Using a Harmony Search Algorithm/Spiking Neural Network Hybrid for Speech Recognition in a Noisy Environment

David Reid
Liverpool Hope University
reidd@hope.ac.uk

Mark Barrett-Baxendale
Liverpool Hope University
barretm@hope.ac.uk

Abstract

The Harmony Search Algorithm (HSA) is a recently constructed algorithm that is based on the improvisation process that musicians often demonstrate in performances [1],[2]. In the HSA each musician plays a note in an effort to find the best harmony with his/her fellow musicians. In the context of Computer Science, this improvisation process corresponds to a decision variable (musician) generating a value in order to find a best global optimum solution (a pleasing harmony). The HSA involves three possible choices; repeating a tune or pattern exactly from memory, repeating a tune or pattern with slight variation of one or two variables (adjusting the pitch slightly), or using new random values or notes. Geem [1],[2] formalized these processes as harmony memory, pitch adjusting, and randomization. Recently the HSA has been applied to neural networks. This has been done in two ways. Firstly, by modification of the weights of the neural network by using the HSA as the network is trained, and secondly by using the HSA as a post-processing tool as a way to reduce the chances of selecting a sub-optimal solution.

We propose using the HSA in a new way and on a new generation of neural networks. Spiked Neural Networks (SNN) [3],[4],[5] encode information according to the temporal differences and correlations between spike trains. This new type of neural network is not only biologically more realistic than other neural networks; we propose that SNN have a natural symbiosis with the HSA.

Keywords: harmony, search, spiking neural, speech, recognition, hybrid.

1. Harmony Search Algorithm

The Harmony Search Algorithm (HSA) [1],[2] is an algorithm inspired by the improvisation processes that a group of musicians undertake when they perform together. In the HSA each musician plays a note in order to find the best harmony at a particular point in time. In the context of Computer Science this is equivalent to a number of decision variables (musicians) generating values (notes) in order to find a global optimum solution (harmony).

The mechanics of producing a pleasing harmony are as follows:

1. ***Memory Consideration***
   This involves choosing a note (or decision variable) from memory. This is often called the Harmony Memory Consideration Rate or HMCR.
2. ***Pitch Adjustment***
   This involves choosing a note (or decision variable) from memory and then altering it by a small amount (within set bounds). This is often called the Pitch Adjustment Rate or PAR.
3. ***Random Selection***
   This involves playing a random note (or generating a new decision variable) within set bounds. This is often called Random Selection or RS.

The steps to perform a HSA are as follows:

Step 1)

Initialize the problem by providing an objective function. The HSA tries to find a vector *X* of size *n* that matches this objective function. Thus we generate *k* random vectors of *X* where k is equal to the harmony memory size. Therefore harmony memory (HM) can be depicted as follows:

$$HM = \left[\begin{array}{ccc|c} x_1^1 & \cdots & x_n^1 & f(x^1) \\ \vdots & \ddots & \vdots & \vdots \\ x_1^k & \cdots & x_n^k & f(x^k) \end{array}\right]$$

Where $f(x^k)$ is the fitness function of the vector *k.*

Step 2)

Generate a new vector x' for each component x_i' by performing the three actions as defined above, they are:

1. Memory Consideration

Choosing a vector x' with the probability based on the harmony considering rate (HMCR) from HM.

2. Pitch Adjustment:

With a probability based on the Pitch Adjustment Rate (PAR) change the value of x_i' by a small amount:

$$x_i' \leftarrow x_i' \pm \delta$$

for discrete variables and

$$x_i' \leftarrow x_i' \pm fw.rand(0,1)$$

for continuous variables

where δ is the amount between two neighbouring values and *fw* is the maximum limit of change.

3. Random Selection:

With a probability based on a Random Selection (RS) rate change x_i' to an allowable random value taken from HM.

Step 3)

Update HM. If x' is better than the worst vector in HM then replace it with x'

Step 4)

Check the termination criteria and repeat steps 2-4 if criteria not reached.

In this way the HSA mimics the composition rules of musicians in order to search for the best solution to a problem by using a fitness function. This overcomes many of the drawbacks of a Genetic Algorithm [6],[7] in that it considers the relationship between values in a more sophisticated manner. That is, the Genetic Algorithm only works if the relationship in a chromosome are carefully considered. (Often Gray coding is used for real values). The HSA does not suffer from this limitation as neighbouring notes/values can be any value within the allowable range and the values themselves say little about their relationship. Moreover the HSA, unlike the Genetic Algorithm, has implicitly embedded within its workings the concept of an explicit temporal memory. These ideas linked to the randomization functions means that the HSA is a more robust mechanism with which to minimize the chances of a solution suffering from divergence or settling on local minima.

