Using Neural Networks to Model Bounded Rationality in Interactive Decision-Making*

By Daniel Sgroi†

September 2, 2003

Abstract

This paper considers the use of neural networks to model bounded rational behaviour. The underlying theory and use of neural networks is now a component of various forms of scientific enquiry, be it modelling artificial intelligence, developing better pattern recognition or solving complex optimization problems. This paper surveys the recent literature in economics on their use as a plausible model of learning by example, in which the focus is not on improving their ability to perform to the point of zero error, but rather examining the sorts of errors they make and comparing these with observed bounded rational behaviour.

Key words: Neural Networks, Bounded Rationality, Learning, Repeated Games, Industrial Organization.

Journal of Economic Literature Classification Numbers: C72, D00, D83.

*The author is grateful to AEA Technology plc for financial support, and to Daniel J. Zizzo and an anonymous referee for useful comments.

†Churchill College and Department of Applied Economics, University of Cambridge, U.K. Mailing Address: Department of Applied Economics, Sidgwick Avenue, Cambridge, England CB3 9DE; e-mail: daniel.sgroi@econ.cam.ac.uk.

I. INTRODUCTION

Neural networks can be loosely defined as *artificial intelligence models inspired by analogy with the brain and realizable in computer programs.* Simply put, a neural network is trained by receiving a stream of data corresponding to an input and a stream of data corresponding to an output. The network builds an algorithm designed to find a pattern linking the input to the output, and when a good enough algorithm is found, the network is ready to try new input data in an attempt to predict the output.

A classic example is in signature recognition. Thousands of signatures are fed to a new neural network, complete with the names of those responsible for the signatures. The network develops an algorithm for linking the signatures to the known names. This might involve finding patterns such as the curves and slopes of certain letters. Then one of the named individuals might offer the network a new signature, with the challenge being put to the network of discovering the identity of the signatory, based on the pattern-recognizing algorithm the network has developed. The neural network is trained in this way until it can consistently get the names right a high fraction of times. When we are convinced that the neural network has developed a good algorithm for recognizing signatures, we can then release it into the real world to contend with thousands of new signatures and signatories. Another example, of more use to econometrics would be to use neural networks as a means of numerical optimization. Once again a network is trained on observed data, and asked to predict what new input would produce as output, in a broadly similar way to more standard methods such as regression. In these sorts of uses the main concern is making sure the neural network gets things right.

However, there is another, smaller literature which aims to exploit not simply the problem-solving ability of neural networks, but the primary characteristic of neural networks which set them apart from other forms of numerical optimization: that they are based on a model of learning by example constructed with

biological plausibility in mind. In this literature the aim is to keep the neural network as biologically plausible as possible and then see what success rate such a restricted network can achieve. To give an example, consider a neural network trained to do very well at a certain game, by receiving a stream of data revealing the correct output for given inputs. Having mastered this game the neural network will have necessarily built up an algorithm used to transform the inputs to the correct output. The network is then asked to use this algorithm to play a completely new game. The results of this experiment provide us with not the best possible way to play the new game, but an example of how a model built by analogy with the human brain, having learned one game, might play a new one. The applications of this are considerable: how would a firm used to monopoly switch to dealing with a duopoly; how might a master chess player function playing checkers; how might a child trained to be well-behaved at home perform in her first day at school, etc. The important point is that the network learns through example by being exposed to understood sets of inputs and outputs. From these it develops a method of pattern recognition, or algorithm, linking the inputs to outputs. Then the network faces new situations and must try to solve them based on this method. Certainly we could improve the network's odds of success exogenously by adding known means of solving problems, but the interesting question for an economist studying behaviour is how well will such a network perform, not what we have to do to get its success rate up to 100%.

