Counting with Neurons: Rule Application with Nets of Fatiguing Leaky Integrate and Fire Neurons

Christian R. Huyck (c.huyck@mdx.ac.uk) and Roman V. Belavkin (r.belavkin@mdx.ac.uk)
Middlesex University, London NW4 4BT, UK

Abstract

This paper shows a system that performs simple symbolic processing. The system is based entirely on fatiguing Leaky Integrate and Fire neurons, a coarse model of neurons. Following Hebb, the symbols are encoded by neurons that form Cell Assemblies. Additionally simple rules of the form $if X \rightarrow X+1$ are encoded by Cell Assemblies, and this symbolic computation is performed. Finally, a more complex rule $while X < F \rightarrow X = X + 1$ is encoded using variable binding via a compensatory learning rule. This rule performs the symbolic computation of counting entirely subsymbolically. The binding can be erased and reused via spontaneous neural activation. Unlike the symbolic parallel, the counting rule fails at times when humans might fail.

Introduction

Neurons have inspired a range of connectionist models including multi-layer perceptrons, self-organising maps, and Hopfield nets. These connectionist models are useful for a range of tasks, but they make simplifications that may invalidate them as neural models. For example, most multi-layer perceptron (MLP) systems learn by the supervised algorithm of error back propagation, while neural learning is unsupervised. One work, [Wiles and Elman, 1995], manages to count with a recurrent MLP network, though it is not clear how this dynamical systems model relates to neural processing.

There is a wide range of models of neurons, but they are usually used to model neural behavior and are rarely used for higher cognitive tasks. This paper describes a system based on model neurons that simulates a simple symbolic processing task.

There are a range of cognitive architectures based around rules (e.g. ACT–R [Anderson and Lebiere, 1998], SOAR [Laird et al., 1987], and EPIC [Kieras et al., 1997]). These have proven quite successful in modelling psychological data and for applications.

An alternative approach to the rule based model is a neural model. Neural models are more accurate descriptions of biological neurons than other connectionist models. It is unclear how or even if cognition is implemented by rules. It is however clear that cognition can be achieved by neurons. The unanswered question is how cognition is generated by neurons.

The scientific community has a sound if incomplete understanding of the function of biological neurons, but there is much less known about how they work together. This is at least partially due to the difficulty of inspecting a large number of neurons in a functioning animal. Scanning techniques, such as fMRI, are too coarse to view individual neural behavior. So computational modelling is a good method to explore large numbers of neurons working together.

While there have been advances in implementing ACT–R in a connectionist system [Anderson and Lebiere, 1998], it is not clear how it would be implemented in a neural system. In particular, it is not clear how the connections between the neurons could be learned.

There is a long-standing neurally based psychological theory. Hebb proposed Cell Assemblies (CAs) as the basis of human thought over 50 years ago [Hebb, 1949]. There has been little large scale modelling of CA using neurons. While it is relatively well understood how to categorise inputs using CAs and other attractor networks, it is not clear how more sophisticated processing can be done. For instance, it is not well understood how symbolic processing can be implemented with CAs.

In this paper we describe a neural system that simulates a form of symbolic processing. Simple arithmetic is a common symbolic task. The simulations first implement a series of simple rules that perform an add one calculation. This requires rules that only use Input CAs that represent constants. These simple rules are then used as a basis for a second series of simulations that count. This counting makes use of variable binding.

The simulations are based on fatiguing Leaky Integrate and Fire (LIF) neurons, a reasonable model of biological neurons. These simulations show a neural implementation of two sorts of rules. Simple rules involving primitives are used to add. A complex rule using variable binding is used along with the simple rules to count. Later bindings are affected by earlier bindings and may lead to problems in counting that might be exhibited by a human.

The paper initially gives a background of work on CAs. There is then a description of the simple rules, followed by a description of the system that counts. These sections are written for the uninitiated neural modeller and are followed by a section that specifies details of the model. The paper concludes with a discussion.