2. Spiking Neural Network

Spiking neurons [3],[4],[5] are a third generation of neural networks that use temporal data in order to code information. This new type of neural network can be defined as follows:
Let N be a set of spiking neurons. Let P be a set of pre-synaptic neurons such that $P \subseteq N$, where the axon of each $p_j \in P$ makes contact with the dendrites or soma of each neuron $n_i \in N$ to form a synapse. A weight $w_{pj,ni} > 0$ is given for a synapse, a response function $\varepsilon_{pj,ni} : R^+ \to R$ for each synapse and a threshold function $\Theta_{ni} : R^+ \to R$ for each neuron.

Thus, there exists a set $S = \bigcup_{k=1}^{K} \{s_k\}$ of K synapses, with each synapse s_k forming a connection between each p_j to each n_i i.e. $s_k \subseteq PxN$. Each n_i receives a spike at a specific time t from each of the pre-synaptic neurons p_j.

The synapse s_k receiving this spike will have an inhibitory or excitatory response (figure 1(a)). This is the Post Synaptic Potential (PSP).

If the neuron has a threshold set to some specific value T_i then it will fire if at time t $\sum_{n=0}^{k} s_n \geq T_i$ (figure 1(c)), whereas if this value is less there is no firing (figure 1(b). When firing does take place there follows a relative refractory period when the possibility of firing again is diminished and also an absolute refractory period where there is no possibility of firing.
The PSP function usually approximates the synaptic function by using a sharp exponential rise and slower decay in response to each spike. In practice the following equation is a good approximation:

$$\varepsilon(s_k) = \frac{1}{1-(\tau_{s_k}/\tau_{n_k})}\left[\exp\left(-\frac{s_k}{\tau_{n_i}}\right) - \exp\left(\frac{s_k}{\tau_{n_i}}\right)\right] H(s_k)$$

Equation 1.

where $H(S_k)$ is a Heaviside step function which vanishes for $s \leq 0$ and has the value of 1 for $s > 0$, τ_{S_k} is a synaptic time constant and τ_{N_i} is a neuronal time constant.

$$\varepsilon(s_k) = \frac{s_k}{\tau_{S_k}}\left[1 - \exp\left(-\frac{s_k}{\tau_{S_k}}\right)\right]$$

Equation 2.

In practice we have found that equation 2 is an adequate approximation. Learning can be both supervised and unsupervised in such a network. Learning in the supervised network is usually implemented using the SpikeProp method [8]. This is very similar to the error-backpropagation method by Rumelhart [9] except that the firing time of the neuron n_i is shifted by altering its refractory period(s). Thus, learning is achieved by temporally co-coordinated firing of related neurons.

.

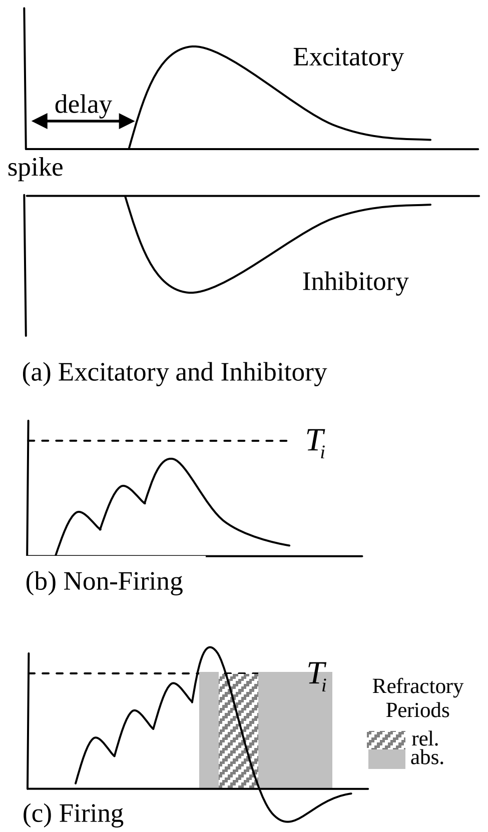

Figure 1. Spiking neuron activity

Unsupervised learning usually involves selecting a neighbourhood around a winning neuron and altering the firing times of its neighbours to fire in sympathy with the winning neuron. Again, as with SpikeProp, the firing time is shifted toward the winning neuron. Distant neurons' firing times are moved away from the firing time of the winning neuron. A typical algorithm uses the physical distance to determine how firing times should be synchronized with the winning neuron. In SNN a number of coding schemes exist. They can be broadly classified as follows:

rate coding

This assumes that the information is contained in the firing rate of the neuron. Information about the stimulus is implied by the firing rate. As the firing rate generated by a given stimulus varies from trial to trial, neuronal responses are usually characterised probabilistically using this coding scheme. In this coding scheme individual spikes are less important than the rate of spiking activity. This has the advantage that it is a highly robust (in terms of noise) coding scheme but has the disadvantage that it is also highly inefficient.

temporal coding

A temporal code is when precise, or approximate, spike timing is assumed to carry information.