The next section provides a formal discussion of neural networks in the context of an N-player game with a non-trivial finite action space. This section can be skipped by those with a good working knowledge of neural networks. Section 3 attempts a non-formal characterization of the key elements of neural networks which make them suitable models for bounded rational learning. Section 4 summarizes the literature on neural network learning in general and more specifically how a neural network can learn how to play new games having played similar games in the past. Section 5 surveys several interesting examples of the use of neural networks in economic decision-making including: neural

network learning in repeated instances of the same game especially repeated play of the Prisoner's Dilemma; applications to industrial organization; and intergenerational learning. Section 6 concludes.

II. UNDERSTANDING NEURAL NETWORKS

This section provides an introduction to the functioning of feedforward neural networks, focusing on their use as possible models of bounded rational behaviour and can be skipped by those with a working knowledge of neural networks and backpropagation. There exist many books which provide good overviews of neural network learning, such as White (1992) and Anthony and Bartlett (1999), and some papers with brief primers on feedforward neural networks, such as Leshno *et al.* (2003) and Sgroi and Zizzo (2002), though the latter is more restricted in its focus.[1] Finally, Cho and Sargent (1996) provides an excellent introduction with special emphasis on a subset of feedforward neural networks known as perceptrons, the simplest form of neural network.

In this section we fix the notion of neural network learning within a game theoretic framework (an N-player normal form game with a non-trivial finite action space). The major assumption which greatly simplifies matters is to consider one unique optimal action, which corresponds to a unique pure strategy Nash equilibrium.

II. 1 Defining a Neural Network

A neural networks typically learns by exposure to a series of examples or *training set*, and adjust the strengths of the connections between its nodes. It is then able to perform well not only on the original training set, but also when

[1] Sgroi and Zizzo (2002) uses a similar set of definitions though does so in the specific case of 3×3 games played by 2 players, and considers a neural network with exactly 2 hidden layers.

facing problems never encountered before. Consider a feedforward neural network, or more simply C to be a machine capable of taking on a number of states, each representing some computable functions mapping from input space to output space, with *hidden* layers of further computation between input and output. Hidden layers can be thought of as intermediate layers of computation between input and output. Since we see the input go in, and the output come out, but do not directly see the activity of intermediate layers, they are conventionally called *hidden*. The following definition formalizes what we mean by a neural network.

Definition 1 *Define the neural network as* $C = \langle \Omega, X, Y, F \rangle$ *where* Ω *is a set of states,* $X \subseteq \mathsf{R}^n$ *is a set of inputs,* Y *is a set of outputs and* $F : \Omega \times X \mapsto Y$ *is a parameterized function. For any* ω *the function represented by state* ω *is* $h_\omega : X \mapsto Y$ *given by* $h_\omega(x) = F(\omega, x)$ *for an input* $x \in X$*. The set of functions computable by* C *is* $\{h_\omega : \omega \in \Omega\}$*, and this is denoted by* H_C*.*

Put simply, when C is in state ω it computes the function h_ω, providing it is computable. In order to reasonably produce answers which correspond to a notion of correctness such as a Nash strategy, which explicitly chooses the best reply to the optimal action of a rival player in a game, we need to *train* the network. First we will consider the form of the network in practice, and start by defining an activation function.

Definition 2 *An activation function for node* i *of layer* k *in the neural network* C *is of the logistic form*

$$a_i^k = \frac{1}{1 - \exp\left(-\sum_j w_{ij}^k u_{ij}^{k-1}\right)} \tag{1}$$

where u_{ij}^k *is the output of node* j *in layer* $k-1$ *sent to node* i *in layer* k*, and* w_{ij} *is the weight attached to this by node* i *in layer* k*. The total activation flowing into node* i*,* $\sum_j w_{ij}^k u_{ij}^{k-1}$*, can be simply defined as* t_i*.*

The situation faced by the neural network is well modelled by a normal form

game $G = \langle N, \{A_i, u_i\}_{i \in N} \rangle$ of full information with a unique pure strategy Nash equilibria. Actions are given by $a_i \in A_i$. Feasible action combinations are given by $A = A_1 \times A_2 \times ... \times A_N$. Payoffs for player i are given by $u_i : A_i \mapsto \mathsf{R}$ which is a standard von Neumann-Morgenstern utility function. Payoffs are bounded, so $\exists Q \geq 0$ such that $\mid u_i(a) \mid \leq Q$ for all a and since we can normalize payoffs to be drawn from a uniform $[0, 1]$ and then revealed to the players before they select an action, we have $\forall_{i,a}$, $\sup u_i(a_i) = 1$.

