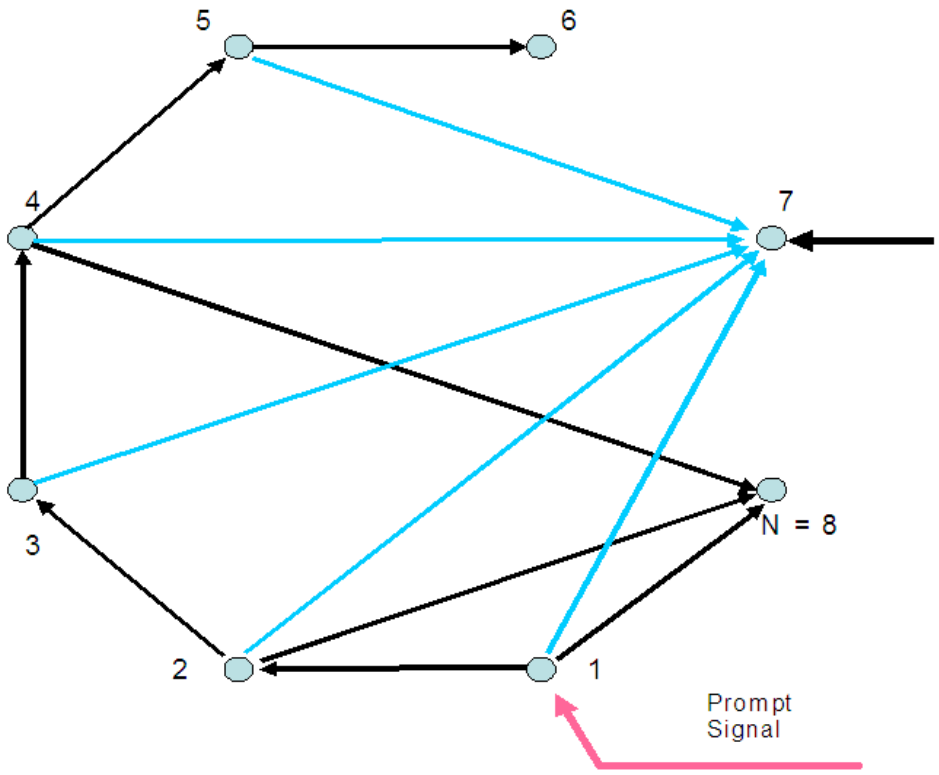

Figure 6.10. **Depiction of the cluster's capability to learn an entirely new sequence: the blue lines indicate the connections formed by the additional sequence learned using a different neuron as the training neuron.**

A further point is that additional sequences can be learned, by designating a "Tail end" neuron as the receptor of the training sequence. Figure 6.10 shows the final state when neuron 7 is the designated output node and the sequence to be learned is $I(n) = [1,1,1,1]$. For this second training session, the delay line is already built and steps B and C are already carried out. Only step A remains, resulting in the blue colored weights in the leading into neuron 7. Although neuron 8 will still fire in the sequence it has been

trained to repeat, it will have no effect on the weights going into neuron 7. Thus several sequences can be learned without over-writing.

To complete our description, we set down the particularly simple mathematical algorithm. Let $\kappa(n)$ denote the index of the last-neuron-to-fire at time n, with $\kappa(n) = 1$. Define the following limiter function:

$$\mathrm{L}_{\bar{W}}(x) = \begin{cases} 0, x < 0 \\ x, 0 \le x < \bar{W} \\ \bar{W}, x \ge \bar{W} \end{cases} \tag{6.1}$$

Then the learning algorithm is:

$$\begin{gathered} \mathrm{w}_{\kappa(n),N}(n+1) = \mathrm{L}_{\bar{W}}\left[\mathrm{w}_{\kappa(n),N}(n) + \bar{W}T(n)\right] \\ m^* = \arg\max_{m \neq N}\left\{\mathrm{w}_{\kappa(n),m}(n)\right\} \\ \mathrm{w}_{\kappa(n),m}(n+1) = \mathrm{L}_{\bar{W}}\left[\mathrm{w}_{\kappa(n),m}(n) + 2\bar{W}\left(\delta_{m,m^*} - \frac{1}{2}\right)\right], \forall m \neq N \\ \mathrm{w}_{\kappa(n),m}(n+1) = \mathrm{L}_{\bar{W}}\left[\mathrm{w}_{m,\kappa(n)}(n) - \bar{W}\right], \forall n \neq \kappa(n-1) \\ \kappa(n+1) = m^* \end{gathered} \tag{6.2}$$

6.2 Network Demonstration

This portion demonstrates the Fast Learning Algorithm (FLA), Eq. (6.2), presented in the previous section, using a simple MatLab simulation. The following figures show the structuring of a 10 neuron, initially all-to-all, connected network with randomly assigned connection weights. It is of note that while in the previous section's example, the weight values around the perimeter were assumed to be decreasing in the clockwise direction, this simulation does not utilize that assumption. Thus, as the

network restructures itself to learn the training pattern, the path from one firing neuron to the next will not be orderly as it was previously. The program will maintain neuron 1 as the prompt neuron and the training pattern, $T = [\,1\;1\;1\;1\;0\;0\;1\;1]$, will be applied to neuron N, in this case $N = 10$.

