

To appear in *Connection Science*
Vol. 00, No. 00, Month 20XX, 1–21

A Spiking Half-Cognitive Model for Classification

Christian R. Huyck^{a*} and Ritwik Kulkarni^a

^a*Middlesex University, The Burroughs, Hendon, London, UK*

(Received 00 Month 20XX; accepted 00 Month 20XX)

This paper describes a spiking neural network that learns classes. Following a classic Psychological task, the model learns some types of classes better than other types, so the net is a spiking cognitive model of classification. A simulated neural system, derived from an existing model, learns natural kinds, but is unable to form sufficient attractor states for all of the types of classes. An extension of the model, using a combination of singleton and triplets of input features, learns all of the types. The models make use of a principled mechanism for spontaneous firing, and a compensatory Hebbian learning rule. Combined, the mechanisms allow learning to spread to neurons not directly stimulated by the environment. The overall network learns the types of classes in a fashion broadly consistent with the Psychological data. However, the order of speed of learning the types is not entirely consistent with the Psychological data, but may be consistent with one of two Psychological systems a given person possesses. A Psychological test of this hypothesis is proposed.

Keywords: classification, cognitive model, spiking neurons, Hebbian learning

1. Introduction

Classification of input stimuli posits itself as the first building block of higher order cognitive functions. It has been studied intensely for several decades in terms of computational learning since the advent of Perceptrons in 1957 and before that for several centuries as a philosophical discussion since Plato's ideas of clustering 'similar' objects. This paper proposes a cognitive model of classification based on simulated spiking neurons that learn via a Hebbian learning mechanism. Our aim, rather than to build a perfect classifier, is to build one that reproduces error in a similar manner to humans when they classify.

A good cognitive model behaves like a person. Better ones behave like people on a range of tasks. This paper accounts for some of the data of human subjects from the study described in Shepard, Hovland and Jenkins' 1961 paper Learning and Memorization of Classifications ([Shepard et al., 1961] and see section 2.1). It explores the performance of human subjects over a forced binary classification task involving self discovered rules¹. Subjects were presented with images that had three salient dimensions that took binary attributes thus leading to 2^3 possible instances that were then divided into two classes of four instances each. The logic of dividing the eight instances into two classes is what forms the base classification complexity. With three binary dimensions (see Figure 1), there exist six different types of classification scenarios such that within each of the six scenarios, any different arrangement of the binary classes can be represented as a simple rotation over one of the three dimensions. However none of the six types can be

*Corresponding author. Email: c.huyck@mdx.ac.uk

¹Shepard, Hovland and Jenkins refer to classes, though modern terminology often refers to categories. This paper refers to classes throughout.

represented as a rotation of other types of scenarios. Subjects were shown the stimuli images in a self-paced manner and were informed if their choice of the binary class was correct or not. Thus subjects had to discover the underlying logic rule behind the classification, or, in other words, implicitly discover which of the six scenarios is in use and then map it to the binary classes. Subjects did better on some types than on other types. It was found that the performance of subjects was type-dependent, such that the ease of classification loosely predicted the number of logic rules necessary for classification. However this relationship was lost with time, over successive trials, indicating that once the rule had been discovered the complexity of classification had a less significant impact on performance.

These six types of classification problems can be mapped to other categories of classification problem. Two of the types are linearly separable (see section 3.1). When inputs are linearly separable, the problem becomes a trivial matter of finding the decision boundary for a Perceptron [Minsky and Papert, 1969], whereas for nonlinearly separable inputs, adding hidden layers and using Backpropagation can solve many classification problems. Even more powerful are the statistical approaches that use probabilistic measures such as Bayesian networks [Friedman et al., 1997, Bielza et al., 2011], which have found solid ground in the domain of artificial intelligence, and machine learning approaches [Yarkoni et al., 2011] that can detect relations in large datasets. However when exploring the mechanisms of cognition, there is a need to understand the underlying neural processes that can bring about the rich classification behaviour seen in human and other animal studies. For example, Multi-Layer Perceptrons (MLPs) are used to classify this data (see section 3.3); however it is not at all clear how this mechanism relates to a cognitive model. Statistical approaches do give an insight about what is going on during complex classification processes; for example the hierarchical models of object recognition [Riesenhuber and Poggio, 1999] provide an account of a biologically plausible way of tackling invariant object recognition. Yet the question of grounding these models in biologically relevant neural network models remains an open domain of exploration.

In the context of this task, a good cognitive model will learn all six types of classes. Like humans, learning will vary by type, but not within type. The model will both learn the simpler types faster, and make fewer errors on the simpler types. None the less, eventually, the system based on the model, like humans, will be able to perform almost perfectly on even the most complex type.

Since the brain is made of neurons, a neuro-cognitive model is closer to the underlying computation than other computational models, for example, rules. As topology is important, a recurrently connected model is more biologically realistic, as is a sparsely connected model. Spiking models are even closer approximations than continuously valued neurons, and are particularly important because biological learning is based on spikes. The models described below have only a few thousand neurons, and are thus far from the billions of neurons in a human brain, and those neurons are very simple models compared to actual biological neurons. None the less, the models are constrained by many of the parallel biological factors, and thus inform fuller understanding of the underlying neural activity from which cognition emerges, and that cognition.

While the machine learning community makes use of linear separability as a property of a classification task, the Cognitive Science community has explored natural kinds [Rosch and Mervis, 1975]; these are the types of classes that humans and other animals typically form, and so are of particular interest to cognitive modelling. Shepard et al.'s six types of classification problems can also be mapped to natural kinds (see section 2.2).

Spiking neural networks provide a biologically relevant platform where several important details of neural processes are maintained while abstracting over

minute details of cellular mechanisms; they are computationally viable and, to some extent, analytically understandable for the simulation of cognitive behaviour [Ghosh-Dastidar and Adeli, 2009]. Spiking neural networks with a specific network architecture have been shown to be effective and efficient for the task of unsupervised clustering of complex “real life” data [Bohte et al., 2002]. This article attempts to go one step further by not only restricting computation to biologically plausible mechanisms of classification but also by adding an extra layer of cognitive processing over self-organised clustering to try and replicate a Psychological study on classification.

This paper explores to what extent a classifying model, based on spiking Fatiguing Leaky Integrate and Fire (FLIF) neurons [Huyck and Parvizi, 2012] with a modular recurrent architecture, is successful in replicating Shepard et al.’s study. The model is a modified version of a similar model [Huyck and Mitchell, 2014] used for several machine learning benchmark classification tasks.

In these simulations, attractor states are learned. The input converges on particular states linking the input to the class. The original mechanism works for the linearly separable classes. However, a combination of the input features is needed for the other classes.

The remainder of the paper starts with a literature review of the classification tasks (section 2.1), natural kinds (section 2.2), and the basics of the neural and plasticity models used in the later simulations (sections 2.3.1 and 2.3.2). Next simulations derived directly from the earlier system (section 3.1), with modified input (section 3.2), and using MLPs (section 3.3) are described, and the results discussed. Section 4 describes the final system that is used as the cognitive model. Section 5 discusses the model and concludes that the model is at best a half-cognitive model, aligning with only one system of a commonly used two system model of the mind [Kahneman, 2011]; it also proposes a testable prediction. Section 6 concludes.

