SSNNS - A Suite of Tools to Explore Spiking Neural Networks

Heike Sichtig
Bioengineering Department
Binghamton University
Binghamton, NY 13902
hsichtig@binghamton.edu

J. David Schaffer
Philips Research North Am.
345 Scarborough Road
Briarcliff Manor, NY 10510
dave.schaffer@philips.com

Craig B. Laramee
Bioengineering Department
Binghamton University
Binghamton, NY 13902
claramee@binghamton.edu

ABSTRACT

We are interested in engineering smart machines that enable backtracking of emergent behaviors. Our SSNNS simulator consists of hand-picked tools to explore spiking neural networks in more depth with flexibility. SSNNS is based on the Spike Response Model (SRM) with capabilities for short and long term memory. A genetic algorithm, namely CHC, is used independently to generate such example systems that produce patterns of interest. Foundational work in the growing field of spiking neural networks has shown that precise spike timing may be biologically more plausible and computationally powerful than traditional rate-based models[4][7]. We have been using evolution to discover neural configurations that produce patterns of interest.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Experimental Design, Time Series Analysis*
; I.6.4 [**Computing Methodologies**]: Simulation and Modeling—*Model Validation and Analysis*

General Terms

Algorithms, Design, Experimentation, Theory, Verification

Keywords

Spiking Neural Networks, Genetic Algorithms, Complexity, Learning, Temporal Pattern Recognition, Machine Learning

1. INTRODUCTION

The McCulloch and Pitts publication in 1943 can be seen as the beginning of artificial neural networks. In their paper, a highly simplified model of basic brain cells, called MCP neurons, is presented. Since then we have been attempting to build machines using principles that are abstracted from how the brain could produce highly complex patterns. Those MCP neurons had limitations. You could implement any boolean function, but had to design each one. The original Perceptrons, introduced by Rosenblatt, went a step further by including a learning capability, but were limited to linearly separable patterns. In fact, Perceptrons were intended to be pattern recognition devices where their association units correspond to a feature or pattern detector. That was extended to non-linear patterns by Rumelhart et al.[10] with the multi-layer perceptrons and the backprop algorithm. Still, there were problems that seemed to defy these networks, including time series patterns.

Complex systems, such as ourselves, financial markets, weather, politics, dating and other areas present challenging issues connected to perception, analysis and prediction of their spatial-temporal data. We hope to use our smart machines for more precise time series prediction.

One of the most promising technologies to represent spatial-temporal data is spiking neural networks (SNNs). The spike response model (SRM) is a generalized leaky integrate and fire model and well suited to simulate neuronal communication at the abstract level. The SRM model expresses the membrane potential at time t as an integral part over the past[4]. Synapses connect neurons via axons and dendrites. The neuron connecting to the synapse is called the presynaptic neuron, whereas the neuron connected from the synapse is the postsynaptic neuron. In theory, the synapse can behave in 3 ways: static, depressing or facilitating towards the postsynaptic neuron. The dynamic synapse as proposed by Markram allows for facilitating and depressing behavior for transient dynamics[8][9]. In facilitating synapses the response grows with successive presynaptic spikes, whereas depressing synapses lower the response in a frequency-dependent way. The reinforcement learning as proposed by Hebb in 1949 is the main hypothesis for a neural mechanism to explain long term learning[5]. Hebbian learning has been successfully applied to spiking neural networks by Song et al.[12].

Mass et al.[7], have shown how the spiking model in general can do everything that older style models could, and often with many fewer neurons. Markram et al.[9], have shown that the dynamic synapse enables at least the possibility for networks that extract many different features from the same spike train. This suggests network capabilities, but doesn't show either how to achieve them or what they might be used for. Their work is all about a single synapse. Hebb proposed a simple model for long term potentiation at the synapse[5], and Kandel later showed that it was physiologically realistic[1]. But there still seems to be no real insight into how this low level function can translate into network behavior, with the exception of the simple idea that if initial wiring provides an over abundance of synapses, that perhaps Hebbian learning can prune away the excess and tune what's left to perform very accurately.

Our SNN simulator, called SSNNS, is coupled with a genetic algorithm that independently evolves parameters to predict spike patterns of interest, our smart machines. This

GECCO'08, July 12–16, 2008, Atlanta, Georgia,USA.
ACM 978-1-60558-131-6/08/07.

hybrid SNN-GA architecture can be used to investigate processes that lead to learning on an abstract level and can help researchers to develop a more comprehensive understanding of the intricate workings that govern pattern recognition and learning. Our produced smart machines might reveal novel insights that are usually black-boxed because we know what is on the inside. The genetic algorithm, CHC [3], is an excellent tool to investigate unknown fitness landscapes. Solutions may be very scarce, hence, CHC is a good choice for searching over complex landscapes. The restarts in CHC allow us to *jump* to different places for searching.