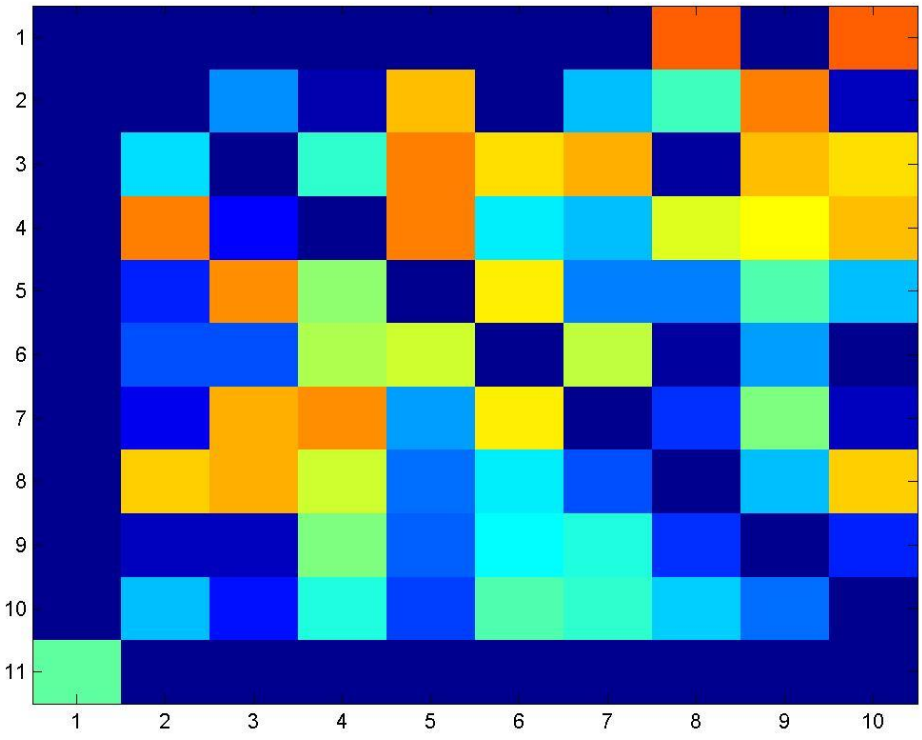

Figure 6.11. Representation of the FLA connections at $t = 1$: neuron 1 has fired, the connections to all but neuron 6 and neuron 10 have been made ineffective.

In coordinate (1,8) the firing of neuron 1 can be seen to have strengthened the connection with neuron 8. Because the training element for this time step is 1, the connection to the output neuron, 10, has been strengthened as well. All other connections to neuron 1 have been made ineffective. Note that row 11 of the figure indicates the chronological firing pattern of the output neuron and column 10 presents the strengthening of the weights spatially.

The figure below presents the state of the network at a later point during training. The row index indicates the neuron that the pulse is coming from and the column index is the neuron receiving the pulse.