2. Background

The research literature on learning, even on learning classifications, is vast (see [Kotsiantis et al., 2007, Ashby and O’Brien, 2005] for review). Below, the particular study of Shepard, Hovland and Jenkins is described. This is followed by a brief description of work on natural kinds to explore the type of classes humans typically learn. The section then switches to simulated neural systems, as this paper describes a simulated neural model of classification.

2.1. *Learning and Memorization of Classifications*

The classic paper [Shepard et al., 1961] selects a small subset of the possible classification problems. It works only with two class problems, and only with problems with four items in both classes. Shepard, Hovland and Jenkins work with different triples of binary variables, but one set they used frequently is the triple *Squares* vs. *Triangles*, *Black* vs. *White*, and *Large* vs. *Small*. This yields eight items. These items can be laid out on a cube with items differing by only one feature sharing an edge (see Figure 1). One triple is a large black triangle.

A particular classification task involves four items of the eight points of the cube as one class, with the remaining four as the other class. For instance, one class might be the *Black* items, and the other would be the *White* items.

Shepard, Hovland and Jenkins had subjects learn these classification tasks. Subjects

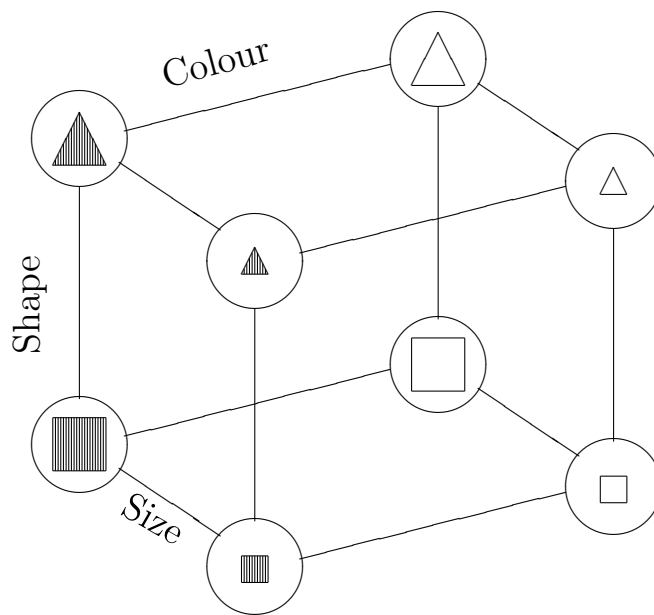


Figure 1. Cube arrangement of items used in this paper and Learning and Memorization of Classifications.

were told the two possible responses before the first item was presented. The item was then presented, and after they answered, they were told if they were correct or not.

While there are $\binom{8}{4} = 70$ possible classes of this type, these can be grouped into six types. Table 1 shows the six types. The classes are described by the solid circles vs. the empty circles on the cube arrangement of items. Rotating the cube will give other classifications, but these are in a real sense identical. For example, the classification based on the *Black* items is identical to the classification based on the *Square* items. They are both of Type 1. They are equally complex, and if the features are considered equally salient, they should be and are equally easy to learn. Table 2 shows the exact same types with example pictures.

<p>Type 1</p>	<p>Type 2</p>	<p>Type 3</p>
<p>Type 4</p>	<p>Type 5</p>	<p>Type 6</p>

Table 1. The six different types of class, using the Cube description from Figure 1. Solid circles show one class and empty circles the other.

Classifications of different types cannot be transposed. For example, one Type 4 classification has the first class composed of the *Black* elements except for the small black

square, and with the large white triangle; in some sense, the large black triangle is central to this class as the other three elements all share two of its features. One Type 3 classification has the *Black* elements except for the small black square, and with the small white triangle; it has no central element. Section 3.1 has a further analysis of the different types.

















































Type 1	Type 2	Type 3
 	 	 
 	 	 
 	 	 
 	 	 
Type 4	Type 5	Type 6
 	 	 
 	 	 
 	 	 
 	 	 

Table 2. The six different types of class, using examples. These echo the cube description from Table 1.

This is, of course, what Shepard, Hovland and Jenkins found. Within a classification type, subjects performed very similarly. Performance includes both accuracy and time. However, between types, performance differed. Subjects did best on Type 1 and worst on Type 6. Performance on Type 2 was slightly worse than Type 1, and Types 3, 4 and 5 were lumped together with performance between Types 2 and 6. Type 1 > 2 > 3&4&5 > 6.

For example, when trying a Type 1 classification, subjects on average mis-classified 12 on the first trial and five on the fifth. For Type 6, they mis-classified 65 on the first trial and 15 on the fifth. A cognitive model should reproduce these errors.

The differences between types effectively becomes insignificant with successive trials. It does take longer for some types of classes to be learned than others, but eventually, subjects discover how to classify each task. The type of stimulus has an effect only as a novelty factor and the subjects are soon able to correctly classify stimuli.

2.2. Natural Kinds

Classes that are typically used by humans and other animals are not based on mathematically precise rules. Like the word 'games', most concepts are not defined by necessary

and sufficient conditions [Wittgenstein, 1953]. Instead most concepts are based around family resemblances [Rosch and Mervis, 1975]. Members of a class share many features, though none may have all of the associated features. These natural kinds are the type of classes that animals typically develop.

A member of a family shares many features with many other members of the family. Natural kinds typically refer to objects in the real world, for instance cats, or chairs.

All classes are not natural kinds. In a two class problem based on eight items composed of three binary features (like the problems in section 2.1) in some classes many members do not share many features with other members of the family. With the types from section 2.1, Type 4 is what is most typically referred to as a natural kind. All members share at least two features of the three features. Type 1 class membership is determined by a necessary and sufficient feature; this is also a natural kind but a very simple form.

Features are not shared preferentially within a class for Type 2 and Type 6 problems, so they are not natural kinds. Features are shared preferentially in Type 5, but one element of the class shares no features with the others, so the class is not a natural kind. Type 3 is not a particularly sound natural kind as members do not share many features. On this shared common feature analysis Type $1 > 4 > 3 > 2 \& 6 > 5$.

Of course, this does not mean that people are incapable of using classes based on rules or even arbitrary classes; they can also develop classes based on necessary and sufficient conditions; and they can develop classes based on simply enumerating the members of the class. It is clear that most if not all humans can develop these types of class.

2.3. *Simulated Neural Systems*

Neural simulations are becoming increasingly accurate. The number of neurons used is increasing, and the biological faithfulness of the neural model is improving. However, the current state of the art is incapable of modelling a brain at a neural level with any reasonable degree of fidelity. Consequently, simulations must necessarily be an approximation of the underlying neural process.

Existing neural and connectionist models are solving more and more problems. The state of the art in machine learning is deep nets [Schmidhuber, 2015]. These can be used to learn very complex classification tasks, however they do not necessarily align with biology, nor provide a good cognitive model. More biologically accurate simulations have been used for classifications (e.g. [Wade et al., 2010].) There are however few spiking cognitive models (e.g. [Zipser et al., 1993, Mongillo et al., 2008, Huyck, 2009, Stewart and Eliasmith, 2011]), though cognitive neuroscientists are attempting to unravel the neural basis of cognitive phenomena (e.g. [D’Espisito, 2007]). The authors are not aware of any other spiking cognitive models of classification.