II. 2 Inputs and Outputs

Consider a set of input nodes each recording and producing as an output a different value from the grid of normal-form payoffs expressed as an *input vector* $x_k = \left(x_k^1, x_k^2 ..., x_k^{N \times A_N \times A_N}\right)$; for example in a two player game where each player has 2 strategies this provides an input vector with $2 \times 2 \times 2 = 8$ elements. Now consider a second set of nodes (the first hidden layer). Each node in this second layer receives as an input the sum of the output of all input nodes transformed by the activation function of node i in layer 2. All of the nodes in the second layer send this output a_i^2 to all nodes in the second hidden layer, which weights the inputs from all i of the first hidden layer, by the activation function to produce a_i^3, and so on. Eventually these numbers are sent to the final layer of nodes to produce an output y_k which forms a vector representing the choice of strategy in the game. For example, the vector $(1, 0, ..., 0)$ would imply that the neural network player's choice is the pure strategy embodied by selecting the top row in a normal-form game, $(0, 0, ..., 1)$ would imply the penultimate row, and $(0, 0, ..., 0)$ the bottom row. Of course there is nothing restricting the neural network from choosing values other than 0 or 1, so it might select $(0.8, 0, ..., 0)$ which would suggest that it is certain it does not wish to pick any strategies except the top row strategy or possibly the bottom row strategy though with less confidence. The network's output is interpreted as a probability vector so their sum must add to 1 (the residual forming the probability placed

on the bottom row in the normal form payoff matrix interpretation).

II. 3 Training

Training essentially revolves around finding the set of weights that is most likely to reproduce the actual Nash equilibrium of the game faced (for example, we might wish to train a neural network to produce an output $(1, 0, ..., 0)$ for a given set of inputs). During training C receives a sequence of random games until some stopping rule determines the end of the training at some round T. The training sample consists of M random games. If $T > M$, then (some or all of) the random games in M will be presented more than once. The training sample is a sequence of input vectors as defined above except that the elements of each input vector must ensure a unique pure strategy Nash equilibrium. If any x_k fails to achieve this a new vector is selected and so on until this condition is satisfied. C transforms input vector to output vector and does so M times with a new set of inputs x_k and outputs y_k. Assume that each vector x_k is chosen independently according to a fixed probability distribution P_T on the set X subject to the requirement that the game has a unique Nash equilibrium. The probability distribution is fixed for a given learning problem, but it is unknown to C, and for our purposes will be taken to be a uniform $[0, 1]$. The information presented to C during training therefore consists only of several sequences of numbers.

Definition 3 *For some positive integer m, the network is given a training sample:*

$$\begin{aligned} x^M &= \left(\left(x_1^1, x_1^2 ..., x_1^{N\times A_N \times A_N}\right), \left(x_2^1, x_2^2 ..., x_2^{N\times A_N \times A_N}\right), \right. \\ &\quad \left. ..., \left(x_M^1, x_M^2 ..., x_M^{N\times A_N \times A_N}\right)\right) \\ &= (x_1, x_2, ..., x_M) \in X^M \end{aligned}$$

The labelled examples x_i are drawn independently according to the probability

distribution P_T subject to the condition that each vector x_t ensures the existence a unique Nash equilibrium in pure strategies. If this condition fails a new vector is drawn from P_T. A random training sample of length M is an element of X^M distributed according to the product probability distribution P^M.

