Neurodynamics

A spiking neural network implementation of path finding

Mark Saroufim
University of California at San Diego
msaroufim@eng.ucsd.edu

December 11, 2012

Abstract

We propose an implementation of Izhikevich spiking neural networks to solve a 2D path finding problem. Given a 2D grid of size nxn, we can solve the path finding problem with a spiking neural network consisting of n^2 neural populations. We propose that the activation of a population encodes the exploration of a certain state with convergence encoding the next best state. The series of next best states terminating at the destination node is indeed the shortest path to the destination. This paper discusses the theoretical foundation of this research along with a simplified experiment for a fully connected 2x2 graph graph.

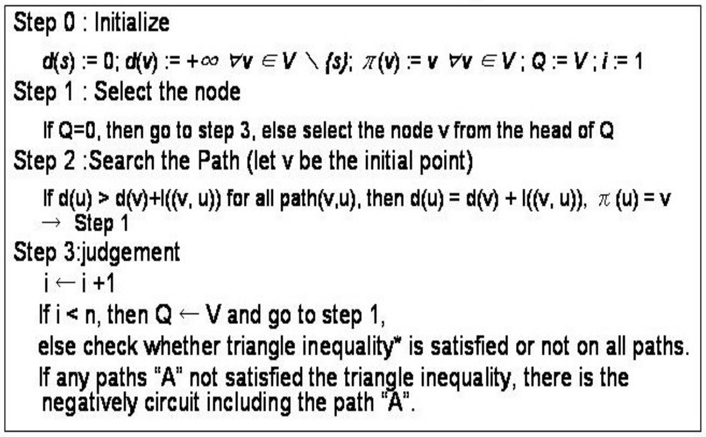
1 Introduction

Spiking neural networks (Maas and Bishop, 1999) represent an attractive alternative to neural networks in the traditional sense (Bain, 1973; James 1980). The advantages of using Spiking neural networks lie not only in the fact that spiking models are being iteratively refined by advances in experimental neuroscience but also that spiking neural networks can be implemented in neuromorphic VLSI circuits. A proposed reinforcement learning algorithm for spiking neural networks (Florian 2005, 2007) suggests that learning in the brain works by modulating spike-timing dependent plasticity (STPD) with a global reward signal which the literature suggests is the mesolimbic dopamine reward pathway (Berridge and Robinson, 1998). This project will aim to implement the Izhikevich spiking neural network inspired by Florian's proposed spiking neural network to solve a path finding problem. The path finding problem can be summarized as the problem of finding the shortest path between a source and destination. We will restrict our study to an agent moving in a two dimensional plane composed of fully connected 2x2 networks. The environment effectively looks like a fully connected graph composed of four nodes. We will thus have a population of spiking neurons for every possible state with again all populations being fully connected. The activation of a population will encode the decision of moving from one state to another. The neural populations are fully connected with excitatory synapses in a way as to create a winner takes all mechanism where only one population of neurons can be active at a time. The whole network thus learns by being reinforced or punished based on the desirability of the agent being in a certain state (a certain population vector being active). Florian's paper suggests that his model converges faster and is less sensitive to the initial condition weights of the spiking neural network. We hope that our experiment will indeed validate Florian's findings in terms of theoretical vs experimental bounds on convergence rate, network size and sensitivity to initial conditions (values of inhibitory synaptic weights). Eventually we are also interested in implementing our project on a neuromorphic VLSI circuit. Although that task will be of more concern in the winter quarter.

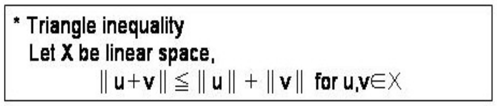
2 Proposed Methodology

2.1 The teacher

We are interested in solving a path finding problem. Given a source node s, a destination node t and a graph G we are interested in finding the shortest path between s and t. Several approaches exist to solve this problem such as Dijkstra's shortest path algorithm. However, for our purposes a dynamic programming approach to solving this problem might more appropriate which finds the shortest path to t $\forall s \in G$, All pairs shortest path in this case is known as the Bellman Ford Algorithm and is illustrated below (M. Diaz , Y. Iwai Boston University). The actual python code is included in the appendix.



```
Step 0 : Initialize
    $d(s) := 0; d(v) := +\infty \ \forall v \in V \setminus \{s\}; \ \pi(v) := v \ \forall v \in V; Q := V; i := 1$
Step 1 : Select the node
    If Q=0, then go to step 3, else select the node v from the head of Q
Step 2 :Search the Path (let v be the initial point)
    If d(u) > d(v)+l((v, u)) for all path(v,u), then d(u) = d(v) + l((v, u)), $\pi$ (u) = v
    $\rightarrow$ Step 1
Step 3:judgement
    i $\leftarrow$ i +1
    If i < n, then Q $\leftarrow$ V and go to step 1,
    else check whether triangle inequality* is satisfied or not on all paths.
    If any paths "A" not satisfied the triangle inequality, there is the
    negatively circuit including the path "A".
```



```
* Triangle inequality
  Let X be linear space,
        $\| u+v \| \leqq \| u \| + \| v \|$ for u,v$\in$X
```