For the purposes of this paper, what is needed is a good cognitive model of classification. While other models, (e.g. symbolic models) may account for data, a relatively faithful neural model may both address the key aspects of classification, and support the development of more complex simulated neural systems. This paper develops a more accurate cognitive model based on a reasonably biologically faithful neural system.

2.3.1. *Neural Models*

Perhaps the key component of a neural simulation is the neural model itself (see [Brette et al., 2007] for a good review). There are many popular models, with compartmental models (e.g. [Hodgkin and Huxley, 1952]) being the most biologically faithful.

Spiking point models are widely used, being effective but computationally simpler than

compartmental models. The integrate and fire model [Lapicque, 1907] is the simplest and earliest version. These models integrate activity from incoming synapses; if they surpass a threshold, they spike. The spike propagates from the firing neuron to all post-synaptic neurons propagating activity and continuing the cycle.

Modern point models typically include leak (e.g. [Stein, 1967, Brette and Gerstner, 2005]) to form leaky integrate and fire (LIF) models. If a neuron does not fire, it retains some of its activity, but some leaks away. The simulations described below model leak with Equation 1. A_j^t is the total activity neuron j has at time t . D is the leak or decay rate, which is greater than 1. If the neuron does not fire at time $t - 1$, ignoring incoming activity, some of the activity leaks away. When a neuron fires, its activity is reset to 0.

$$A_j^t = A_j^{t-1}/D \quad (1)$$

The LIF model can be extended to include fatigue. When a neuron fires, it fatigues and becomes more difficult to fire. In the simulation below, this and firing are modelled by Equation 2. The summation is the incoming activity where V_i is the set of pre-synaptic neurons that fire, and w_{ij} is the synaptic weight from neuron i to neuron j . θ is the firing threshold and F_j is the current fatigue of neuron j . As fatigue increases, it becomes more difficult for the neuron to fire.

$$\theta + F_j < A_j = \sum_{i \in V_i} w_{ij} \quad (2)$$

When the neuron fires, fatigue increases by a constant F_c , unless it has a total fatigue less than -0.25 . If this is the case, fatigue is halved. If the neuron does not fire, fatigue decreases by another constant F_r .

This FLIF model aligns relatively well with biological data [Huyck and Parvizi, 2012]. The model is discrete updating every 10 ms. With the four parameters set (threshold θ is 2.2; decay D is 1.12; fatigue increase F_c is 0.45; and fatigue recovery F_r is 0.01) over 90% of simulated spikes map to biological spikes (rat somatosensory neurons).

An additional benefit is that the neurons fire spontaneously when they are not externally stimulated. If a neuron has not fired for some time, it will become hypo-fatigued, and spike. To prevent frequent spikes after hypo-fatigue, the firing rule halves fatigue when it is negative. This principled mechanism for spontaneous firing allows synaptic strength to move to neurons that have not been activated by the environment.

2.3.2. Plasticity

Another major consideration in neural simulation is plasticity. How do the synaptic weights change? The general scientific opinion is that plasticity is Hebbian, though there remains interest in non-Hebbian plasticity (e.g. [Senn et al., 2002]). Hebbian plasticity is local involving only the pre and post-synaptic neurons [Hebb, 1949]. The weights change so that if the pre-synaptic neuron tends to cause the post-synaptic neuron to fire, the weights increase. Less explicit is that the weights decrease if the pre-synaptic neuron does not tend to cause the post-synaptic neuron to fire.

This description is very general and has led to many plasticity rules. Spike timing dependent plasticity has received a great deal of attention [Bi and Poo, 1998,

Song et al., 2000]. In this sort of Hebbian rule, the precise timing of the pre and post-synaptic neurons are very important. In many example rules, the difference is considered at the two firing simultaneously. If the pre-synaptic neuron fires after the post-synaptic neuron, the synaptic weight is reduced; if the pre-synaptic neuron fires first, the weight increases. There is some biological evidence to support this type of rule, and as it involves only two neurons, the system can be studied both in-vitro and in-vivo.

An older and simpler rule is known as Oja's rule [Oja, 1982]. This rule forces the synaptic weight towards the likelihood that the post-synaptic neuron co-fires along with the pre-synaptic neuron. So, if half the time the pre-synaptic neuron fires, the post-synaptic neuron fires, the weight is .5. This of course can be weighted by a scaling constant. This is the basis of the compensatory learning rule used in the simulations below, shown by Equations 3 and 4. Oja's rule is reflected by the component of the Equations without the exponent. w_{ij} refers to the weight of the synapse normalised between 0 and 1. R is the learning rate, and C the weighting factor (the current synaptic weight). The weight increases when the neurons co-fire (Equation 3) and decreases when the pre-synaptic neuron fires and the post-synaptic neuron does not (Equation 4).

$$\Delta_+ w_{ij} = R * C[(1 - w_{ij}) * 10^{(W_B - W_k)}] \quad (3)$$

$$\Delta_- w_{ij} = -R * C[w_{ij} * 10^{(W_k - W_B)}] \quad (4)$$

The exponential component of the equations is the compensatory modifier. This forces the total synaptic weight a neuron has towards a constant. W_B is the target total synaptic weight for a neuron, and W_k is the total current synaptic weight. When the current weight is larger than the target weight, the increases is less, and the decrease is more. Similarly, when it is below the target weight it increases more and decreases less (see [Huyck, 2007] for a more complete explanation).

In this context, the synaptic strength of a neuron can mean the strength coming into a neuron, the strength leaving a neuron, or some combination of the two. When the strength is based on the synapses leaving the neuron, it is called post-compensatory learning, and when it is based on the strength entering a neuron it is called pre-compensatory learning. The simulations described in this paper do not make use of a blend of the post and pre-compensatory learning on a particular synapse.

When a neuron is not stimulated by the environment and has weak incoming synaptic connections, it still fires due to hypo-fatigue. If it is learning via a pre-compensatory learning rule, as it has weak incoming synaptic strength, it will tend to increase its weight and become involved in circuits over time.

3. Standard Simulated Neuron Learning System

This section describes two spiking neural models for classification. The first is based on an earlier classification model, and the model does not classify some of section 2.1's types well. The second model modifies the input, and, in the simulations run, classifies perfectly. Unfortunately, this perfect classification does not fit well as a cognitive model, since subjects make mistakes. A third section describes an MLP that categorises the data. This performs more or less perfectly, and is thus, also, a poor cognitive model.

3.1. Only Modification for Domain

An earlier simulated model has been used for many standard classification tasks [Huyck and Mitchell, 2014]. The neural model was a FLIF model (see 2.3.1), and there was a four subnet topology (see Figure 2). The subnet topology was based on exterior and interior subnets. The exterior *Input* and *Output* subnets are a relatively simple way of handling a range of input features and numbers of classes. The separation of the interior *SOM* (self organising map) and *Hidden* subnets allows the classes to develop more slowly with the *SOM* subnet accounting for some of the feature combination, while the *Hidden* subnet built up attractor space for the classes. A comparison of the learned weights shows that synapses between neurons in the *Hidden* subnet has large type dependent differences, while other connections do not (see section 4.2).