Our research is an extension of COGANN-92 workshop on Combinations of Genetic Algorithms and Neural Networks that surveyed various schemes for combining genetic algorithms with neural networks [11]. The presented smart machines are composed of spiking neural networks with specific building blocks and a genetic algorithm is used to evolve network parameters and learning rules. It is our hope to devise a general learning rule for such systems.

2. CURRENT RESEARCH

2.1 SSNNS Simulator

The underlying mathematics of the synchronous simulator is the general SRM model (see Equation 1), consisting of a function for refractoriness η and its own spike response ϵ. Information is kept at every time step according to a post synaptic potential lookup table, using a spike response function that keeps track of the intensity of action potentials over time. The length and intensity of action potentials can be adjusted by tweaking either the membrane or synaptic time constant. The spike response function is useful in modeling a more biologically plausible process.

The SRM model in Equation 1 is a combination of η for refractoriness and ϵ for spike response.

$$u_j(t) = \sum_{t_j} \eta(t - t_j) + \sum_i \sum_{t_i} w_{ji} \times \epsilon(t - t_i - d_{ji}) \qquad (1)$$

The computation time is t and t_i are the spike times of neuron i (pre-synaptic neuron), and t_j are the spike times of neuron j (post-synaptic neuron). The spike response function is weighted by w_{ji}, the weight of the connection from pre-synaptic neuron i to post-synaptic neuron j. When computing the spike response, the previous spikes might be delayed; thus, d_{ij} is the delay of the connection from pre-synaptic neuron i to post-synaptic neuron j. The post-synaptic potential of neuron j is calculated at every time step t, starting from the first input until maximum simulation time.

2.1.1 Calculation of η in SSNNS

For every neuron the refractoriness is calculated by using an exponential decay function at every time step. Equation 2 shows the calculation of η, ETA_0 is the negative spike-after potential and U_R is the resting potential.

$$\eta = (ETA_0 - U_R) \times \exp^{\frac{-(\Delta t)}{T_r}} \qquad (2)$$

T_r is the refractory time constant. The refractoriness is computed by repeatedly taking the time difference t from last spike time of neuron j to current computation time. The spike response function for the computation of PSP using the SRM model is shown in Equation 3.

2.1.2 Calculation of ϵ in SSNNS

$$\epsilon = \frac{1}{1 - \frac{T_s}{T_m}} \times \exp^{\frac{-(\Delta t)}{T_m}} - \exp^{\frac{-(\Delta t)}{T_s}} \qquad (3)$$

It is an exponential growth function with 2 parameters for rise time and length of PSP adjustment. Rise time is adjusted with the synaptic time constant T_s and length with the membrane time constant T_m. The time difference Δt is from the spike times of the pre-synaptic neuron to current computation time. The spike response function(ϵ) allows for more flexibility in order to control time in history and intensity simultaneously. Alternatively, the spike response can be modeled using the synaptic time constant only (see Equation 4).

$$\epsilon_{alt} = \frac{t}{T_a^2} \times \exp^{\frac{-(\Delta t)}{T_a}} \qquad (4)$$

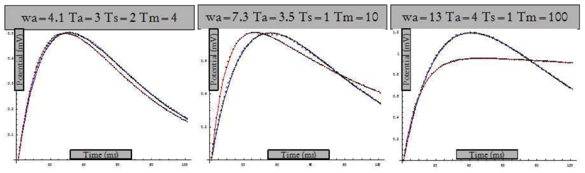

Figure 1: PSP Calculation Using Eq. 3 for ϵ (Altering Shape) and Eq. 4 for ϵ_{alt} (Same shape)

Although the alternative spike response model(Equation 4) can estimate the spike response model very closely using only the synaptic time constant, as seen in the first graph of Figure 1, certain PSP forms can only be simulated with the spike response model(Equation 3), especially when the potential has a long tail. The second and third graph of Figure 1 show the rigidness of the alternative spike response model. This motivated us to use Equation 3.

2.1.3 Dynamic Synapses

The spike response at the post-synaptic neuron is computed by summing all synaptic responses. There are two kinds of synapses that can be simulated, static or dynamic. The latter synapse is computed using dynamic memory buffers introduced by Markram. Markram et al.[8][9], developed their synapse model in order to explain the property that real neurons exhibit facilitation and inhibition (see Figure 2). Their model can make a synapse responsive to the rate(frequency), the spacing(derivative) or the sum (integral) of the incoming spike train with suitable adjustments of their parameters.

Figure 2 represents the postsynaptic potential of a neuron over time. The lines are the actual spikes produced when the threshold is crossed.