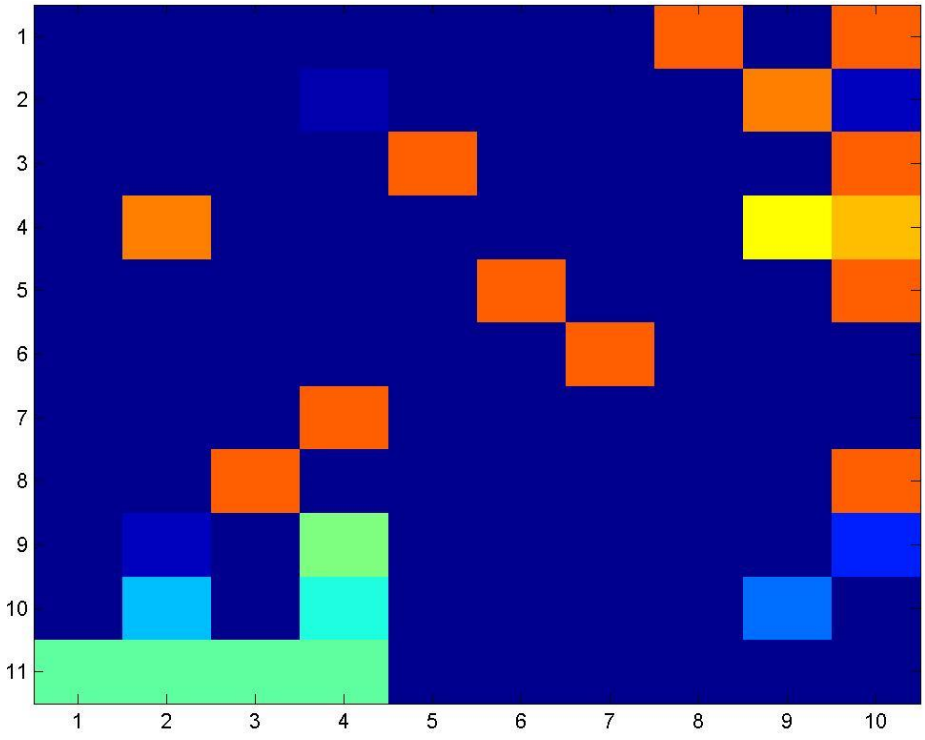

Figure 6.12. **Representation of the FLA connections at $t = 2$: neuron 7 has fired, the connections to all but neurons 5 and 8 and coming from 1, are ineffective.**

By tracing the indices we can see that the following path has been formed:

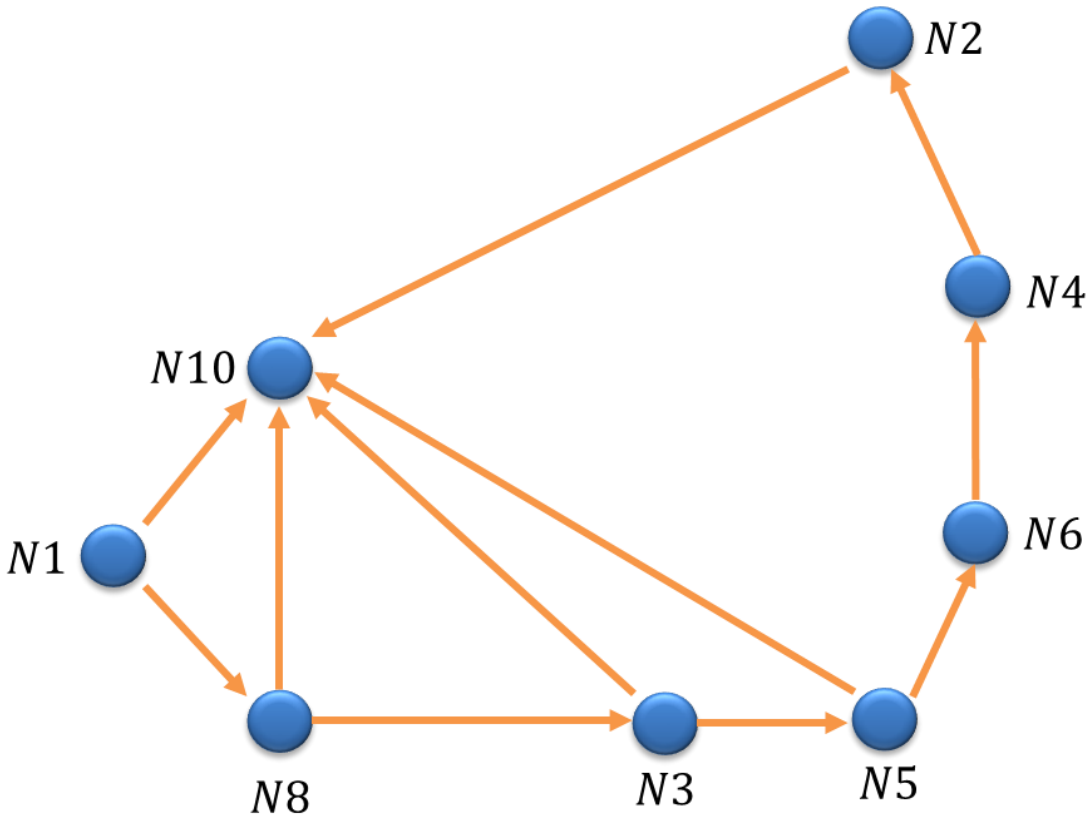

Figure 6.13. Depiction of the shape of the self-structuring network at the point represented previously

Now moving forward to when training has been completed, we have:

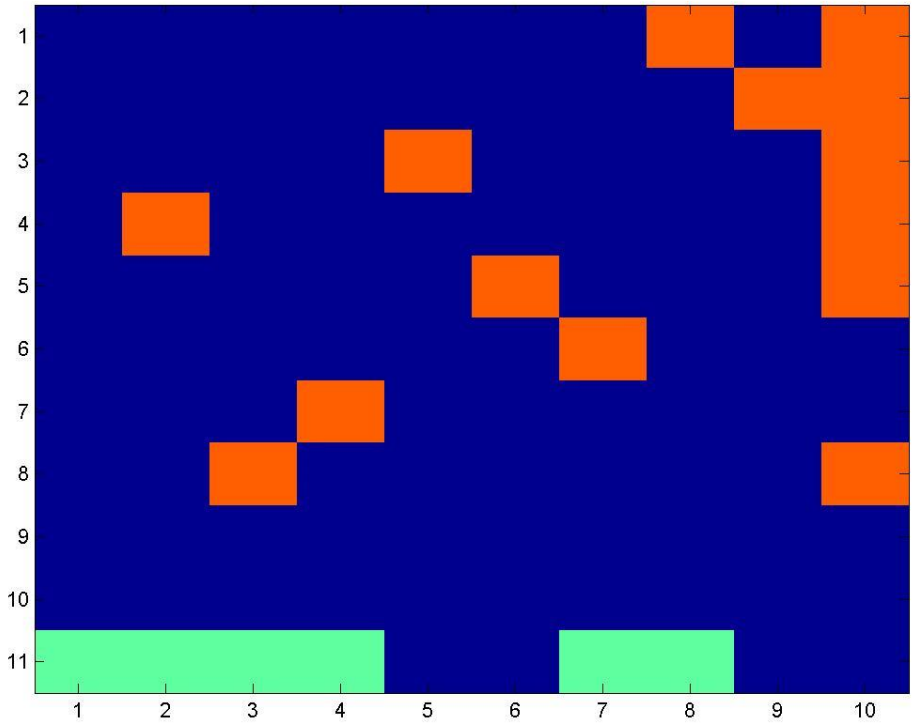

Figure 6.14. **Representation of the FLA when training is completed: the spatial connections between the final neuron and other neurons is shown in column 10, and the temporal sequence generated by the network is shown in the last row**

Note that the temporal sequence in row 11 matches the training sequence. The figure below shows the final network shape.

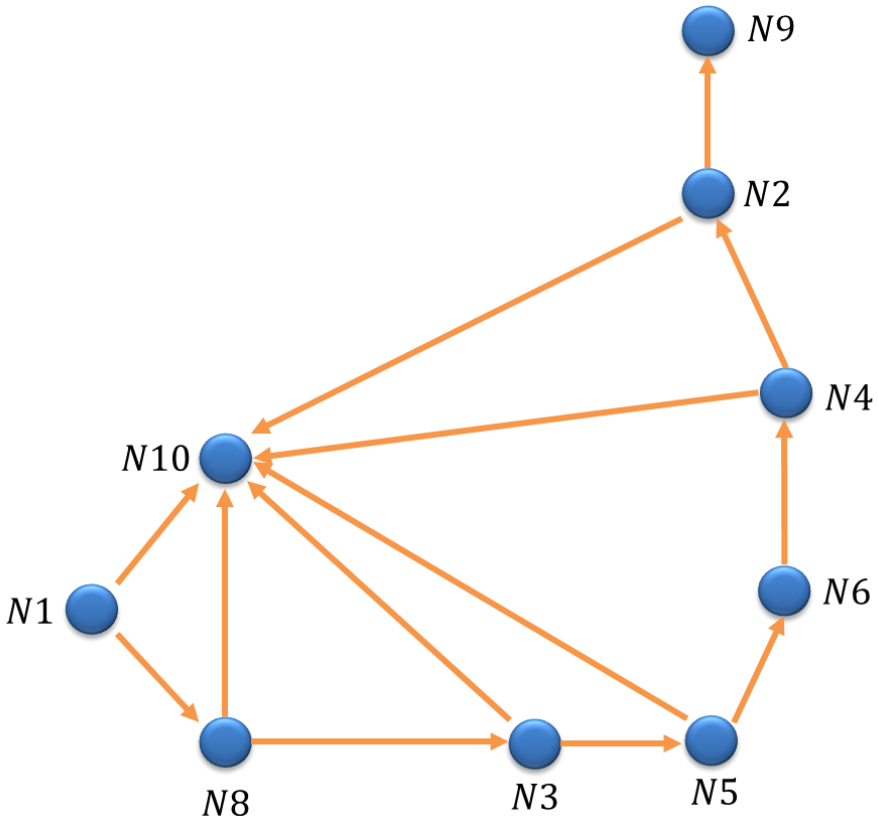

Figure 6.15. **Depiction of the final shape of the self-structuring network after training.**

7. CONCLUSION

In summary, this thesis presents a neural network model and framework describing the architecture required to accomplish two tasks, (1) recognition of image data and matching with stored patterns for image noise reduction, and (2) rapid learning of temporal sequences for dynamic system characterization.