A number of studies have found that the inter-spike intervals can be detected on a millisecond time scale in the brain. This strongly suggests that precise spike timing is a significant element in neural coding. Temporal coding refers to temporal precision in the response that also says something about the stimulus. The advantage to this type of coding is that it is efficient. Information can be carried as the length of time between just two spikes, or even the first stimulus to spike.

A number of coding techniques exist in the literature[10],[11], first time to spike, inter spike interval, synchronous spiking and spike interval and phase encoding (where repeating patterns of activations between neurons is used as a way of encoding information) to name but a few.

However, the interplay between stimulus and encoding dynamics makes the identification of a temporal code very difficult. Despite this, because of their efficiency, temporal coding techniques are by far the most popular method of spike coding used in SNN research.

population coding

This coding technique takes into account the coordination of activities of a number of neurons. In population coding, each neuron has a distribution of responses over some set of inputs, and the responses of many neurons may be combined to determine some value. Not only is this biologically realistic (the motor and sensor areas of the brain are well known to use this coding method) but it also echoes back to older neural network research involving self-organising maps or topological feature maps.

3. HSA-SNN Hybids

It is suggested that the three basic processes in the HSA described earlier by Geem are natural candidates for a mechanism to correlate, classify and control (at least) two aspects of SNNs; that is, the spike trains and the patterns of activation of the synapses themselves. Moreover the HSA functions can be integrated into the normal working of a neuron, or groups of neurons, within a SNN with only slight modification to the neuron's traditional functionality.

harmony spike train tuning

As previously suggested, we propose that neurons in a SNN not only perform a simple leaky integrate and fire function (LIF), but also can naturally utilize the three basic processes used in the HSA. This can be done by classifying spike trains using the HSA and by regulation of those spike trains by the HSA.

In this sense pitch adjustment could correspond to modification of the relative, or absolute, refractory period of the neuron, thus the allowable rate of frequency of spikes would directly correspond to pitch. This is functionally the same as rate encoding and temporal encoding as discussed earlier. This similarity also extends to randomization in the HSA hybrid which would simply be the inclusion or exclusion of spikes. Randomization, as with the traditional HSA, would ensure that a wide variety of alternatives are explored within the neural solution space. Most importantly, the incorporation of harmony memory into the spiking neuron allows the neuron to recognize/remember previous combinations of spike trains (either temporally or by their rate). This can be done in two ways; firstly, by the simultaneous control of neuronal connections. To this end a new Multi-Spiking Neural Network model could be used [12]. In this model information from one neuron is transmitted to the next in the form of multiple spikes via multiple synapses (a rate encoding method; SpikeProp learning using HSA). Secondly, a more sophisticated model than LIF for the neuron itself could be used. Gestner describes the Cumulative Spike Response Model [13]. In this model refractoriness and adaptation were affected by the combined effects of the spike action potentials of several previous spikes, rather than, as with LIF, only the most recent spike. Similarly, Gerstner also describes how noise (randomness) can be included into the Spike Response Model by replacing the strict threshold criterion by a stochastic process [14]. Such non LIF based models of the neuron can easily incorporate harmony memory processes by allowing some preprocessing in the neuron; in other words by using the HSA to process the combined activation of a number of spikes received in a neuron before making a decision about how that neuron should react (what should its threshold be set to, what should its refractory period be etc). In other words the neuron learns how to behave when given a particular set of spike combinations by using the HSA to attune itself to that circumstance. For example in Figure 2 below the repeated pattern labeled ***a*** in the incoming spike train may cause a change in the values depicted by any of the arrows in the PSP.