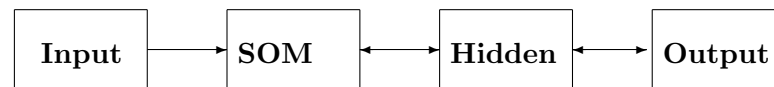


Figure 2. Gross topology of classification simulations. Boxes represent subnets, and arrows excitatory connections between subnets. There are (not indicated) connections within the *SOM*, *Hidden*, and *Output* subnets.

The *Input* subnet consists of neurons that represent the input features of class examples. The *Output* subnet is divided into groups of neurons for each class. During training, instances of the training set are presented by stimulating the associated input neurons in the *Input* subnet, and the correct class neurons in the *Output* subnet. Plastic synapses in the system then learn. In this paper, one training (or testing) instance is called an epoch typically lasting 75 cycles, with external stimulation lasting for the first 40 cycles, and the system allowed to run on its own for the remaining 35. The full testing set is typically presented several times, with a typical duration of 24,000 cycles (320 epochs).

The system uses post and pre-compensatory learning and the type of plasticity is based on the pre-synaptic subnet. It is post-compensatory for the *Input* subnet, and pre-compensatory for the remaining subnets. In all the simulations below the target synaptic strength (W_B) is 5 for *Input*, 3 for *SOM*, 5 for *Hidden*, and 10 for *Output* neurons. This is consistent with the earlier classification work.

Testing is done by presenting the inputs as during training; this uses an identical epoch length. The method for choosing a class for a given input is to count the number of *Output* neurons firing in each class during the entire testing epoch. If the first class has more neurons firing, it is selected; otherwise, the second class is selected.

A one subnet inhibitory neuron was used for each of the *SOM*, *Hidden* and *Output* subnets. This had a synapse from each of the neurons in the subnet, and if the number of neurons firing exceeded a threshold, inhibition (not shown in Figure 2) was sent to all of neurons in the subnet; the inhibition was proportional to the number of neurons firing beyond the threshold.

This basic system was used for several classification tasks from the University of California at Irvine’s archive [Hettich and Bay, 1999]. These included the Yeast, Iris and Cars tasks. All performed near other standard classification algorithms, for example Self-Organizing Maps [Kohonen, 1997].

This standard topology was used as a starting point for binary classifications. The only modifications were to accommodate the new domain. Using this initial topology, the *Input* subnet needed six separate inputs, three pairs of feature values. Each feature-value is allocated 20 neurons; when an instance is presented, 50 of the possible 60 input neurons is externally stimulated. Similarly, the *Output* subnet is broken into two classes,

each with 50 neurons; 20 of these were externally stimulated. Training lengths were the typical 24,000 cycles.

These parameters are derived from the prior simulations on other classification tasks. The number of neurons externally activated should be less than the total possibly activated to increase the variance in the learning. There is no proof of optimality of this topology; indeed, the authors strongly suspect that it is not optimal, just a good point in the vast parameter space.

The network topology is as usual; there are 1000 neurons in the *SOM* subnet, and 1000 in the *Hidden* subnet. Each *Input* neuron connects randomly to 20 neurons in the *SOM* subnet, each neuron in the other subnets connects to 10 other neurons in its own subnet, each *SOM* neuron connects to 10 *Hidden* neurons, each *Hidden* neuron connects to 15 *SOM* neurons and 10 *Output* neurons, and each *Output* neuron connects to 10 *Hidden* neurons. The *Output* subnet has subnet inhibition starting at 50 neurons, and the *SOM* and *Hidden* subnets starting at 100. These are all values used for earlier classification experiments. All code is written in java and can be found at <http://www.cwa.mdx.ac.uk/NEAL/code/CANTShepard.tar.gz>

After training, learning was turned off, and the 8 items were each presented 5 times and classified. For each type, the process was repeated 50 times. Each new run uses a different random seed for topology generation, so each network is almost certainly unique.

Depending on the type of class, performance on the task ranges from perfect to chance (see Table 3). The columns differentiate by type. The two rows represent different classification tasks within a type. These simulations run true to Shepard, Hovland and Jenkins' types with different classes within the types performing almost identically. There is only one instance of Type 6.

T1	T2	T3	T4	T5	T6
100	52.3	75.6	98.3	73.6	51
100	51.2	76.4	93.6	73.6	-

Table 3. Performance on classification tasks by type. The first row shows one instance of a type, and the second a different instance of that type. For example, for T1, the first is *Black* vs. *White*, and the second *Square* vs. *Triangle*. There is only one instance of a T6 classification task.

As can be seen, Types 1 and 4 are almost perfectly classified. Types 3 and 5 perform at about 75%, and Types 2 and 6 perform just slightly above chance. This behaviour is driven by the direct relationship between the input features and the output classifications. In the case of Type 1, the primary input feature determines the output classification. In the first class, if the item is *Black* it is of class A, and if it is *White* it is of class B. In Type 4, three features appear in one item, and two of them appear in the other three. For example, the first row of Table 3 is the class *Black* without the small black square, and with the large white triangle. *Black*, *Square* and *Large* each appear in large black square, and each of the other class members has two of the features (e.g. large white square has *Large* and *Square*). The natural kind mechanism prevails, and the items are classified almost perfectly. There are two attractor states.

Types 3 and 5 also have skewed input relations. In both cases one feature appears in three of the four items, but does not appear in the fourth. In the first row, Type 3 is the *Squares* with the large black triangle, without the large white square. In this case the system also forms two attractor states, it just does not include the outlier in the state. Thus, during classification the system gets roughly a quarter of the items wrong. Similarly Type 5 in row 1 is the *Blacks* with the small white square, without the small black square. On both of these types, the classifier overgeneralizes. Do note, that Type

3 could also be described as the *Blacks* without the small black triangle and with the small white square. All class items share more features than all class items in a Type 5 class.

Types 1 and 4 are linearly separable. Types 2, 3, 5 and 6 are not.

Inspection of 10 runs of Type 5 classification showed that there were 106 classification errors. 99 of them were the outlier. For Type 3, 88 of the 98 errors were from the two outliers in each class.

The system performs quite poorly on Types 2 and 6. In this case, the input features do not directly align with the output classes. For instance, the Type 2 classification in row one is the black squares, and white triangles. In this case, each feature value appears in exactly two items in each class. Similarly, in Type 6 each feature appears in exactly two items in each class. In row one, class A is made up of the small black square, large white square, large black triangle, and small white triangle. The system finds it difficult to combine features, and gets roughly chance classification behaviour. This relates to the exclusive or problem [Minsky and Papert, 1969].

Classification is successful due to the learning of attractor states. When inputs are presented, neurons in the *SOM* and *Hidden* subnets fire causing the appropriate neurons in the *Output* subnet to fire. These are not Cell Assemblies [Hebb, 1949] because they do not fire persistently. These attractor states are Cell Assembly like because cell assemblies are attractor states.