Background

Hebb proposed CAs as the basis of human thought [Hebb, 1949], and there has been a long history of work based around CAs (e.g. [Sakurai, 1998, Palm, 1990]). The basic idea is that concepts are represented by groups of neurons, called CAs, that have high mutual synaptic strength. If enough of these neurons fire, a cascade of neuronal firing causes the reverberating circuit to remain active; this is called CA ignition. After the stimulus ceases, the circuit can still remain active via this reverberating activity.

There are a wide range of benefits of the CA model. Perhaps the most powerful benefit over rule based models is symbol grounding [Fodor, 2000]; symbols can be learned from the environment and thus have a basis.

Hebb's main argument for this type of model was figure-ground separation. CAs are ignited for particular items in the environment (figures) making them more salient, and these salient figures can then be separated from the background.

A third advantage is that CAs give a neurally based explanation for long and short-term memory. A short-term memory is the ignition and persistence of a CA; as long as the activity persists the memory is active. Long-term memories happen when the CA is formed via synaptic weight adjustment.

A CA is an attractor state; most configuration of neural activations never occur. Instead, when neurons are activated, particular neural activation patterns are preferred. These patterns or states are called attractor states. A CA is an attractor state. So, a network of fatiguing LIF neurons is an attractor net. There has been significant work with attractor nets such as Hopfield nets [Hopfield, 1982], and these attractor nets have been used to model the brain [Amit, 1989].

Attractor nets, and thus nets of fatiguing LIF neurons, are good at categorisation. An initial state is given to the network, and it settles into a stable state that represents the category of the initial input.

Unfortunately, there has been little work with getting attractor nets to do more than categorise. A notable recent exception has a CA based system being used for robots that have visual and textual input [Knoblauch and Palm, 2001].

While there has been a long history of CA-based models that account for psychological phenomena, there has been little work in developing large scale neural simulations of CA activity. These systems are based on idealised models of CA behavior. A good example of such a model is the TRACE system [Kaplan et al., 1991]. Another model shows how a letter matching task might be done [Dalenoort, 1985].

The system described below can be thought of as a bridge from older idealised CA models for psychological tasks to neural processing models of CAs. The neural models are capable of learning symbols based on environmental stimulus [Braitenberg, 1989, Huyck, 2004, Palm, 1990, Sakurai, 1998].

Simple Rules

It may not be clear to those who commonly develop software using rules, but there are different types of rules. In this section, a system that implements several simple rules of the form $if 1 + 2 \rightarrow 3$ is presented. In the next section an extension that counts is presented. The simulations use fatiguing LIF neurons. There has been considerable interest in LIF neurons (e.g. [Tal and Schwartz, 1997]), though comparatively little work has made use of fatigue. The model we use is a relatively simple model of neurons, but to a large degree it is biologically plausible.

The simulation sections are written for the neural model novice. Details of the model and simulations are provided in

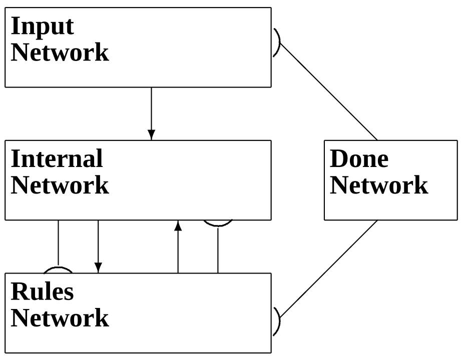

Figure 1: Simple Rule Topology

the section entitled Details of the Model.

In the simple rule simulation, there are four networks used in the process of rule application, called the Input, Internal, Rule and Done nets. Each contains neurons (between 200 and 2600), and these neurons form CAs (between 1 and 13). Figure 1 represents the relationships between these networks. Arrows represent excitatory activation and forks represent inhibition.

The general flow of control of the simulation is:

1. The Input net is activated from the environment igniting CAs in it.

2. The Input net passes activation onto the parallel Internal net, igniting parallel Internal CAs; the Internal net is used to store later internal state.