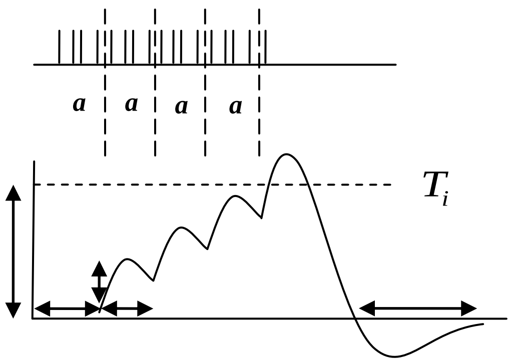

Figure 2. PSP Influenced HSA

harmony active synapse tuning

At a courser grained level, the HSA can, as with spike trains described above, be used to classify the synaptic activity and also be used to regulate the synaptic activity within a system. Gütig used this idea in the "temptron" [15] that has its theoretical roots in population encoding as described earlier. Gütig describes a biologically plausible synaptic learning rule that enabled neurons to learn specific decision rules even when information was deeply embedded into the spike patterns. This work underlined how a system could be built that had a high capacity to decode information embedded in the distributed patterns of spike synchronicity. It is suggested in this paper that not only is rate coding and synchronicity characteristics that are well catered for by the HSA (in fact these are fundamental characteristics of the functioning of the HSA) but also that other spatiotemporal synaptic activity, such as inter spike interval and phase encoding, as well as population encoding, are also very well served by the HSA.

For the purposes of this paper we focus on Harmony Active Synapse Tuning using the population encoding method (although the other methods suggested in this paper are equally valid potential areas of research).

4. Implementation

A hybrid HSA-SNN was built for this paper that uses Harmony Active Synaptic Tuning. This uses a "mesh" of connected spiking neurons that are connected to their local neighbours and also connected to input neurons that delimit the cochlear input frequencies. This is shown in Figure 3.

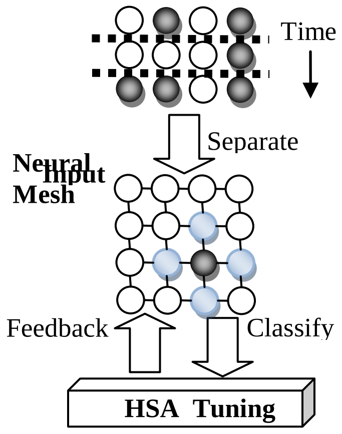

Figure 3. Neural Influenced HSA

In Figure 3 the input "cochlear" (in this case only corresponding to 4 frequency ranges) fire at different times, each time transmitting a spike to the neural "mesh". The "mesh" synapses compete for activation by suppressing neighbouring neurons activation and heightening their own sensitivity to the specific patterns of activation form the input "cochlear". Both the synapses connected to the input cochlear neurons and the neural activation of the "mesh" neurons are subject to, and influenced by, the HSA. In this way the input synapses, and by implication the synapses connected to each neurons neighbours, are tuned by both spike activation and by HSA mediation.
We used this architecture in order to classify simple instructions to control a computer game/robot functioning in a classroom environment. In this system we used a SNN to respond to different frequencies in order to pre-process the speech signal in a similar fashion to the cochlea. The HSA uses the firing rates of the discrete neurons in order to recognize the spoken words. This was done to allow the SNN and the HSA to function together in order to provide both an effective noise filter whilst simultaneously providing a robust recognition system.
The system described was written in the Processing language and as a preliminary test was used to classify different patterns of on a 8 bit "cochlear" (corresponding to 8 frequency ranges). A simple 8x8 mesh of neurons was used to classify patterns emanating from the input cochlear. As a initial experiment, primarily in order to test the basic code functionality, the words used where the simple utterances "up", "down", "left", "right", "stop" and "fire". These where recorded as different combinations of bit patterns on the 8 bit input "cochlear" where each bit position represented a specific frequency. These patterns where then classified by competitive activation of the neural mesh and by feedback from the HSA.

In the system the HSA was used for two basic functions.

1. To filter, separate and "pre"-classify the input spike trains, making it easier for the subsequent competitive SNN to recognise the result
2. To use the HSA to classify the activation of firing of the synapses in the neural mesh itself.

The screenshot in Figure 4 shows the system in action. Here the word "up" has been classified in the mesh by using a competitive SNN using HSA tuning. In this particular example the HSA has filtered the spike trains by classifying the activity of each input bit frequency during the training of the system (up was entered 10 times). As result of this the neuron 21 and 26 are maximally activated. It is these combinations of neuron activations that are subsequently used in the HSA in order to perform the final classification (in this experiment this training was also performed continuously).