Now our agent (the spiking neural network) will not be aware of the DP computations. The DP computations will be used to create a teacher that will supervise the progress of our spiking neural network agent by punishing or rewarding him appropriately.

2.2 The Student

We wish to train a spiking neural network to find the shortest path in an nxn 2D graph. The spiking neural network will consist of n^2 neural populations. The population of neural networks will be fully connected including self connections with excitatory synapses. The synaptic weights are updated using a covariance STDP rule and a reward function determined by the teacher. The covariance STDP rule was derived from [11] $\frac{d}{dt}w_{ij} \propto (v_i - (v_i))(v_j - (v_j))$.
However, what allows the student to learn is a simple learning rule we call the bytwo rule $r(t)$. Every activated population i will query the teacher to determine whether it is part of the shortest path and if it is that population will be rewarded if not it will be punished.

3 Theoretical Foundation

3.1 The Model

The model we propose adds an extra dimension to Equation 3.1 in [12]
$\Delta w_{ij} = \int_0^{\inf} A_+ \exp(-t/\tau_+) x \exp(-xt) dt + \int A_- \exp(t/\tau -) x \exp(xt) dt + r(t)$

With a reward rule $r(t)$ which can be written as

$$r_i(t) = \begin{cases} w_{ji} \leftarrow 2w_{ji} & \text{if } i \text{ on shortest path} \\ w_{ji} \leftarrow w_{ji}/2 & \text{if } i \text{ not on shortest path} \end{cases}$$

3.2 The Algorithm

```
Construct Spiking Neural Network
Initial population = Active
While destination population != Active
    $w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$
```

3.3 Other potential reward signals

We can also substitute the above reward signal to make it only increase the interconnectivity of neural populations which are on the shortest path

$$r_{i1}(t) = \begin{cases} w_{ii} \leftarrow 2w_{ij} & \text{if } i \text{ on shortest path} \\ w_{ii} \leftarrow \frac{1}{2}w_{ii} & \text{if } i \text{ not on shortest path} \end{cases}$$

We can also reward a population by doubling its number of neurons

$$r_{i2}(t) = \begin{cases} neuronnums_i \leftarrow 2neuronnums_i & \text{if } i \text{ on shortest path} \\ neuronnums_i \leftarrow \frac{1}{2}neuronnums_i & \text{if } i \text{ not on shortest path} \end{cases}$$

We could also double the number of synapses

$r_{i3}(t) = synapsenums_{ij} \leftarrow \frac{1}{2}synapsenums_{ij}$ and $synapsenums_{ji} \leftarrow 2synapsenums_{ji}$
if i on shortest path

And Finally we could double the probability of synapses to the next best state of releasing their vesicles

$r_{i4}(t) = pr(synapse)_{ij} \leftarrow \frac{1}{2}pr(synapse)_{ij}$ and $pr(synapse)_{ji} \leftarrow 2pr(synapse)_{ji}$
if i on shortest path
and if $pr(synapse) > 1$ $pr(synapse) \leftarrow 1$

All of these rules have one thing in common they make the most desired state/population the most likely to be activated. However, it is not biologically plausible that the brain uses just one of these reward signals what is more likely is some linear combination of reward signals that modify different parameters of the spiking neural network
$r(t) = \sum_d w_d r_{id}$

3.4 Correctness Proof

To prove that the algorithm is correct we will prove two things, the first is that the algorithm indeed converges and the second is that the algorithm can only terminate at the destination node. A reasonable assumption for a path finding problem is that a source node exists or within the framework of our model that the external current will excite an arbitrary neural population. Now given the existence of an initial state, the key is our bytwo reward rule $r(t)$. We will consider a 2x2 fully connected grid to be the base case of our recursion. The full problem is an nxn grid consisting of these 2x2 connected grids but the nxn grid is not fully connected. Initially the initial state population will excite all of its neighbors, in the case of the 2x2 grid it means all the neural populations will be excited.