Concerning the recognition of image data and matching with stored patterns, we saw in section one that Stage II neural models based on the average firing rate by nature cannot encode data as well as spike-time based neurons. In addition to the Stage II neurons, backpropagation algorithm is found to be slow and unreliable. Experimental evidence based on human reaction times lends support to spike-time based neurons being a more accurate description of biological processes due to their faster processing time. Therefore, of the many models developed to accurately portray the mechanics of neurons, spiking neural net models are adopted to accomplish the tasks.

There are many different SNN models that mathematically model the action of interconnected neurons, most of which focus on the evolution of neuron potential. However, while these models simulate the mechanical operations of neurons, sometimes in extreme detail, they do not attempt to find and isolate the necessary feature(s) responsible for the desired functionality. This thesis shows that the finite, initial slope of the postsynaptic response is a necessary feature that enables recognition of data and matching data to stored patterns in a competitive structure. That is demonstrated in the Competitive Classifying Unit, which is hierarchically composed of groups of

Comparator neurons that make up Shift Blocks which are the basic units of Associators. The Competitive Classifying Unit architecture exploits a competitive process to achieve very rapid identification and matching with low sensitivity to non-concurrent arrival times of inputs entering comparator neurons. Thus, the system will function correctly as long as the incoming pulses are just short of a pulse width apart, or as long as there is the slightest overlap between responses. Furthermore, the system has low sensitivity to variations in the firing threshold, showing that when the threshold is allowed to vary within several different intervals, the system will select the correct match up to 90% of the time with a standard deviation of 0.8 and 100% with smaller standard deviations.

It is shown that certain homeostasis conditions are necessary and sufficient for synchronous and periodic firing. Stability conditions for periodic firing are determined. This provides the basis for an all-to-all connected, spiking net that maintains an internal "clock." The training process for learning a periodic sequence of binary numbers is constructed. A simple synaptic weight update within a sparsely connected system of N neurons is simulated and is capable of quickly learning binary sequences.

Subsequently, we developed a more generalized version of this structure, still based on the network theory from section four. In the new version, the "delay" lines are preset, or in other words, the basic structure of the all-to-all connected network is not predetermined: it starts with purely randomly weighted connections and constructs the interconnection pattern in real time. As the network is trained with a sequence, it takes on whatever structure that will accomplish the task. This more general algorithm is capable of self-organizing in the manner that had been assumed from the start in the

previous network. Finally, a MatLab simulation demonstrates the fast learning capability of this algorithm.

Future work would be to address the assumption that the training sequence and the network are firing with the same period. The question would be: Can a network be trained to learn a new firing period? Similarly, different groups of neurons have been found to have different firing periods. Can information be encoded in the firing period itself?

The human brain is still a black box. While we are gaining more and more knowledge of the minutiae of brain mechanics, we are still searching to find out which features will allow us to harness specific capabilities. While the search goes on, this thesis has answered a few of the plethora of questions.

REFERENCES

[1] K. Mehrotra, C. K. Mohan and S. Ranka. Elements of Artificial Neural Networks MIT Press, 1997.

[2] D. Marr. Vision : A Computational Investigation into the Human Representation and Processing of Visual Information, 1982.

[3] S. Cavallari, S. Panzeri and A. Mazzoni. “Comparison of the dynamics of neural interactions between current-based and conductance-based integrate-and-fire recurrent networks.” Frontiers in Neural Circuits, vol. 8, pp. 150-157, 2014. <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3943173/>.

[4] J. R. Rabunal and J. Dorrado. Artificial Neural Networks in Real-Life Applications 2006. <http://lib-ezproxy.tamu.edu:2048/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=140158&site=ehost-live>.

[5] X. Gu. "Spatial-temporal-coding pulse coupled neural network and its applications," in Neuronal Network Research Horizons, M. L. Weiss, Ed., 2007.

[6] Neural Networks for Vision, Speech, and Natural Language Ed. Myers and Nightingale, 1992.

[7] H. Lin. “Identification of spinal deformity classification with total curvature analysis and artificial neural network.” Presented at the 27th Annual International Conference of Engineering in Medicine and Biology, 2005.

[8] A. Meyer-Bäse. “Neural net computing for image processing,” in Computer Vision and Applications a Guide for Students and Practitioners, H. Haussecker, Ed., 2000.

[9] M. Ghasemi Varnamkhasti. “Aging fingerprint characterization of beer using electronic nose.” Sensors and Actuators, vol. 159, pp. 51-59, 2011.

[10] R. Schuster, S. Schulter, G. Poier, M. Hirzer, J. Birchbauer, P. M. Roth, H. Bischof, M. Winter and P. Schallauer. “Multi-cue learning and visualization of unusual events.” Presented at Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference, 2011.

[11] L. Wang, S. Z. Der and N. M. Nasrabadi. “Automatic target recognition using a feature-decomposition and data-decomposition modular neural network.” Image Processing, IEEE Transactions, vol. 7, pp. 1113-1121, 1998.