2.1.4 Hebbian Learning

$$F(\Delta t) = A_{pos} \times exp^{\frac{\Delta t}{\tau_{pos}}} \qquad if \quad \Delta t < 0 \qquad (5)$$

$$A_{neg} \times exp^{\frac{\Delta t}{\tau_{neg}}} \qquad if \quad \Delta t > 0 \qquad (6)$$

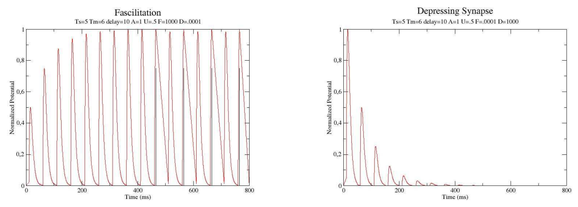

Figure 2: Facilitating and Depressing Synapse

F is the approximation of the dependence of synaptic modification on spike timing in percent shown in Equation 6 and 7. A is the maximum amount of synaptic modification when Δt is close to 0. τ is the range of pre- to postsynaptic interspike intervals over which synaptic strengthening and weakening occurs. For example, weight can be modified over the long term by adding $F(\Delta t)$ to the weight[12].

2.2 SSNNS Limitations

The first assumption in SSNNS is that our selected SRM neuron model exhibits accurate characteristics for abstract modeling and understanding. Furthermore, the specific function for refractoriness and spike response reflects our desired behavior for the artificial neuronal architecture. Second, we presume that the PSP lookup table can be scaled infinitely large using a cutoff criterion and dynamic memory allocation. Our heuristic limitation is that the last value of the PSP lookup table must be less than 0.001; otherwise the effect would be too big to ignore. Potentially, the PSP could have an unlimited tail. This raises questions about the history of spike trains. For example, what should be the maximum time of impact a spike should have? And should there be a cutoff? This simulator has built-in flexibility to explore all directions. It is not limited by biological plausibility. Hence, restrictions need to be implemented in order to claim biological relevance. On average, physiological limitations of the human brain categorize the PSP duration as being about 1-2ms long. SSNNS has no such limitations beside computer memory and cost.

Another important characteristic of SSNNS is the static threshold, it always stays 1. This is motivated by simplification and the assumption that a general threshold criterion exists. Lastly, the step size and computation in SSNNS is limited to certain step sizes. The smallest unit is one microsecond and the largest is one second.

2.3 Hybrid SNN-GA

The presented SSNNS simulator can be coupled with a genetic algorithm to evolve parameter sets for any given problem of interest. It is possible to evolve the neuron and synapse structure and topology of the network. This is work in progress.

The simulator is controlled by a set of parameters that govern the activities of each of its elements (e.g. synapses, refractory periods, time constants, etc.). These control parameters are tuned via a modified version of the CHC genetic algorithm which enables the system to produce a desired emergent pattern.

It is our hope to devise a general fitness function to evaluate the goodness of the fit. Figure 4 depicts our current fitness function. Here, solid lines represent spikes, dashed lines show the error function and dashed horizontal lines define region boundaries between spikes. The error function is simply a linear penalty.

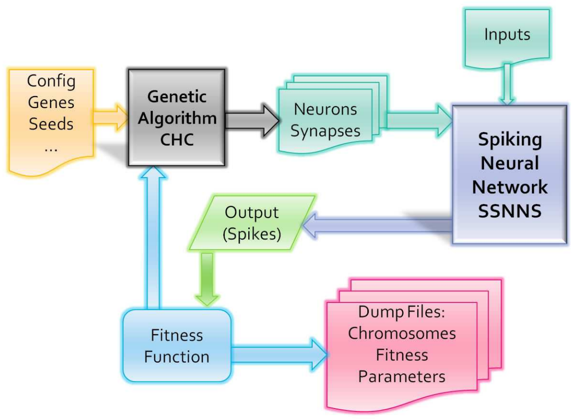

Figure 3: Hybrid SNN-GA Architecture

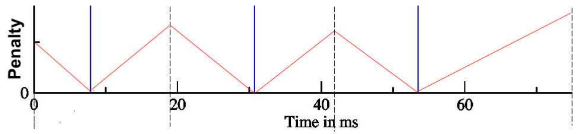

Figure 4: Fitness Function

Our system can possibly be applied to any desired problem that involves matching a given target spike pattern or simply evaluating resulting behaviors such as in robotics. Schaffer et. al[11], have given an overview of possible neural network and genetic algorithm combination applications, such as neurocontrol and those that involve sparse reinforcement as feedback.

3. RESULTS

Preliminary results indicate that our system is capable of limited learning. Using target patterns that are known to be in the domain of our system we achieved convergence to multiple solutions fairly quickly (approximately 1000 generations with populations sizes of 50). Moreover, we find that this convergence is preserved across multiple definitions of fitness.