Definition: Let pop_i define population number i
Let $N(pop_i)$ define the neighbours of population i

pop_i will activate its neighbours $N(pop_1)$ given two things:
(1) pop_i was activated
(2) The excitatory synapses to $w_{pop_i N(pop_i)}$ are strong enough to activate $N(pop_i)$

In this way pop_i will spike $\forall i$. However, in the case of the 2x2 grid only the destination node will spike because at the limit where d is the destination population.
$\lim_{n->\inf} w_{dd}, w_{id} = \inf \, \forall i$ where d is the destination node
$\lim_{n->\inf} w_{ij}, w_{ii} = 0 \, \forall i \neq d$

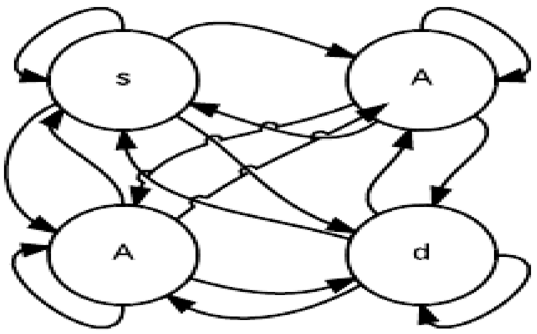
We can thus without loss of generality say that this algorithm will converge iteratively on the next best node until it converges at the destination node (reinforced by the DP teacher)

3.5 Upper Bound Time Analysis

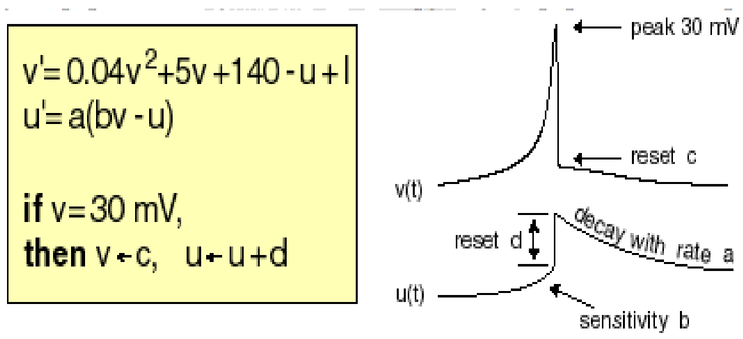
Running the Bellman Ford algorithm on the nxn grid takes $O(|V||E|)$ where $|V|$ is the number of edges in the graph and $|E|$ is the number of edges in the graph. In a fully connect graph $|E| = |V|^2$ so the running time of the Bellman Ford Algorithm boils down to $O(|V|^3)$. It should be worth mentioning that the Bellman Ford algorithm will be run only once for the purpose of designing our teacher.
As for the spiking neural network, it can converge for a 2x2 grid in $O(\log(\max(w_{dj})))$ where d is the destination neural population and j is all other neural populations. Therefore even if our 2x2 grid was initialized with very large weights such as 10^9 then because of the log our spiking neural network will only need 9 iterations to converge. To solve the full nxn path finding problem we will need to solve a 2x2 grid problem n times. The total running time is upper bounded by $O(n \log(\max(w_{dj})))$

4 Experiment

In our experiments we first dealt with the base step of our ”recursive” algorithm. Our algorithm is conveniently greedy which means that a local solution to our base step is on the shortest path to the destination. Below we can find a graphical representation of our unit a 2x2 fully connected grid where s represents the population where the input current was placed and d represents the destination state/population



We simulated 4 fully connected neural populations of varying sizes with excitatory synapses of varying sizes between them. The neural model we used is the Izhikevich neuron shown below [13]. We start by injecting an external current into one of the neural populations and record the activity of all four neural populations over time. We tried to simulate the results on python but it appears that BRIAN was not designed to allow manual dynamic changes in synaptic weights. The workaround is to integrate the bytwo rule into a modified stdp and use the synapse module instead of the connection module.