3. The Internal CAs send activation along to the Rule net; a Rule CA ignites if the Internal net has the antecedent (if) CAs active.

4. The ignited rule CA sends activation to the consequent (then) Internal CA, suppresses the antecedent CAs, and sends activation to the Done network.

5. The Done network has one CA that suppresses the Input and Rule nets.

6. The consequent Internal CA also suppresses the Rule CA that ignited it.

7. The result of the interactions is that the consequent CA is left running, the Done CA is left running, and all other CAs are turned off.

There are 13 CAs in the Input network, 13 in the Internal network, 10 in the Rule network, and one in the Done network. The Input and Internal CAs correspond to the numbers 1 to 12 and the $+$ sign. The 10 rules correspond to $if 1 + 2 \rightarrow 3$ through $if 1 + 11 \rightarrow 12$.

For example, neurons in the 1, 2, and $+$ Input CAs are externally activated. This causes a cascade of activation within these CAs leading to many neurons firing in each step. After 10 steps, external activation is removed. If the Internal net were isolated, those CAs would continue to run indefinitely.

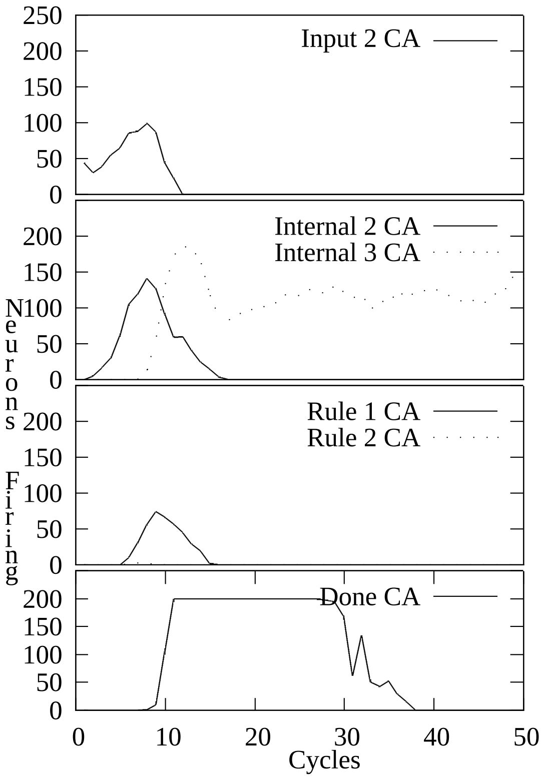

Figure 2: Neural Activity in different networks for $1 + 2 \rightarrow 3$

Activation is passed from the Input 1, 2, and + CAs to the 1, 2, and + Internal CAs causing them to ignite. In turn activation is passed from the ignited Internal CAs to the Rule net. Since the 1, and + CAs are active and they are antecedents of all of the rules, all of the rules receive some activation. Inhibitory connections between the Rule CAs causes the rules to compete. As the $if 1+2 \rightarrow 3$ rule is the only one with three antecedents active, it receives more activation than other rules and wins the competition and it ignites.

When the $if 1 + 2 \rightarrow 3$ CA ignites, it sends activation to the 3 Internal CA and also inhibits the Internal 1, 2, and + CAs. More or less simultaneously, the done CA ignites and the 3 Internal CA ignites. Next, the Done CA turns off the Input 1, 2, and + CAs, while the Internal 3 CA inhibits the Internal 2 CA. The combination of fatigue in its own neurons, loss of activation from the Input net, and suppression from the Internal 3 and Rule CA causes the 1, 2, and + Internal CAs to shut down. Finally, the inhibition from the Internal 3 and Done CAs causes the $if 1 + 2 \rightarrow 3$ Rule CA to shut down.

Figure 2 shows this process. It shows the number of neurons firing per cycle for the described CAs. Note that some neurons in the second rule fire, but the CA is suppressed by the first rule when it ignites.