[12] A. P. Engelbrecht. Computational Intelligence an Introduction, 2007.

[13] X. Pan, B. Lee and C. Zhang. “A comparison of neural network backpropagation algorithms for electricity load forecasting.” Presented at Intelligent Energy Systems (IWIES), IEEE International Workshop, 2013.

[14] N. Belghini, A. Zarghili, J. Kharroubi and A. Majda. “Color facial authentication system based on neural network.” Presented at Information Science and Technology (CIST), 2011.

[15] L. V. Fausett. Fundamentals of Neural Networks : Architectures, Algorithms, and Applications, 1994.

[16] E. Soria, J. D. Martín and P. J. G. Lisboa. "Metaheuristic procedures for training neural networks," SpringerLink (Online service), Eds., 2006.

[17] S. Thorpe and D. Fize. “Speed of processing in the human visual system.” Nature 381(6582), pp. 520, 1996. <http://lib-ezproxy.tamu.edu:2048/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=9606165495&site=ehost-live>.

[18] F. Y. Ahmed, B. Yusob and H. N. A. Hamed. “Computing with spiking neuron networks, A review.” International Journal of Advances in Soft Computing & its Applications, vol. 6, 2014.

[19] W. Gerstner and W. Kistler. Spiking Neuron Models: Single Neurons, Populations, Plasticity, 2002.

[20] W. Mass and C. M. Bishop. Pulsed Neural Networks 1999.

[21] W. Gerstner. Plausible neural networks for biological modelling, Kluwer Academic Publishers, Dordrecht, Boston, 2001.

[22] A. N. Burkitt and G. M. Clark. “Analysis of integrate-and-fire neurons: Synchronization of synaptic input and spike output.” Neural Computing, vol. 11, pp. 871-901., 1999. <http://dx.doi.org/10.1162/089976699300016485>.

[23] W. Gerstner. Time structure of the activity in neural network models. American Physical Society. vol. 51, pp. 738-758., 1995. <http://link.aps.org/doi/10.1103/PhysRevE.51.738>.

[24] J. J. B. Jack. Electric Current Flow in Excitable Cells, 1975.

[25] I.A. Kuznetsov & A.V. Kuznetsov, “Can numerical modeling help understand the fate of tau protein in the axon terminal?” Computer Methods in Biomechanics and Biomedical Engineering, 2015.

[26] L. Abbott and T. B. Kepler. “Model neurons: From hodgkin-huxley to hopfield,” Statistical Mechanics of Neural, 1990.

[27] R. FitzHugh. “Impulses and physiological states in theoretical models of nerve membrane.” Biophys. J. vol. 1, pp. 445-466., 1961. <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1366333/>.

[28] E. M. Izhikevich. “Simple model of spiking neurons.” IEEE Trans. Neural Networks, vol. 14, pp. 1569-1572., 2003.

[29] R. B. Stein. “Some models of neuronal variability.” Biophys. J. vol. 7, pp. 37-68. 1967.

[30] W. Gerstner, and W.M. Kistler. Spiking Neuron Models – Single Neurons, Populations, Plasticity. Cambridge University Press, Cambridge, U. K, 2002.

[31] R. Eckhorn, H. J. Reitboeck, M. Arndt and P. Dicke. “Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex.” Neural Computing. vol. 2, pp. 293-307. 1990. <http://dx.doi.org/10.1162/neco.1990.2.3.293>.

[32] J. Bibby, "Axiomatisations of the average and a further generalization of monotonic sequences," Glasgow Mathematical Journal, vol. 15, pp. 63-65, 1974.

[33] G. Bi and M. Poo, “Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type,” J. Neuroscience.,vol. 15, pp. 10464-10472, 1998.

[34] L. I. Zhang, H. W. Tao, C. E. Holt, W. A. Harris, and M. Poo, "A critical window for cooperation and competition among developing retinotectal synapses," Nature, vol. 395, pp. 37-44, 1998.

[35] V. Egger, D. Feldmeyer, and B. Sakmann, "Coincidence detection and changes of synaptic efficacy in spiny stellate neurons in barrel cortex," Nature Neurosci., vol. 2, pp. 1098-1105, 1999.

[36] G. Q. Bi and M. M. Poo, "Distributed synaptic modification in neural networks induced by patterned stimulation," Nature, vol. 401, pp. 792-796, 1999.

APPENDIX

Although the input and output images appear 3D, all computations were performed on 2D shapes. MatLab's 3D plot function generated these 3D images from 2D input to enable detection of miniscule differences not apparent using the 2D plot function.