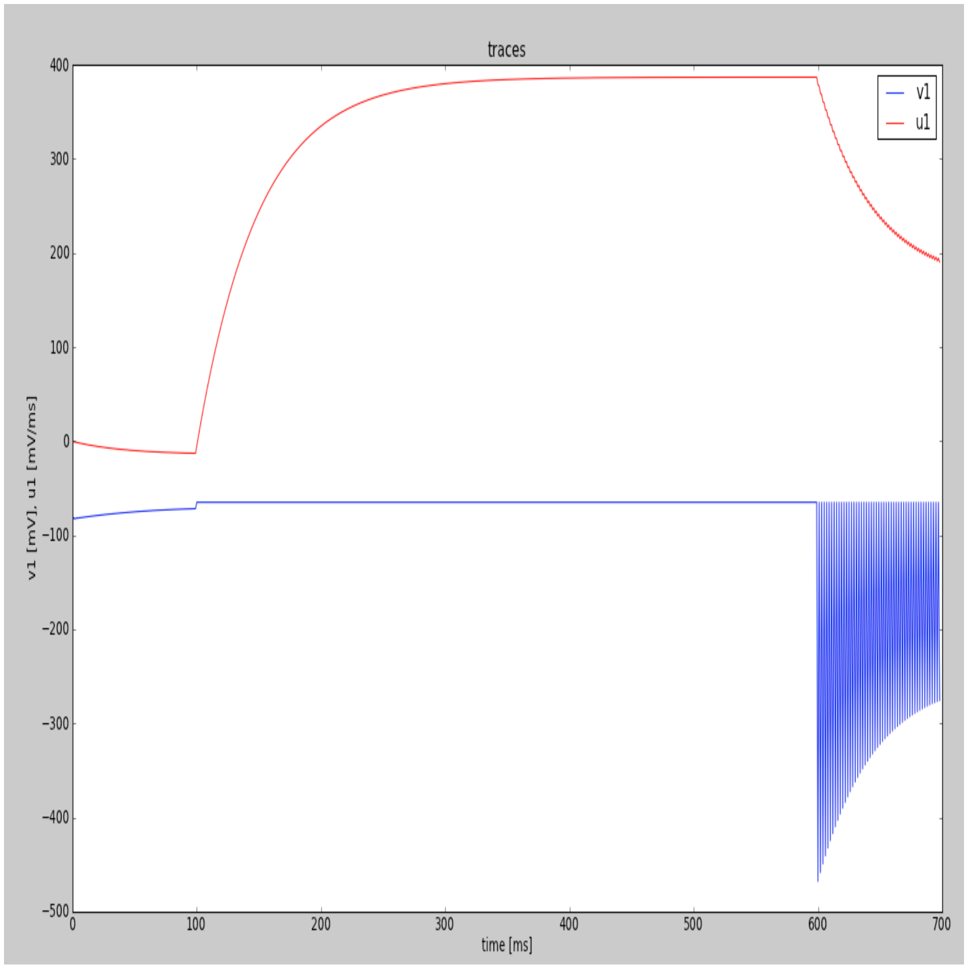


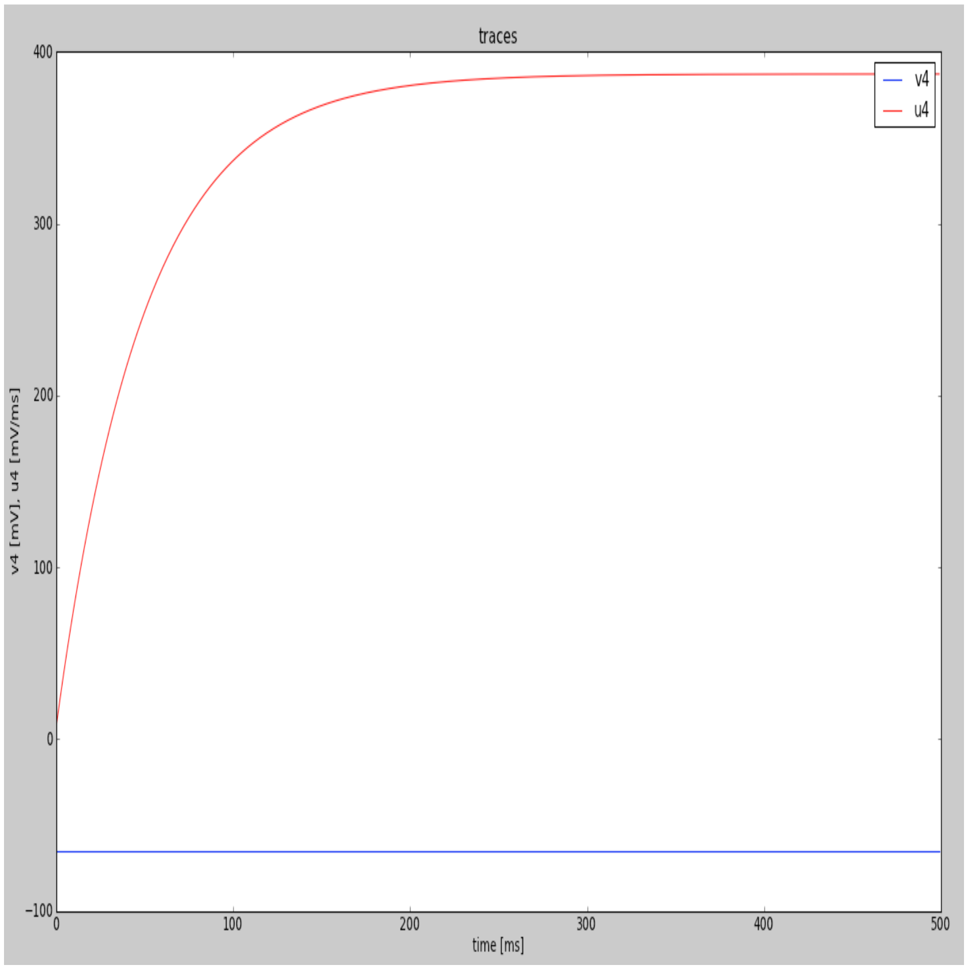
Electronic version of the figure and reproduction permissions are freely available at www.izhikevich.com.