Figure 5 shows an example spike pattern that is known to exist in our system. Solid black corresponds to neuron 1 spikes and dashed black to neuron 2 spikes. Our system can find close solutions for this sample pattern. See Figure 6 for an illustration of an evolutionary learning curve for our sample pattern. The solid line represents the average and the dotted line the best individuals. Notice that the log performance gets only close to zero. Preliminary results for other hand crafted patterns show convergence in 4 out of 10 experiments.

However, Watt's tonic buster[13] has been proven as a challenge to our system. The GA can get close but often has trouble getting the exactly right output pattern. Explorations of limited parts of these particular fitness landscapes

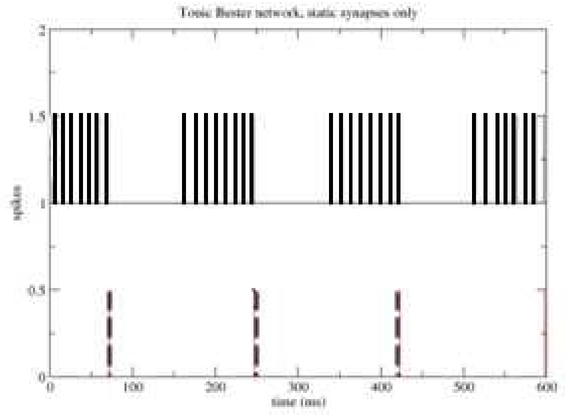

Figure 5: Spike Pattern

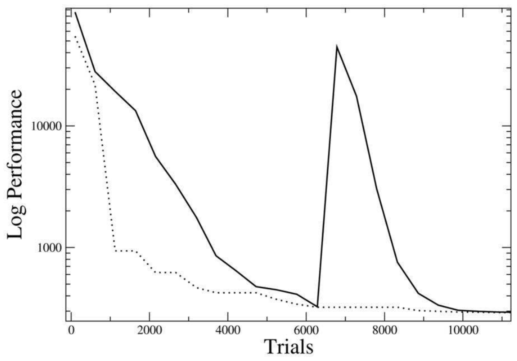

Figure 6: Learning Curve for Known Pattern

show they have a lot of problems. So, we are exploring variations on the chromosome representations and fitness functions. Additionally, we are examining more complex target patterns that may or may not be in the domain of our simulator.

4. CONCLUSIONS

Our preliminary results indicate that our system is capable of learning a limited set of dynamic neuronal spiking patterns. Additionally, we find that in our system there are multiple ways to generate similar emergent properties. As we progress to include more complex patterns we hope to identify a limiting set of the patterns that our system can learn to generate, and develop an understanding of the necessary conditions for engineering emergent behavior.

5. FUTURE RESEARCH

A major milestone will be the evaluation of our architecture with known patterns, such as Bohte's temporal XOR benchmark[2], Watt's tonic buster[13] and Jin's spike sequence recognition[6]. Furthermore, our current results indicate that it lies in the future to invent schemes for evolving the topology as well. Applications of this technology may include sound recognition or false intensive care unit alarm detection.

6. REFERENCES

[1] I. Antonov, I. Antonova, E. R. Kandel, and R. D. Hawkins. Activity-dependent presynaptic facilitation and hebbian ltp are both required and interact during classical conditioning in aplysia. *Neuron*, 37(1):135–47, Jan 2003.

[2] S. M. Bohte and J. N. Kok. Applications of spiking neural networks. *Information Processing Letters*, 95(6):519–520, September 2005.

[3] L. J. Eshelman. The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. *Foundations Of Genetic Algorithms*, pages 265–283, 1990.

[4] W. Gerstner and W. Kistler. *Spiking Neuron Models - Single Neurons, Populations, Plasticity.* Cambridge University Press, August 2002.

[5] D. O. Hebb. *Organization of behavior*. New York: Wiley, 1949.

[6] D. Z. Jin. Spiking neural network for recognizing spatiotemporal sequences of spikes. *Physical Review E*, 69, 2004.

[7] W. Maas and C. M. Bishop. *Pulsed Neural Networks.* MIT Press, Cambridge, MA, USA, March 2001.

[8] W. Maas and H. Markram. Synapses as dynamic memory buffers. *Neural Networks*, 15:155–161, 2002.

[9] H. Markram, Y. Wang, and M. Tsodyks. Differential signaling via the same axon of neocortical pyramidal neurons. *Neurobiology*, 95:5323–5328, April 1998.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel distributed processing: Explorations in the microstructure of cognition*, volume 1, pages 318–362. MIT Press, 1986.

[11] J. D. Schaffer, L. D. Whitley, and L. J. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*, pages 1–37, Philips Labs., Briarcliff Manor, NY, 6 Jun 1992.

[12] S. Song, K. D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926, 2000.

[13] L. Watts. Event-driven simulation of networks of spiking neurons. *Advances in Neural Information Processing Systems*, 6:927Ů934, 1994.