In the case of Type 1 and Type 4, there are only two attractor states needed, one for each class. In Type 2 and Type 5, four attractor states are needed, two for each class. In Type 2 these all respond to two input features. In Type 5, for each class there is one state responding for three items that share features, and one state that responds to one item. For Type 6, 8 attractors are needed, one for each item. This analysis is most complex for Type 3 because the outlier items in a class share no features.

3.2. Three Feature Input Model

A minor adjustment to the model yields almost perfect classification. Instead of the individual inputs, the system now receives the triples as inputs. There are now eight possible inputs, one for each class element. When that element is presented, the neurons associated with it and only it are stimulated; for example, presentation of the big black square is done by stimulating its input neurons; when big black triangle is presented an entirely different set of neurons are stimulated.

The method is almost the same as in section 3.1. Instead of six possible inputs of 50 neurons, the system now has eight possible inputs of 100 neurons. Instead of activating 50 neurons for input, the system activates 60. There are 10 synapses from each neuron in the new *Triple Input* subnet to neurons in the *SOM* subnet. Like the *Input* subnet, learning is post-compensatory, and S_B is 5. Otherwise, the method remains the same.

T1	T2	T3	T4	T5	T6
100	100	100	100	100	100

Table 4. Performance on classification tasks by type with triples as inputs.

Table 4 shows the results. In this case, over 50 nets of each type, every classification is correct. It is possible to fail, but in this case it has not.

Following the natural kind analysis, each input feature is uniquely mapped to an output class. Consequently, this task is even easier than the Type 1 task in section 3.1, as there

are no superfluous inputs. For all of the types, eight attractors are formed. Each directly maps an input to an output.

While it is a solid learning system, it is of course, a poor cognitive model. The system should fail when subjects fail, and performance should vary on different types of classes.

3.3. *Classifying with a MLP*

In some sense, all six types of classification task are extremely simple. A simple caching algorithm that merely memorised the eight items and their classes would perform perfectly after one training instance. Consequently, standard machine learning algorithms should perform well.

For instance, the tasks were run with MLPs learning via Backpropagation. There were three inputs and one output, and the number of hidden nodes was varied. Inputs were 1.0 and 0.0 feature values as were the training class values. The simulations trained for 1000 iterations.

Number Hidden Nodes	1	2	3	4	5
T1	8	8	8	8	7.89
T2	6	7.25	7.89	8	7.9
T3	6	7.65	8	7.91	7.87
T4	8	8	8	7.99	7.98
T5	5.26	7.41	7.77	7.94	7.78
T6	6	6.39	7.89	7.98	7.48

Table 5. The average correct classifications out of eight for a particular type and number of hidden nodes in an MLP simulation. This is the average over 100 runs, and numbers like six or eight have resulted from every run giving six or eight.

The outputs were real valued, and the decision process for each item was if the value was > 0.5 it was of class 1, and if it was ≤ 0.5 it was a 0. Table 5 shows the average results over 100 runs of each condition. With one hidden node there is a linear separator. It is interesting that types 1 and 4 are classified perfectly; they are linearly separable. Types 2, 3 and 6 are each time have six eight correct; the separating plane has been drawn at one of the best possible places.

Beyond this, there is not much to comment on as a cognitive model. The results are almost perfect on all types with four hidden nodes. One could not reasonably posit that as time goes on, extra hidden nodes are added. Indeed, as the addition of the fifth hidden node (and more not shown) leads to a decay of performance while human subjects continue to do perfectly, this is a poor model. An MLP is a connectionist system inspired by neurons. In this case, it is not a good cognitive model.

MLPs can act as solid cognitive models (e.g. [Plaut et al., 1996]). However, in this case, the function that is learned is not a good match to the cognitive phenomenon. In the case of classification, the underlying phenomenon being modelled is an attractor state emerging from biological neurons, and modelling with attractor dynamics formed by neural mechanisms is closer to the underlying phenomenon. Even if it were a good cognitive model, if they performed equally well, the simulated neural system would be better than the MLP.

4. Combining Triples and Singletons

A system that combines the systems from sections 3.1 and 3.2 more closely echoes subjects' data. The gross topology of that system is shown in Figure 3.

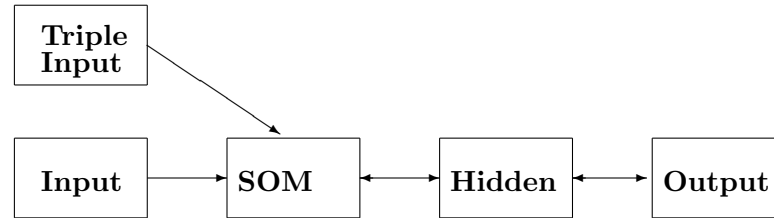


Figure 3. Gross topology of combined input mode classification simulations. Boxes represent subnets, and arrows excitatory connections between subnets. The *Triple Input* subnet has been added extending the topology from Figure 2. There are (not indicated) connections within the *SOM*, *Hidden*, and *Output* subnets.

The method combines those from sections 3.1 and 3.2. 50 and 60 neurons are externally stimulated in the *Input* and *Triple Input* subnets respectively. That is, when an input is presented, 50 of the 60 neurons are stimulated in the *Input* subnet, and 60 of the 100 neurons are stimulated in the *Triple Input* subnet in both training and testing. 20 of the 50 neurons are stimulated in the *Output* subnet during training.

The open variable is number of connections from each *Input* and *Triple Input* neuron to neurons in the *SOM* subnet. There are 20 from *Input* neurons (as in section 3.1), and a varying number from *Triple Input* neurons. The results shown in Table 6 indicate that as the number of *Triple* synapses increases, the classification performance increases.

Triple Synapses	T1	T2	T3	T4	T5	T6
5	100	94.8	96.8	99.9	80.2	93.6
10	100	98.1	98.4	100	85.6	98.4
20	100	99.3	99.7	99.9	87.2	99.0

Table 6. Performance with single and triple feature inputs on classification tasks by type. The row refers to number of synapses per *Triple* neuron to a *SOM* neuron.

This is qualitatively close to the Shepard data, but the behaviour of Type 5 is particularly bad. One option is that for this type of class, more time is required for training.

4.1. Time

All of the simulations described to this point have been trained for 24,000 cycles; each input set of 8 items has been presented 40 times. This is sufficient for several types, but Type 5 in particular might require longer. So, simulations with varying training lengths were run. These are illustrated in Figures 4 and 5.

The simulations have been run on each type for increasing durations. The durations have been increased by 600 cycles, a presentation of each item. The evaluation was then run over each of the 8 items 5 times. This was then repeated five times, and the average performance plotted. This shows that Type 1 is learned most quickly, converging around 15,000 cycles. Type 2 takes much longer, not converging until around 32,000 cycles. Type 3 is slightly quicker than 2, converging around 25,000 cycles.