5 Results

Note: The simulation code is available and free for distribution.
The two plots below show how membrane voltage and membrane recovery variable vary with time after an external current was placed on population 1. At time = 100 ms we inject an external current into population 1 which causes a spike that eventually gets inhibited at time = 600 ms. In the meantime, the membrane voltage of the fourth/destination population spikes and remains active through the experiment. This indeed confirms the validity of our model for a 2x2 grid as we can clearly see how the first starting population is eventually inhibited and how the fourth destination population becomes and remains active encoding the spiking neural network's awareness of being in the next best state and in the case of our 2x2 grid in the best state. This approach works for an

arbitrarily sized fully connected graph but the experiments are not yet sophisticated enough to deal with more difficult types of graph.





6 Acknowledgements

We'd like to thank Emre Neftci for his valuable insight and expertise in spiking neural networks.

7 Conclusion

We have proposed an implementation of spiking neural networks to solve a 2D path finding problem. Using STDP and a simple bytwo reward rule. The presence of a teacher is biologically plausible since we could argue that a rational teacher could evolve given enough time through random mutations. The proposed algorithm can thus find a path in a graph, the experiment shows how this algorithm can find a path in a fully connected nxn graph. Future work will deal with making the graphs more realistic with obstacles. Also it is worth noting because synaptic weights increase and decrease exponentially because of the bytwo rule, membrane voltage can becomes unrealistically large. Computationally this approach works but the physiology of dendrites dictates that membrane voltage is always within a certain range.

8 Bibliography

[1] A. Bain (1873). Mind and Body: The Theories of Their Relation. New York: D. Appleton and Company. [2] W. James (1890). The Principles of Psychology. New York: H. Holt and Company.

[3] L.F. Abbott and W. Gerstner, ”Homeostasis and Learning through Spike-Timing Dependent Plasticity”, Methods and Models in Neurophysics, Elsevier Science, 2004.

[4] Goodman DF and Brette R (2008) Brian: a simulator for spiking neural networks in Python. Front. Neuroinform. doi:10.3389/neuro.11.005.2008

[5] W. Maass and C. Bishop (1999) Pulsed Neural Networks ISBN 0-262-13350-4

[6] G. Cauwenberghs, K. Kreutz-Delgado, T. Sejnowski, M. Arnold, S. Deiss, J. Murray, N. Schraudolph (2011) Robust Adaptive Large-Scale Neuromorphic Systems in Human-Machine Sequential Games IIS Core Program in Robust Intelligence

[7] K. Berridge , T. Robinson (1998) What is the role of dopamine in reward: hedonic impact, reward learning, or incentive salience? Bain Research Reviews 28

[8] R. V. Florian (2007), Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. Neural Computation 19 (6), pp. 1468-1502.

[9] R. V. Florian (2005), A reinforcement learning algorithm for spiking neural networks. In D. Zaharie et al. (eds.), Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005), Timisoara, Romania, pp. 299-306. IEEE Computer Society, Los Alamitos, CA.

[10] Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. Cambridge,MA: MIT Press.

[11] L.F. Abbott and W. Gerstner Homeostatis and learning through spike-timing dependent plasticity

[12] E. Izhikevich, N. Desai (2002) Relating STDP to BCM , Letter Communicated by Sejnowski

[13] E. Izhikevich Simple Model of Spiking Neurons, IEEE Transactions on Neural Networks (2003) 14:1569-1572