Counting using Rules with Variable Binding

The simple rules are used as the basis of a more complex system that counts from one variable to another. For example, the system may count from 3 to 6, and the same system can be reused to count again from 4 to 9.

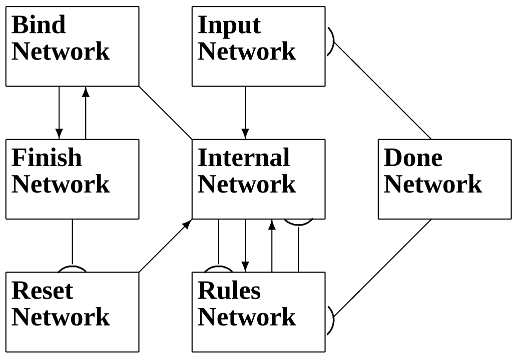

Figure 3: Topology of the Counting System

The simple rule topology is augmented with three new nets, Bind, Finish, and Reset. This topology is shown in Figure 3. The line between the Bind and Internal networks represents learned connections in both directions. The simple rule architecture from Figure 1 forms the core of this topology. As with Figure 1, arrows represent excitation and forks inhibition.

Initially, the final value is remembered by the system by presenting it to the Input network, and stimulating the Finish CA. Later this value is erased by spontaneous activation.

When counting starts, the initial value is presented to the Input Network and the Reset CA is stimulated. Both Reset and Finish nets have one CA each. The Reset CA sends activation to the Internal 1 and + CAs causing them to ignite. The ignited Internal CAs activate rules, which turn off the Internal 1 and + CAs, and ignite the subsequent CA. The reset net is still on, so the Internal 1 and + CAs come on again and the process is repeated until the final Internal CA is ignited.

The initial presentation of the Finish CA causes the Bind CA to ignite. The Bind CA has connections to and from the Finish CA and the Internal network. The only part of the system that learns is the Bind network and the connections to and from the Internal and Finish networks. As the Bind CA is now active it learns becoming associated with the final Internal CA. This learning is unsupervised and is based on neural co-firing. So, when the final Internal CA comes on, the Bind CA and thus the Finish CA comes on.

The Finish CA has strong inhibitory connections to the Reset CA in addition to connections to the Bind CA. So when the Finish CA ignites, it shuts down the Reset CA, and the process stops.

After this process, the binding is partially erased to enable the system to count again. This is done by spontaneous activation in the Internal and Bind Networks. The Bind network has two components with an equal number of neurons. The Bind CA and another component that does not actually reverberate and is thus not a CA. This extra component acts as a synaptic strength sink.

Our simulations use a form of Hebbian learning called compensatory learning [Huyck, 2004]. As with all Hebbian learning, excitatory synapses are strengthened when the neurons they connect fire simultaneously. An anti-Hebbian learning rule is also used so that the excitatory synapses are weakened if only the presynaptic neuron fires without the postsy-

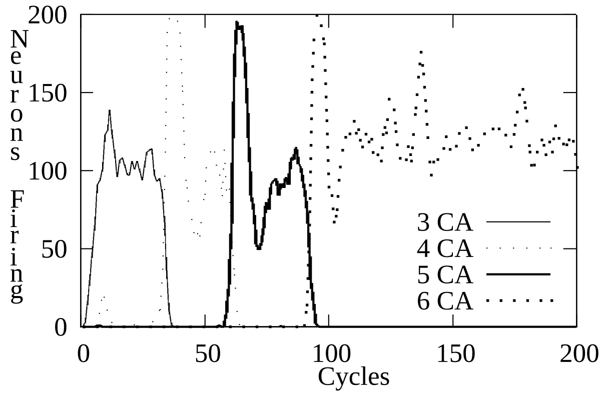

Figure 4: Selected Activity in the Count System Counting from 3 to 6

naptic neuron firing. The compensatory learning rule forces the total synaptic strength of a neuron toward a predefined constant value. There are parallel rules for inhibitory neurons so that co-firing inhibitory synapses are reduced toward zero becoming less inhibitory.