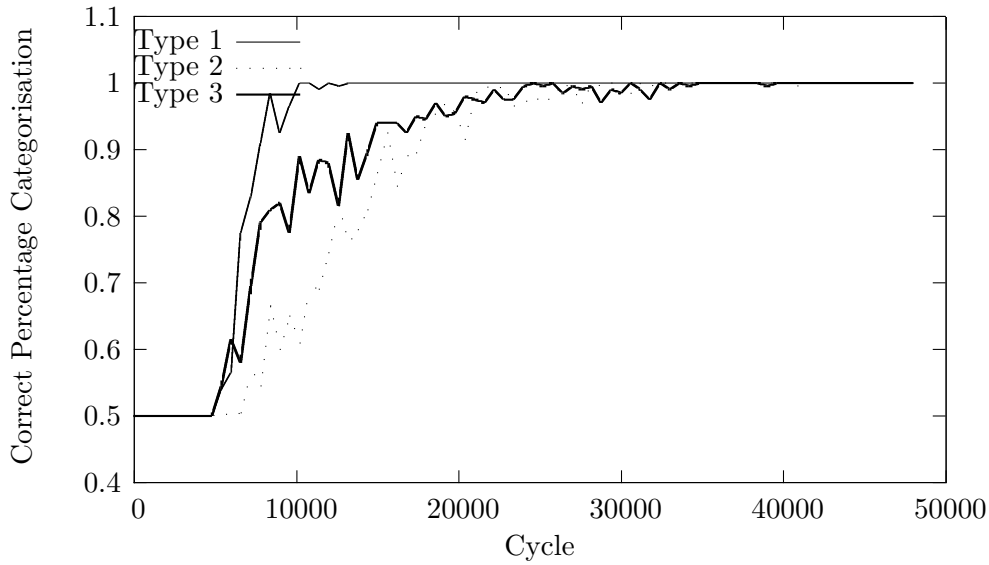


Figure 4. Classification performance by type and cycle. Results for Types 1, 2 and 3. Type 1 converges most rapidly.

Similar to Type 1, Type 4 converges quite quickly, around 18,000 cycles. Type 6 is similar to Type 2 converging around 25,000 cycles. Type 5 takes the longest, not converging until 48,000 cycles.

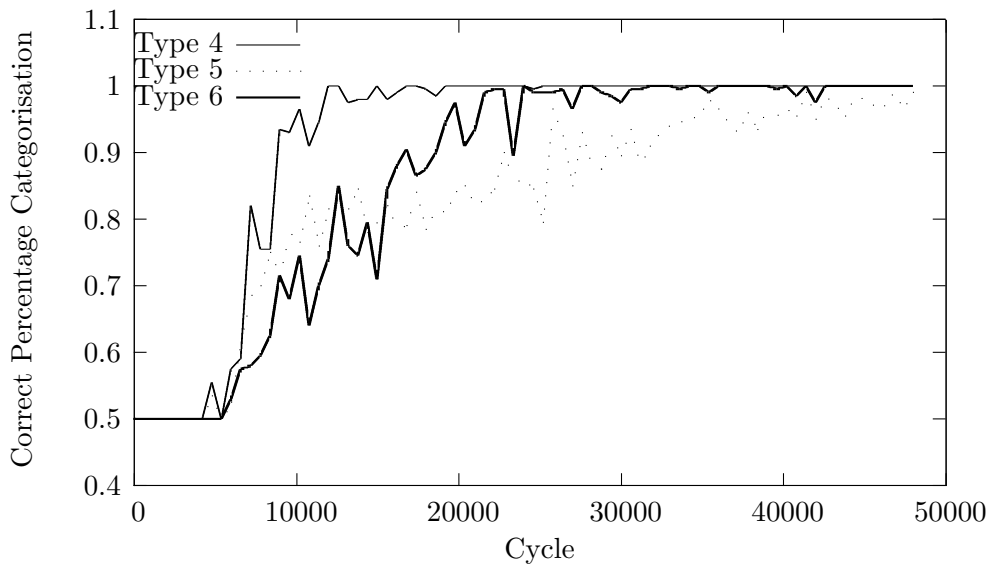


Figure 5. Classification performance by type and cycle. Results for Types 4, 5 and 6. Type 5 takes the longest.

If the standard experiment is run, training for 36,000 cycles, most types are classified perfectly (see Table 7). Types 3 and 5 are approaching complete classification, but are still not there. With 60,000 cycles or more of training, errors rarely occur. This echoes human performance!

Triple Synapses	T1	T2	T3	T4	T5	T6
10	100	100	98.8	100	97.5	100

Table 7. Performance with single and triple feature inputs on classification tasks by type trained for 36,000 cycles.

4.2. Synaptic Topology Analysis

Following the results seen in the classification performance of the network (Table 7), the network topology was analysed to see if any structural differences arise in the final state of different subnets. Gross topology measures such as rank, degree, clustering and eigenvalue analysis did not reveal any significant difference in the structure of the subnets when compared across different types of inputs (not shown). However, the number of cycles required to reach peak performance for the network shows a clear separation among the different types of inputs. This suggests that the input types are differentiated by the number of weight changes required for all the subnets to reach a steady state. The distribution of synaptic weights was analysed for all the connections, both feed-forward and recurrent, between different subnets, for example, *Input* to *SOM* subnet, *SOM* to *Hidden*, *Hidden* to *Hidden* subnet and so on. The weights were compared after running the simulation for 60,000 cycles, which has peak performance for all input types. The distributions of weights arising from each input type were compared using the two-sample Kolmogorov-Smirnov (KS) test after collating results from 10 different runs. The intention of using the KS test was not so much to examine if the weight distributions arise from the same probability distribution but it was more to get an estimate of how much the distributions differ for each input type in terms of their shape or location. Thus the p-values reported below in Table 8 are not a test of the 'significance' of the results but instead are more to quantify the difference in distributions.

Hidden to Hidden Synapses	T1	T2	T3	T4	T5	T6
T1	1.0	.052	.995	.995	.346	.016
T2		1.0	.052	.230	.494	.995
T3			1.0	.832	.346	.008
T4				1.0	.665	.147
T5					1.0	.346

Table 8. p-values for comparing the distributions of synaptic weights for the 6 different types of inputs using the Komolgorov-Smirnov test. Nets trained for 60,000 cycles.

Amongst all the subnets, meaningful differences were only seen in the recurrent connections of the *Hidden* subnet while other subnets did not show substantial type specific differences. As seen in Table 8, T5 seems to differ from all other nets in how the weights get distributed once the network has reached peak performance with a weak similarity only with T4. T6, barring its similarity with T2, also seemingly differs quite a bit from all other nets. These comparative trends are not only seen in figures 4 and 5, as the rate of rise in the classification performance, which partly reflects the findings in Table 8, but also in the similarity of classification performances seen in Table 6. Hence the KS test strongly suggests that the bulk of the classification differences among input types occurs in the *Hidden* subnet. The *Hidden* subnet forms the last stage before outputs are decided and apart from being recurrently connected with itself, it also gets afferent connections from the *SOM* subnet and importantly from the *Output* subnet, thus having a rich information flow within it.

This analysis shows that, depending on the type of class they have as input, the

networks have learned something substantially different. Even at this coarse level, the weight distributions are different. This relates to the number of and size of the attractor states that are formed. Type 1 and 4 require only two attractor states, and thus are similar. Type 2 requires four equally sized attractor states, and that is similar to type 5, which requires 4 states though two are large and two small. Type 2 is also similar to Type 6, which requires eight equally sized states. Type 3 is most similar to Type 4 as its two attractor states are more diffuse.