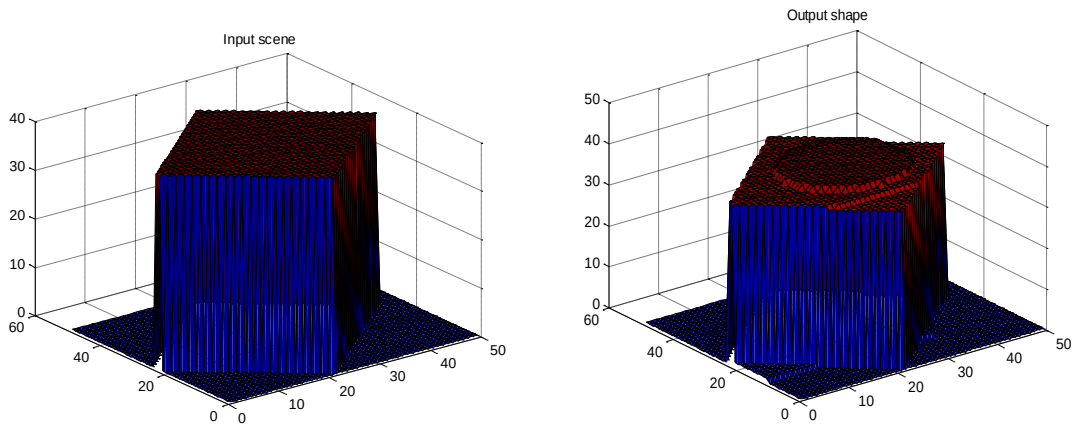

Figure A-1 Comparison of input, a square, (Left) and output (Right) of the gradient descent program.

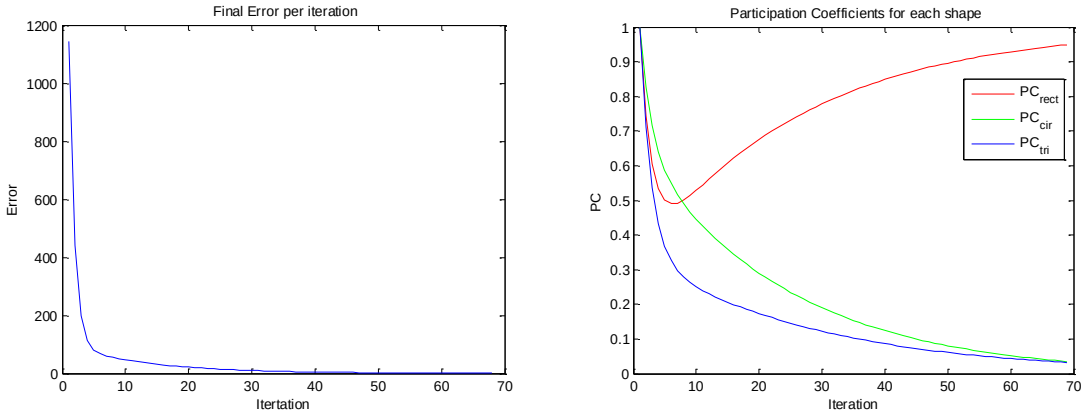

Figure A-2 Graphical representation of the error (Left) and the participation coefficient values for each test shape (Right).

The above is the final error between the input scene and the output image.

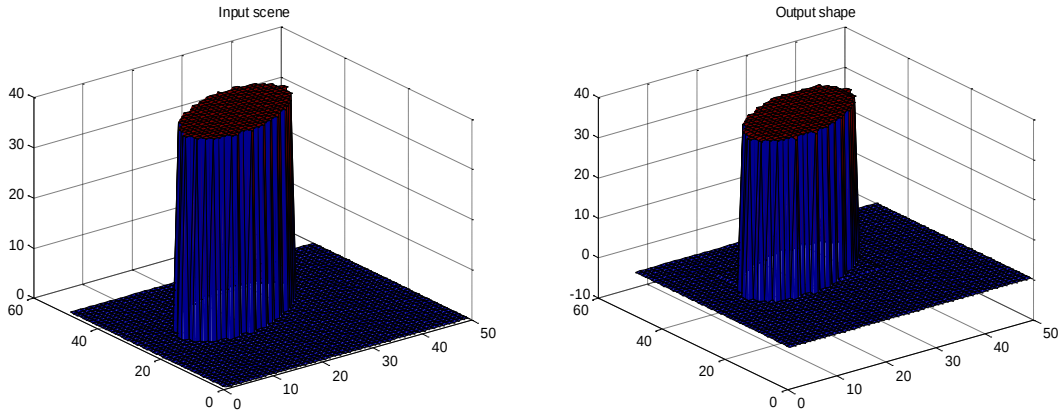

Figure A-3 Comparison of input, an oval, (Left) and output (Right) of the gradient descent program.

Due to the constraint that all participation coefficient parameters should sum to one, competition results in the rectangle "winning" around the seventieth iteration.