Assume that $T > M$. In this case, training might be *sequential*: after $q \times M$ rounds (for any positive integer q s.t. $q \times M < T$), M is presented again, exactly in the same order of games. If training is *random without replacement*, it is less restricted to the extent that the order in which the random games are presented each time is itself random. If training is *random with replacement*, in each round the network is assigned randomly one of the random games in M, until round T.

II. 4 Backpropagation

Having selected a sample sequence of inputs, x, and determined the unique Nash strategy associated with each, α, we need to consider how C *learns* the relationship between the two, to ensure that its output y_k will approach the Nash strategy appropriate for an input vector x_k and do so for the full sample of input vectors. The most common method used is backpropagation. First let us define the error function and then go on to define backpropagation.

Definition 4 *Define the network's root mean square error ε as the root mean square difference between the output y and the correct answer α over the full set of $q \times M$ games where individual games are indexed by i, so our error function is:*

$$\varepsilon \equiv \left(\sum_{i=1}^{q \times M} (y_i - \alpha_i)^2 \right)^{\frac{1}{2}}$$

The aim is to minimize the error function by altering the set of weights w_{ij} of the connections between a typical node j (the sender) and node i (the

receiver) in different layers. These weights can be adjusted to raise or lower the importance attached to certain inputs in the activation function of a particular node. The correct answer here is the vector associated with the unique Nash equilibrium in pure strategies. In principle we could use any other measure, including for example training the neural network to select the best or even worst outcome in terms of game payoff.

Backpropagation is a form of numerical analysis similar to gradient descent search in the space of possible weights. Following Rumelhart *et al.* (1986) consider a function of the following form:

$$\triangle w_{ij} = -\eta \frac{\partial \varepsilon}{\partial w_{ij}} = \eta k_{ip} o_{jp} \tag{2}$$

The weight of the connection between the sending node j and receiving node i is denoted by w_{ij}. Since ε is the neural network's error, $\partial\varepsilon/\partial w_{ij}$ measures the sensitivity of the neural network's error to the changes in the weight between i and j. There is also a *learning rate* given by $\eta \in (0, 1]$: this is a parameter of the learning algorithm and must not be chosen to be too small or learning will be especially vulnerable to local error minima. Too high a value of η may also be problematic as the network may not be able to settle on any stable configuration of weights. Define $\partial\varepsilon/\partial w_{ij} = -k_{ip} o_{jp}$ where o_{jp} is the degree of activation of the sender node j for a given input pattern p. The higher is o_{jp}, the more the sending node is at fault for the erroneous output, so it is this node we wish to correct more. k_{ip} is the error on unit i for a given input pattern p, multiplied by the derivative of the output node's activation function given its input. Call g_{ip} the goal activation level of node i for a given input pattern p. Since the first derivative $f'(t_{ip})$ of the receiving node i in response to the input pattern p is equal to $o_{ip}(1 - o_{ip})$ for a logistic activation function, the output nodes k_{ip} can be computed as:

$$k_{ip} = (g_{ip} - o_{ip}) f'(t_{ip}) = (g_{ip} - o_{ip}) o_{ip}(1 - o_{ip}) \tag{3}$$

Now assume that a network has N layers, for $N \geq 2$. As above, we call layer 1 the input layer, 2 the layer which layer 1 activates (the first hidden layer), and so on, until layer N the output layer which layer $N-1$ activates. We can now define the backpropagation learning process.

Definition 5 *Using backpropagation, we first compute the error of the output layer (layer N) using equation 3, and update the weights of the connections between layer N and $N-1$, using equation 2. We then compute the error to be assigned to each node of layer $N-1$ as a function of the sum of the errors of the nodes of layer N that it activates. Calling i the hidden node, p the current pattern, and β an index for each node of layer N (activated by i), we can use:*

$$k_{ip} = f'(t_{ip}) \sum_{\beta} k_{\beta p} w_{\beta i} \tag{4}$$

to update the weights between layer $N-1$ and $N-2$, together with equation 2. We follow this procedure backwards iteratively, one layer at a time, until we get to layer 1, the input layer. A variation on standard backpropagation would involve replacing equation 2 with a momentum function of the form:

$$\triangle w_{ij}^{t} = -\eta \frac{\partial \varepsilon^{t}}{\partial w_{ij}^{t}} + \mu \triangle w_{ij}^{t-1} \tag{5}$$

where $\mu \in [0,1)$ and $t \in \mathbb{N}^{++}$ denotes the time index (so an example game, vector x, is presented in each t during training).