During spontaneous activation, a neuron in the bound Internal CA may fire while a neuron in the Bind CA it is connected to does not. This weakens that connection. Additionally, the bound Internal neuron may be connected to a neuron in the other part of the Bind network. If this neuron also fires, it will remove synaptic strength that may be used for a new binding. Spontaneous neural firing is a property of biological neurons [Bevan and Wilson, 1999, Abeles et al., 1993]. Related work with fatiguing LIF neurons indicates that spontaneous neural activation has useful properties for CA systems [Huyck and Bowles, 2004] and can be used for variable binding [Huyck, 2005].

For example the system counts from 3 to 6. Initially, Finish and the Input 6 CA are externally stimulated causing the input 6, internal 6, Finish and Bind CAs to ignite. As Bind and Internal 6 CA neurons are frequently co-active, the synaptic strength between them is increased.

After binding, the system starts by activating the Internal 3, and the Reset CA. Reset activates the Internal 1 and $+$ CAs igniting them. This in turn activates the $if 1 + 3 \rightarrow 4$ Rule CA, and as in the prior section the Internal 4 CA comes on. Again as in the prior section, Done comes on, the Input layer is shut down, the Rule CA is shut down and the Internal 1, 3, and $+$ CAs are shutdown.

The Reset CA again ignites the Internal $+$ and 1 CAs leading to the rule igniting followed by the Internal 5 CA igniting. This process repeats itself until the Internal 6 CA comes on. As this is bound to the Bind CA and the Bind CA activates the Finish CA, they both come on. The Finish CA suppresses the Reset CA turning it off. The process then ceases.

Figure 4 shows an example of neural activity. It shows the neurons firing per cycle in the 3 to 6 Internal CAs during counting from 3 to 6. Activity in the 1 and $+$ CAs are not shown.

The binding is then erased by spontaneous activation. The system can then be reused for further counting for example from 4 to 9. This process usually succeeds, but interestingly, it fails occasionally by stopping at 6. This is sensible because the system stops an earlier binding that should have been erased. This shows a typical error subjects manifest during various symbolic tasks.

Details of the Neural and Network Model

The basis of the model is fatiguing LIF neurons. Neurons collect activation from other neurons via synaptic connections. If the neuron does not fire some of that activation leaks away. Equation 1 describes the activation of a neuron i at time t if it does not fire at time $t - 1$.

$$A_{i_t} = \frac{A_{i_{t-1}}}{d} + \sum_{j \in V_i} w_{ji}, 0 < d < 1 \qquad (1)$$

The amount of leak is d. V_i is the set of all neurons that have connections to i and fire at time $t-1$. The weight, or synaptic strength, of the connection from neuron j to neuron i is w_{ji}.

Neurons also fatigue so that the more steps they fire the more difficult it becomes for them to fire. This is modelled by increasing the activation threshold θ if a neuron fires as described by Equation 2.

$$\theta_t = \theta_{t-1} + F_c \qquad (2)$$

In Equation 2 the threshold θ at time *t* is set to the threshold at time *t-1* + the fatigue constant F_c. If the neuron does not fire, the threshold is reduced toward the base resting level as in Equation 3.

$$\theta_t = \theta_{t-1} - F_r \qquad (3)$$

The threshold is reduced by the fatigue recovery constant F_r though it never becomes less than θ. So a neuron fires if it has more activity than the threshold plus accumulated fatigue. If it fires, it loses all activity.

In this system, neurons may be inhibitory or excitatory, but they obey Dale's principle [Eccles, 1986] so that a neuron cannot have both inhibitory and excitatory synapses leading from it. The ratio is usually 80/20 excitatory/inhibitory as in the mammalian cortex [Braitenberg, 1989].

The fatiguing LIF parameters are described in Table 1. The first major difference between nets is that the Done network is largely inhibitory because it is used to suppress other networks. The Bind network has a larger threshold and a slightly larger leak factor to make it more tolerant to input noise.