5. Discussion

A valid criticism of the model is that it is post hoc. The authors knew about the data on human performance on classification tasks before developing the model. A reasonable response to this is the prior classification systems; the current model is a variant of those models. The only major addition is the triples as input.

One key point of this paper is that different types of classes take longer to learn. Moreover, this is based on two main factors: the combination of the relationship between class members and singleton input features; and the way features are shared between class members.

The relationship between class members and singleton input features is described in section 3.1. It is relatively simple for even the standard system to learn Type 1 and 4 classes almost perfectly because all the members of the classes of those types share features. This propagates through the middle subnets to form two attractor states, one for each class. This also aligns with natural kinds. Though very simple classes, these two types are natural kinds; Type 1 is a particularly simple form of natural kind as it is based on a necessary and sufficient feature.

Types 2 and 6 take appreciably longer to learn than Types 1 and 4. This is due to the lack of direct information from the singleton input features. In both cases, each feature value has two elements in each class. These do not align with natural kinds. In Type 2, there are four attractor states, and in Type 6, there are eight. In the above simulations, these require the triple inputs to be learned.

Types 3 and 5 do have a relationship between class inputs and feature membership that should support their being learned. However, the second factor, the way features are shared between class members, becomes important. Note in Figure 5 how Type 5 starts to improve before Type 6. This is the benefit of the class to singleton input feature relationship; in the standard system, this takes it to near 75% classification performance. However, the other member of the class shares no features with the three that are correctly classified. It takes the system a long time to learn to correctly classify this element. The Type 3 class has two members that share no features, but they both share one and two features with the other members of the class. Thus, they also need help from the triple inputs, and take longer than all but Type 5 classes.

A benefit of the model is that the basic FLIF model has been used to perform a wide range of activities including virtual agents (e.g. [Huyck et al., 2011]) and neuro-cognitive models (e.g. [Huyck, 2009]). It is hoped that learning and using classes can be readily integrated into this framework.

One of the major questions raised by these simulations is the combination of features. The system makes use of triples of features; indeed it performs perfectly on these inputs alone. In the human brain, the system would have to learn to combine the single input features into these triples. How can this be done?

It is not surprising that feature combination is important. It is an important topic in

machine learning [Lanckriet et al., 2004]. Given n binary features, there are 2^n ways to combine them.

While the simulations do show varied behaviour in learning different types of classes, this variance does not match that shown in Shepard, Hovland and Jenkins' subjects. The precedence shown in the above simulations is Type 1 > 4 > 3 > 2 & 6 > 5; this does match the natural kind analysis the authors proposed in section 2.2. However, the subjects had a precedence Type 1 > 2 > 3 & 4 & 5 > 6. In this sense it is at best a half-cognitive model.

Some of the subjects' tasks involved explicitly creating rules. Clearly, the model presented here is incapable of creating rules. One could speculate that this may have to do with the two Psychological systems described by, among others, Tversky and Kahneman [Kahneman, 2011]. The classification simulations described above account for the fast, automatic, parallel, system 1. The subjects are using both system 1 and system 2, the slow system that can use and generate rules. As the subjects are getting feedback, and concentrating on the task of learning the classes, they may form rules. Indeed this is one of the tasks they must do. So, a full cognitive model would incorporate this functionality. Do note that this speculation provides a testable prediction. If a class (which could be of the two class three binary feature form) was learned solely by system 1, it should follow the natural kind precedence. This might be done by the subject learning the classes passively, while concentrating on some other task. Also note that Shepard, Hovland, and Jenkins consider the natural kind explanation in the form of generalization theory [French, 1953]. Finally, note that Shepard, Hovland and Jenkins give an explanation consistent with the two system speculation; generalization theory works, but in the case of some types, an extra rule based mechanism is needed.

Another valid criticism is that the model is an extreme simplification of the actual neural topology. For instance, the model does no actual vision. This particular concern can be partially addressed by the independence of the visual features (shape, colour and size) in the original Shepard et al. work; their work implied that this type of learning problem was general across modality. None the less, it is clear that these classes are not learned by humans with a few thousand neurons. It is hoped that this neural model is indicative of the actual neural processes involved in the eight feature two class classification, but also in other classification tasks.

While the neural model is relatively simple, it is also not widely used. The use of fatigue has been described elsewhere [Huyck and Mitchell, 2014], but in this paper, the use of hypo-fatigue to drive firing in unstimulated neurons, allows the attractors to spread into areas that have not been directly stimulated from the environment. It also seems likely that this also spreads activity to neurons that are not already in attractors. Similarly, compensatory learning is relatively simple but is also not widely used, and its use has been described elsewhere [Huyck, 2007, Huyck and Mitchell, 2014]. In this case, the post-compensatory rule supports the spread of activation from the *Input* subnet to the others, and the pre-compensatory rule encourages the spread to the other subnets. This can be seen when running a classification simulation; initially, there is no firing in the *SOM* and *Hidden* subnets, but over the epochs, it increases. In general, the compensatory mechanism reduces the likelihood that a neuron will become involved in too many attractors. The neural and learning models are undoubtedly simplifications of the actual biological system, but it is hoped that they are simplifications that include much of the important computational detail.

This paper has compared the spiking model to MLPs, but could have also made use of other systems such as Hopfield nets and other types of attractor nets. There is a large body of simulation work with Hopfield nets (e.g. [Amit, 1989]); this work benefits from a solid mathematical framework supported by restricted topologies, for ex-

ample bi-directional connections; unfortunately, these topologies are not biologically realistic. There are limitations in their ability to learn classes with Hebbian like rules [Jacyna and Malaret, 1989], but that should not affect their ability to learn the Shepard et al. classes. Instead, all of the different types of classes should be learned at about the same rate, making the Hopfield model a poor one.

Similarly, there are solid neural models that do not spike. A great deal of work has been done with rate coded neurons (e.g. [O'Reilly, 1996]), and even persistently active neurons (e.g. [Rumelhart and McClelland, 1982]). The use of a Hebbian learning rule, based on spikes, is not directly relevant to these systems, but similar rules can be developed. Consequently, it would be interesting to see a cognitive model based on continuously valued neurons. It would be particularly interesting if that model implemented the full cognitive model using both systems.

6. Conclusion

The full model from Figure 3 is a sound classification system. In particular, it accounts for all six types of binary classifications. Moreover, it not only learns sound classifiers, it learns them in a type dependent manner, so that more difficult classes take longer to learn.

While the standard model works well for natural kinds, it does not work well for other types. The model has been extended to include triplets as inputs (to yield the full model). These triplets are a weakness of the model because the system does not learn them on its own. Biological neural systems can.

Both models are good because the neuronal models and learning rules are neuropsychologically reasonable. These are biological models. This is the only neural cognitive model of classification the authors are aware of. However, neurons in the systems do not directly correlate with neurons in a brain. For instance, the models only use a few thousand neurons, while many more are almost certainly used, and reused in a brain for these tasks. Instead the models are indicative of mechanisms that might be used in a brain.

Even the full model is not a particularly good cognitive model. While it does learn the different types of classes at different rates, the rates do not echo the precedence shown in human subjects. Instead, the authors propose that it is a good half-cognitive model. Following the two system Psychological hypothesis [Kahneman, 2011], the full model may echo the data presented by the first fast system. What is needed to make it a whole neuro-cognitive model of the task is to add a spiking model of the second slow system.