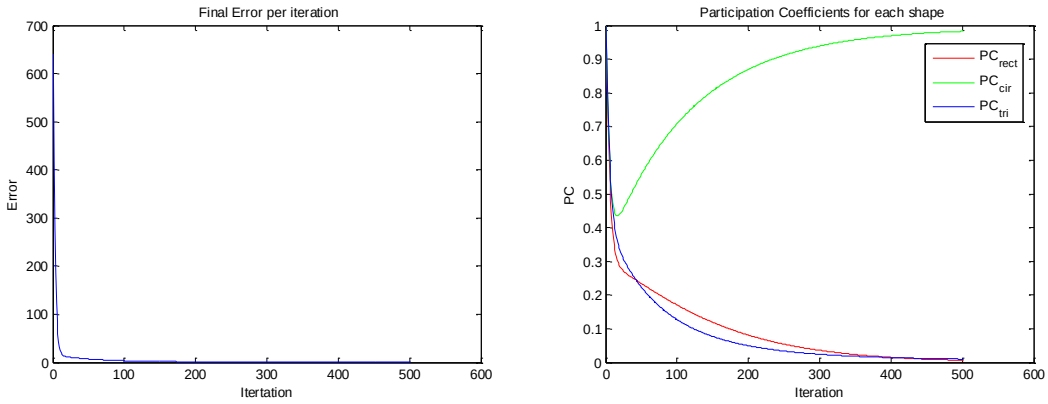

Figure A-4 Graphical representation of the error (Left) and the participation coefficient values for each test shape (Right)

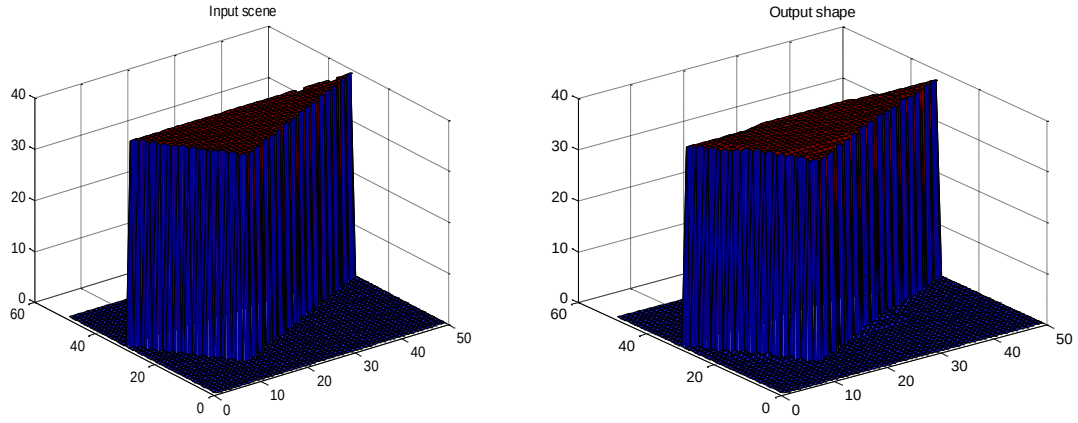

Figure A-5 Comparison of input, a triangle, (Left) and output (Right) of the gradient descent program

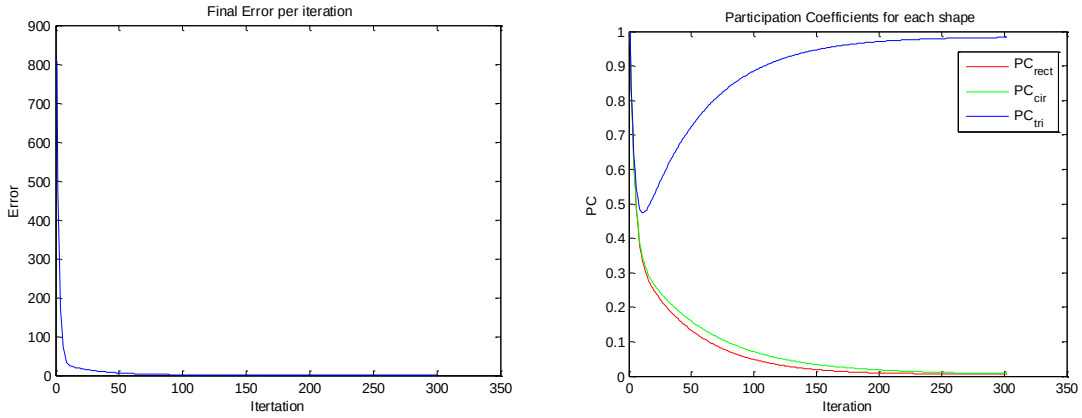

Figure A-6 Graphical representation of the error (Left) and the participation coefficient values for each shape (Right)

It is important to remember that while the results of this matching experiment appear to show an effective method for matching, the process to acquire these results demanded adjustment user adjustment to match each new shape.