Network	θ	d	F_c	F_r	Inhibition
Input	4	1.5	1.0	2.0	20%
Internal	4	1.5	1.0	2.0	20%
Rules	4	1.5	1.0	2.0	20%
Done	4	1.5	1.0	2.0	80%
Finish	4	1.5	2.0	2.0	20%
Bind	6	2.0	2.0	2.0	20%
Reset	4	1.5	2.0	2.0	20%

Table 1: Fatiguing LIF Parameters by Network

Synaptic weights are modified by a compensatory Hebbian learning rule described by Equations 4 and 5 (see [Huyck, 2004]). Equation 4 is applied when neurons i and j fire in the same cycle. Equation 5 is an anti-Hebbian rule applied when the presynaptic neuron i fires and the postsynaptic neuron j does not.

$$\Delta^+ w_{ij} = (1 - w_{ij}) * R * 5^{(W_B - W_i)} \qquad (4)$$

$$\Delta^- w_{ij} = (w_{ij}) * -R * 5^{(W_i - W_B)} \qquad (5)$$

The learning rate is a constant R (0.1 in these simulations), and the first two terms of both rules are the standard correlatory learning rules. The compensatory modifier is the last term. W_B is a constant which represents the average total synaptic strength of the pre-synaptic neuron, and W_i is the current total synaptic strength. This modifier forces the total synaptic weight of a neuron toward W_B.

Synaptic weights for most nets and connections between nets are calculated before hand. The only case where they are learned is within the Binding network, between the Binding and Finish networks, and between the Binding and Internal networks. The target synaptic weight, W_B, for the Bind network is 30, for the Finish network is 35, and for the Internal network is 15.

	CAs	Synapses	Inter-CA	Intra-CA
Input	13	150	1.0†/-0.01	0.01/-0.12
Internal	13	150	1.0†/-0.01	0.01/-0.12
Rules	10	150	1.2†/-0.01	0.01/-4.0
Done	1	150	1.0†/-0.01	none
Finish	1	30	1.0†/-0.01	none
Bind	1‡	50	learned	learned
Reset	1	30	1.0†/-0.01	none

Table 2: Topology within Nets

The topology within nets can be described by Table 2. All CAs consist of 200 neurons and they are orthogonal, no neuron is in two CAs. The Input net has 13 CAs and thus 2600 neurons. Each neuron has 150 synapses to other neurons in the net. These neurons are connected randomly, though a neuron cannot connect to itself. The weights are predetermined, so a neuron in a CA has a connection to another neuron in the same CA with weight 1 if it is an excitatory neuron or -0.01 if it is inhibitory. Connections to other neurons outside the CA have a 0.01 or -0.12 weight. There are two caveats within this table. The Inter-CA connections marked by †have an weight of this number. The 1.0 number (e.g. Input) is calculated by 1.5 - a random number between 0 and 1, and the 1.2 number in Rules is 1.7 minus a random number between 0 and 1. The second caveat, as described in the section on counting and denoted by ‡, is that the Bind net has one CA, but also has an additional 200 neurons that never form a CA; these extra neurons act as activation sinks during unbinding.

Table 3 describes the number of synapses per neuron from one net to another. The presynaptic net is in the column, and the postsynaptic net the row. So, each Input neuron has 50 connections to the Internal network. All of these are randomly assigned. The connections within a net are marked by * and are described in table 2.