Momentum makes connection changes smoother by introducing positive autocorrelation in the adjustment of connection weights in consecutive periods. C's connection weights are updated using backpropagation until round T. T itself can be determined exogenously by the builder of the neural network, or it can be determined endogenously by the training process by setting a require-

ment in terms of maximum allowable error.

II. 5 Summary

A neural network is continuously shown and reshown a set of games until it recognizes the optimal way to play these games. Recognition comes in terms of a simple measure of mean squared error away from the vector representation of the pure Nash equilibrium. This is widely known in the neural network literature as supervised learning and accords with an intuitive notion of a teacher continuously correcting the behaviour of a student until behaviour is close to that expected in a Nash equilibrium. When it has achieved this or close enough to it (when it knows the best way to play in this set of games) it is shown some different games and asked to find the Nash equilibria for these without ever having seen these new games before. It can however use the algorithms (rules) it has already learned in order to allow it to choose correctly the Nash equilibria in those games which it has seen before (the training set).

The next section examines some key characteristics of neural networks and sees where these come into play in the literature on neural network learning in economic decision-making.

III. CHARACTERISTICS OF NEURAL NETWORKS

Consider the way in which a neural network develops. It learns by exposure to a series of examples (a training set) and then adjusts the strengths of the connections between its nodes. Having developed a general algorithm it is able to do well not only on the original training set, but also potentially when facing problems never encountered before. A feature of biological brains is that the connections between neurons are of different strengths, and that they can either increase or decrease the firing rate of the receiving neuron. In neural networks, this is modelled by associating a connection weight to each connection. This

weights the input from the sending node to the receiving node. Since the weight can be either positive or negative, the activation of a node will either increase or decrease the activation of the receiving node. A typical network has an input layer of nodes receiving stimuli from the outside (as real numbers). This input is then transmitted to the nodes the input layer is connected to and multiplied by the respective connection weights. Each node on the downstream layer receives input from many nodes. The sum is then transformed according to the activation function and the result is transmitted to the nodes in the further downstream layer. In such a way, the network processes the input until it reaches the output nodes (the output layer), in the form of new real-valued numbers. The activation level of the output nodes expresses the outcome of network processing, i.e., the network's decision. In other words, the network computes a complex nonlinear transformation mapping the input (a vector of numbers) into an output (another vector of numbers). For the various papers discussed here optimality is often directly associated with playing Nash strategies or maximizing a utility of profit function.

III. 1 Limits to Biological Plausibility

Generally, the optimum parameter or set of parameters for a neural network cannot be calculated analytically when a model is nonlinear, and so we must rely on a form of numerical optimization. Backpropagation (see Rumelhart *et al.,* 1986)) is the most standard method used in the neural network literature for building practical neural networks. The basic intuition behind backpropagation is that of psychological reinforcement: the economic decision-maker tries to learn how to perform better in the task, and the more disappointing the outcome (relative to the "correct" outcome), the deeper the change in connection weights will be. In this sense it is similar to reinforcement learning mechanism discussed in Roth and Erev (1995) and Roth and Erev (1998).

Backpropagation requires a teacher explicitly telling the correct answer dur-

ing training, and this might appear too strong a requirement: it renders backpropagation a more powerful algorithm than is biologically plausible. Backpropagation is more powerful also in another sense: it adjusts individual connection weights using *global* information on how to best allocate output error which is unlikely to occur in biological brains (see MacLeod *et al.,* 1998). These limitations, however, should not be overstated: what they suggest is that backpropagation might be a plausible *upper bound* to the learning of biological neural networks of some given size. Conversely, stronger learning algorithms, of the kind used by White (1992) to show learnability, are much further from biological or cognitive plausibility (White, 1992, page 161). Hence, the non-learnability results with backpropagation discussed in the next section cannot be easily dismissed as an artificial product of too weak a learning rule.