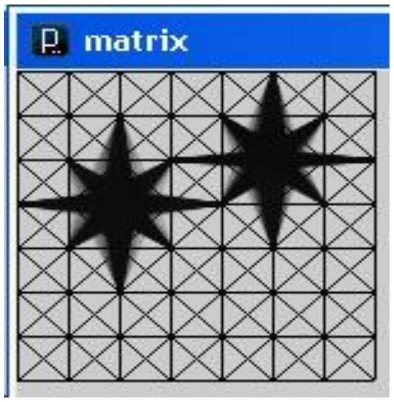

Figure 3. The HSA-SNN System

This very simple prototype serves two purposes. Firstly, in testing and exploring the symbiosis of the two technologies (HSA and SNN); secondly, in showing that this application area suited this symbiosis. On both counts the engine has indicated that both of these criteria have been met. Similarly experimentation with the engine has suggested that the model could fairly easily be extended and applied to other application areas.

5. Conclusion

The paper has demonstrated that not only is a HSA SNN hybrid a feasible symbiosis of technologies but that it is a natural one. Both technologies can be combined in an effective manner to create a potentially powerful classification engine. This is principally because both technologies use at their core the temporal nature of the data as a significant part of their functionality. While the implementation provided for this paper was a “proof of concept” it is envisaged that these ideas could be easily extended. This could be done in two ways; firstly by extending the model presented here. This could be done altering the parameters of the engine, for example altering the number and behavior of the SNN or the characteristics of the HSA; and secondly, by parallelizing the engine. At the moment we are particularly interested in implementing the model on a XMOS system in XC. Similarly we hope to implement the model in an FPGA using a Virtex 6 board in HandleC and investigate running the model on GPU using OpenCL.

The example used in this paper is very small scale. It is a very simple speech recognition task. However, the engine itself can be applied to a number of other types of application. At present we are examining extending this engine for a novel vision system, for use in self organizing robots and for novel data mining and business intelligence applications. In short, now we have seen that the simple prototype engine confirmed that symbiosis between the technologies is a potentially profitable one we are currently extending the engine into different application areas. However, for the present time we are currently translating the Processing code into XC code for parallelization on XMOS chips in order to build a more sophisticated speech recognition engine.

6. References

[1] Geem, Z. W., Kim, J. H.,Loganathan, G. V., A New Heuristic Optimization Algorithm: Harmony Search, *Simulation*, 2001.

[2] Geem, Z. W., Improved Harmony Search from Ensemble of Music Players, *Lecture Notes in Artificial Intelligence*, 2006.

[3] Maass, W., Networks of Spiking Neurons: The Third Generation of Neural Network Models, *Neural Networks, Volume 10*, Issue 9, Elsevier Publishing, Pages 1659-1671, December 1997.

[4] Maass, W., Bishop, C. M., Pulsed Neural Networks, MIT press, ISBN 0-262-13350-4, 1998.

[5] W. Mass, On the computational complexity of networks of spiking neurons, *Advances in Neural Information Processing Systems*, 7, 1995, pp. 183-190.

[6] Mahdavi M., Fesanghary M, Damangir E., An Improved Harmony Search Algroithm for Solving Optimization Problems, *Applied Mathematics and Computation 128*, (2007) 1567-1579.

[7] Lee K, S., Geem Z., W., A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice, *Comput. Meth. Appl. Mech. Eng. 194* (2004) 3902–3933.

[8] Bohte S., M., Kok J., N., Poutre H., L.., Error-backpropagation in temporally encoded networks of spiking neurons, *Neurocomputing (48),* 2002, pp. 17–37.

[9] Rumelhart D., E., Hinton G., E., Williams J., E., Learning representations by back-propagation errors, *Nature 323*, 1986, pp. 533-536.

[10] Theunissen F, Miller JP. Temporal Encoding in Nervous Systems: A Rigorous Definition. *Journal of Computational Neuroscience*, 2, 149—162; 1995.

[11] Rieke F, Warland D, de Ruyter van Steveninck R, Bialek W. Spikes: Exploring the Neural Code. Cambridge, Massachusetts: The MIT Press; 1999.

[12] Ghosh-Dastidar S., Adeli H., A New Supervised Learning Algorithm for Multiple Spiking Neural Networks with Application in Epilepsy and Seizure Detection, *Neural Networks Volume 22*, Issue 10, Elsevier Publishing, Pages 1419-1431, 2009.

[13] Gerstner W., Van Hemmen J.L., Cowan JD, What matters in neuronal locking, *Neural Computation, Volume 8*, Pages 1653-1676, 1996.

[14] Gerstner W., Kistler W.M., Spiking Neuron Models - Single Neurons, Populations, Plasticity, Cambridge University Press, ISBN 0-521-89079-9, 2002.

[15] Gütig R., Somplinsky H., The Tempron: A Neuron that Learns Spike Timing-Based Decisions, *Nature Neuroscience 9*, Nature Publishing Group, 2006.