The weights of these connections are set in all cases except between Internal and Finish and Bind. Input to Internal weights are 2.0 minus a random number if they are parallel, and 0.1 if not; inhibitory connections are -0.1. Internal to Rules have 0.36/-0.01 and 0.01/3.6 weights. Rules to Internal stimulating weights are 2.8/0.01, suppressing weights are 0.01/-4.0, and neutral weights are 0.01/-0.01. Rules to Done are 0.4/-0.1, Done to Input are 0.01/-1.0, and Done to Rules are 0.01/-0.5. Finish to Reset are 0.01/-1.0, and Finish to Rules are 0.01/-4.0. Reset to Internal weights are 0.5/-0.1

	Input	Int.	Rules	Done	F	B	R
Input	*	50					
Internal		*	20			10	
Rules		60	*	10			
Done	100		30	*			
Finish			50		*	15	50
Bind		15			15	*	
Reset		50					*

Table 3: Topology between Nets

to the Internal + and 1 CAs, 0.01/-0.01 otherwise.

Initially some training is needed to put the Finish, Bind, and Internal learnable weights to a good position. This is done by an initial 400 cycles of spontaneous activation to the bind net. This is followed by alternating presentations of Finish and Bind instances with instances of the non-Bind neurons. In each case 50 neurons are selected at random from the patterns and receive external activation. They are presented for 10 cycles, and then the system is allowed to run for 40 more cycles. The fatigue and activity are then reset. This continues for a total of 1600 cycles. So, by the 2000th cycle Finish and Bind are CAs that are to some extent connected.

Binding occurs by presenting 50 neurons of the Input CA to be bound, and 50 of the Finish CA's are externally stimulated for 10 cycles. This is then allowed to run for a further 190 cycles. By this time, the Internal CA is bound to the Bind CA, and thus to the Finish CA.

Unbinding is achieved by 1200 cycles of spontaneous activation of the Bind and Internal networks. During spontaneous activation each neuron has a one percent chance of firing.

The system is by no means perfect, but it works in principle. We ran the system with 50 different nets, each time counting from 3 to 6, then from 4 to 9 to show that binding can be erased and reused. 74% of the time it correctly counted from 3 to 6. On the second test, 50% of the time it counted from 4 to 9; 28% of the time it stopped prematurely at 6 showing that the binding was not properly erased; 22% of the time it ended somewhere else.

Conclusion and Discussion

Clearly this is a very simple cognitive model of counting. Also the model is not matched to psychological data, and timing is described only by cycles.

What is interesting is that a neural model does the symbolic counting task at all. Moreover, from a cognitive modelling perspective, it exhibits some similarities to human symbolic processing: it fails sensibly when counting stops at a prior binding that should have been erased.

Another interesting result is this shows a neural reason for classifying rules. Rules using only constants are different from those using variables. Simple rules are based on constants and are of the form $if c_1 c_2 \ldots c_n \rightarrow c_m c_{m+1} \ldots c_p$ and these only require the proper sort of excitatory and inhibitory links. The counting rule is more complex and could be described by $while X < F \rightarrow X = X + 1$. The $X = X + 1$ portion is handled by the simple rules, but the F portion requires binding.

Of course there is a long way to go to have a complete neurally based cognitive model of this phenomena and even further to have a solid cognitive model. A relatively simple expansion of the system could enable it to learn new simple rules like $if 3+2 \rightarrow 5$ This would require a process of adding beyond one, but would then allow the system to cache results it had processed earlier. This would be similar to chunking [Laird et al., 1987]

There are more complex modifications that could improve the model. It could be tied to psychological data; timing could be modelled by attaching a time, say 10 ms, to a cycle; system parameters could be modified to correspond to failure rates in counting. More rapid binding could be done by using short-term synaptic changes instead of long-term changes. Spontaneous activation could be used all the time instead of just during binding activities. A sensory system could be devised and the system could actually learn numbers by learning to count items in the environment. This would provide a much sounder grounding to the numerical concepts.

While these modifications seem tenable, we have no idea how the connections between nets (aside for Input to Internal) could be learned. There could be another topology that did the same task, but again we have no sound idea how this topology could be learned. The simple rules described above encode a sequence. There has been some work on learning sequences with neurons, and this provides some guidance for learning other processes. This work highlights the need for a better understanding of the mechanisms required to learn symbolic processing tasks with neural systems.