It is difficult to build spiking neural cognitive models of complex cognitive functions, particularly when these functions involve learning. However, by building these models, the understanding of neuro-psychological functioning can be advanced.

Acknowledgements:

This work was supported by the Human Brain Project Grant 604102, Neuromorphic Embodied Agents that Learn. Multi-Layer perceptron code was developed by David Adler.

References

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

- [Ashby and O'Brien, 2005] Ashby, F. and O'Brien, J. (2005). Category learning and multiple memory systems. *Trends in cognitive sciences*, 9(2):83–89.
- [Bi and Poo, 1998] Bi, G. and Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18:24:10464–10472.
- [Bielza et al., 2011] Bielza, C., Li, G., and Larranaga, P. (2011). Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*, 52:6:705–727.
- [Bohte et al., 2002] Bohte, S. M., Poutr, H. L., and Kok, J. N. (2002). Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer rbf networks. *Neural Networks, IEEE Transactions on*, 13(2):426–435.
- [Brette and Gerstner, 2005] Brette, R. and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.*, 94:3637–3642.
- [Brette et al., 2007] Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J., Diesmann, M., Morrison, A., Goodman, P., Harris, F., Zirpe, M., Natschlagel, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Dafison, A., ElBoustani, S., and Destexhe, A. (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 23:349–398.
- [D'Espisito, 2007] D'Espisito, M. (2007). From cognitive to neural models of working memory. *Philosophical Transactions of the Royal Society*, 362:761–772.
- [French, 1953] French, R. (1953). Number of common elements and consistency of reinforcement in a discrimination learning task. *Journal of experimental psychology*, 45(1):25–34.
- [Friedman et al., 1997] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, 29(2-3):131–163.
- [Ghosh-Dastidar and Adeli, 2009] Ghosh-Dastidar, S. and Adeli, H. (2009). Spiking neural networks. *International journal of neural systems*, 19(04):295–308.
- [Hebb, 1949] Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. J. Wiley & Sons.
- [Hettich and Bay, 1999] Hettich, S. and Bay, S. (1999). UCI KDD archive.
- [Hodgkin and Huxley, 1952] Hodgkin, A. and Huxley, A. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544.
- [Huyck, 2007] Huyck, C. (2007). Creating hierarchical categories using cell assemblies. *Connection Science*, 19:1:1–24.
- [Huyck, 2009] Huyck, C. (2009). A psycholinguistic model of natural language parsing implemented in simulated neurons. *Cognitive Neurodynamics*, 3(4):316–330.
- [Huyck et al., 2011] Huyck, C., Belavkin, R., Jamshed, F., Nadh, K., Passmore, P., Byrne, E., and D.Diaper (2011). CABot3: A simulated neural games agent. In *7th Intl W/shop on Neural-Symbolic Learning and Reasoning, NeSYS'11*.
- [Huyck and Mitchell, 2014] Huyck, C. and Mitchell, I. (2014). Post and pre-compensatory Hebbian learning for categorisation. *Computational Neurodynamics*, 8:4:299–311.
- [Huyck and Parvizi, 2012] Huyck, C. and Parvizi, A. (2012). Parameter values and fatigue mechanisms for flif neurons. *Journal of Systemics, Cybernetics and Informatics*, 10:4:80–86.
- [Jacyna and Malaret, 1989] Jacyna, G. and Malaret, E. (1989). Classification performance of a hopfield neural network based on a hebbian-like learning rule. *IEEE*

- Transactions on Information Theory*, 35:2:263–280.
- [Kahneman, 2011] Kahneman, D. (2011). *Thinking, fast and slow*. Macmillan.
- [Kohonen, 1997] Kohonen, T. (1997). *Self-Organizing Maps*. Springer.
- [Kotsiantis et al., 2007] Kotsiantis, S., Zaharakis, I., and Pintelas, P. (2007). *Supervised machine learning: A review of classification techniques*. Springer.
- [Lanckriet et al., 2004] Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L., and Jordan, M. (2004). Learning the kernel matrix with semidefinite programming. *Machine Learning Research*, 5:27–72.
- [Lapicque, 1907] Lapicque, L. (1907). Recherches quantitatives sur l’excitation lectrique des nerfs traite comme une polarisation. *J. Physiol. Pathol. Gen*, 9:620–635.
- [Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). *Perceptrons: An Intoduction to Computing Geometry*. MIT Press.
- [Mongillo et al., 2008] Mongillo, G., Barak, O., and Tsodyks, M. (2008). Synaptic theory of working memory. *Science*, 319:1543–1546.
- [Oja, 1982] Oja, E. (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273.
- [O’Reilly, 1996] O’Reilly, R. (1996). *The Leabra Model of Neural Interactions and Learning in the Neocortex*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.
- [Plaut et al., 1996] Plaut, D., McClelland, J., Seidenberg, M., and Patterson, K. (1996). Understanding normal and impaired word reading: Computational principles in quasi-regular domains. *Psychological Review*, 103:1:56–115.
- [Riesenhuber and Poggio, 1999] Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025.
- [Rosch and Mervis, 1975] Rosch, E. and Mervis, C. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 7:573–605.
- [Rumelhart and McClelland, 1982] Rumelhart, D. and McClelland, J. (1982). An interactive activation model of context effects in letter perception: Part 2. the contextual enhancement and some tests and extensions of the model. *Psychological Review*, 89:1:60–94.
- [Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.
- [Senn et al., 2002] Senn, W., Schneider, M., and Ruf, B. (2002). Activity-dependent development of axonal and dendritic delays, or, why synaptic transmission should be unreliable. *Neural Computation*, 14(3):583–619.
- [Shepard et al., 1961] Shepard, R., Hovland, C., and Jenkins, H. (1961). Learning and memorization of classifications. *Psychological Monographs*, 75:517:1–42.
- [Song et al., 2000] Song, S., Miller, K., and Abbott, L. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926.
- [Stein, 1967] Stein, R. (1967). Some models of neuronal variability. *Biophysical journal*, 7(1):37–68.
- [Stewart and Eliasmith, 2011] Stewart, T. and Eliasmith, C. (2011). Neural cognitive modelling: A biologically constrained spiking neuron model of the tower of hanoi task users. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, pages 65661–65667.
- [Wade et al., 2010] Wade, J., McDaid, L., Santos, J., and Sayers, H. (2010). Swat: a spiking neural network training algorithm for classification problems. *IEEE Transactions on Neural Networks*, 21(11):1817–1830.
- [Wittgenstein, 1953] Wittgenstein, L. (1953). *Philosophical Investigations*. Blackwell, Oxford.

- [Yarkoni et al., 2011] Yarkoni, T., Poldrack, R., Nichols, T., Essen, D. V., and Wager, T. (2011). Large-scale automated synthesis of human functional neuroimaging data. *Nature Methods*, 8:8:665–670.
- [Zipser et al., 1993] Zipser, D., Kehoe, B., Littleword, G., and Fuster, J. (1993). A spiking network model of short-term active memory. *The Journal of Neuroscience*, 13:8:3406–3420.