A neural system develops, in the long-term, by a change in synaptic strength. This self-organisation should enable the system to learn tasks and not have them simply programmed. There have been advances in understanding how a neural system can learn to categorise, but we are not aware of a sound theory that specifies the boundaries of what is learned and when. Perhaps the development of such a theory will shed light on learning more complex processes such as counting.

References

[Abeles et al., 1993] Abeles, M., Bergman, H., Margalit, E., and Vaadia, E. (1993). Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *Journal of Neurophysiology*, 70:4:1629–38.

[Amit, 1989] Amit, D. (1989). *Modelling Brain Function: The world of attractor neural networks*. Cambridge University Press.

[Anderson and Lebiere, 1998] Anderson, J. and Lebiere, C. (1998). *The Atomic Components of Thought*. Lawrence Erlbaum.

[Bevan and Wilson, 1999] Bevan, M. and Wilson, C. (1999). Mechanisms underlying spontaneous oscillation and rhythmic firing in rat subthalamic neurons. *Journal of Neuroscience*, pages 7617–7628.

[Braitenberg, 1989] Braitenberg, V. (1989). Some arguments for a theory of cell assemblies in the cerebral cortex. In Nadel, Cooper, C. and Harnish, editors, *Neural Connections, Mental Computation*. MIT Press.

[Dalenoort, 1985] Dalenoort, G. J. (1985). The representation of tasks in active cognitive networks. *Journal of Cognitive Ssytems*, 1:3:253–272.

[Eccles, 1986] Eccles, J. (1986). Chemical transmission and dale's principle. *Prog. Brain Research*, 86:3–13.

[Fodor, 2000] Fodor, J. (2000). *The Mind Doesn't Work That Way: the Scope and Limits of Computational Psychology*. MIT Press.

[Hebb, 1949] Hebb, D. O. (1949). *The Organization of Behavior*. J. Wiley & Sons.

[Hopfield, 1982] Hopfield, J. (1982). Neural nets and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences*, 79.

[Huyck, 2004] Huyck, C. (2004). Overlapping cell assemblies from correlators. *Neurocomputing*, 56:435–9.

[Huyck, 2005] Huyck, C. (2005). Variable binding of cell assemblies with binding areas and spontaneous neural activation. In *Proceedings of the 22nd Workshop of the European Society for the Study of Cognitive Systems*.

[Huyck and Bowles, 2004] Huyck, C. and Bowles, R. (2004). Spontaneous neural firing in biological and artificial neural systems. *Jour. of Cognitive Systems*, 6:1:31–40.

[Kaplan et al., 1991] Kaplan, S., Sonntag, M., and Chown, E. (1991). Tracing recurrent activity in cognitive elements(trace): A model of temporal dynamics in a cell assembly. *Connection Science*, 3:179–206.

[Kieras et al., 1997] Kieras, D., Wood, S., and Meyer, D. (1997). Predictive engineering models based on the epic architecture for a multimodal high-performance human-computer interaction task. *ACM Transactions on Computer-Human Interaction*, 4:3:230–275.

[Knoblauch and Palm, 2001] Knoblauch, A. and Palm, G. (2001). Pattern separation and synchronization in spiking associative memories and visual areas. *Neural Networks*, 14:763–780.

[Laird et al., 1987] Laird, J., Newell, A., and Rosenbloom, P. (1987). Soar: An architecture for general cognition. *Artificial Intelligence*, 33:1.

[Palm, 1990] Palm, G. (1990). Cell assemblies as a guideline for brain research. *Concepts in Neuroscience*, 1:1:133–47.

[Sakurai, 1998] Sakurai, Y. (1998). The search for cell assemblies in the working brain. *Behavioral Brain Research*, 91:1–13.

[Tal and Schwartz, 1997] Tal, D. and Schwartz, E. (1997). Computing with the leaky integrate-and-fire neuron: Logarithmic computation and multiplication. *Neural Computation*, 9:2:305–318.

[Wiles and Elman, 1995] Wiles, J. and Elman, J. (1995). Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pages 482–487.