In practice, we do know that a neurotransmitter, dopamine, plays a role in biological neural networks analogous to that of the teacher in the backpropagation algorithm: the activation level of dopamine neurons may work as a *behavioural adaptive critic*, i.e. it tells the agent how to adapt its behaviour to successfully deal with a task (Zizzo, 2002). Added to this is the fact that evidence exists to demonstrate that appropriately constructed neural networks can closely approximate the behaviour of human subjects in experimental laboratories in economic situations (as in Sgroi and Zizzo, 2002, and Leshno *et al.,* 2003 discussed in the next two sections).

III. 2 Learning by Example

The key to the learning process is repeated exposure to examples. The players are not assumed to have perfect access to models of the real world, nor are any rules explicitly taught to the network; rather they are simply subjected to a sequence of example games, and then asked to assimilate what general knowledge they can from these examples to play new, never before seen, games. Economic agents may face a large number of decisions throughout their

life. However, notwithstanding the role of dopamine neurons, it is unlikely that most of them ever encounter teachers telling them explicitly what general algorithm to follow in playing normal form games: rather, they implicitly learn to generalize their decision-making ability from the examples they experience and observe (see Zizzo, 2000). So, as in Roth and Erev (1995), the basic idea is that of psychological reinforcement. However, here reinforcement does not entail direct adjustments to economic behaviour. Rather, it operates on connection weights, and only indirectly on behaviour. The difference may appear subtle but is crucial. The *behavioural learner* learns how to behave better in an economic situation, but will be completely naive as soon as it faces a new one: knowing how to perform well in a coordination game tells me nothing on how to perform optimally in, say, a Prisoner's Dilemma. Instead, given enough exposure to examples, the *neural network learner* is able to find a set of connection weights that enables it to perform optimally a majority of times even in economic situations never encountered before. In other words, it learns how to generalize its problem-solving ability.

Most evolutionary games, or learning dynamics, are based on the assumption that only one game is of interest. They then examine whether a player can converge to Nash behaviour within that game. If we consider another game we have to reset the dynamic and start again, forgetting the long process of learning to play Nash completely, or else assume that having mastered one game the player will simply pick a Nash equilibrium perfectly without any further need to learn. A neural network can be used to study learning dynamics in repeated single games as in Cho and Sargent (1996), Cho (1995) and Cho (1996), but it can also allow the player to learn a general method for solving similar games: the decision algorithm used by the player is formed out of a series of observed examples, the results being a decision-rule in which the emphasis is on learning how to play games *in general* as in Sgroi and Zizzo (2002). Hutchins and Hazelhurst (1991) suggests that neural networks twinned can even be used to provide intergenerational learning, so the experience of one generation is not

lost to the next.

III. 3 Levels of Processing Ability

Finally, a neural network learner can be further limited by placing additional constraints upon its structure, such as reducing the number of hidden layers, reducing the number of nodes at any or all layers, reducing the available training time and/or setting less restrictive limits to the errors deemed acceptable. This provides a wide range of ways to classify the neural network. This has been used in some papers to allow different specifications of neural network to approximate different levels of processing ability in economics agents. The most notable example is Rubinstein (1993) discussed in section 5. Other papers specifically aim to find the minimally complex neural network needed to solve a specific problem (see Cho and Li, 1999).

IV. LEARNABILITY AND BOUNDED RATIONALITY

There is a wide range of evidence from the experimental psychology and computer science literature to support the use of neural networks as practical models of human learning especially in the context of learning by example and dealing with problems never encountered before.

There is certainly evidence that children learn by example as in Bandura (1977), and are able to generalize from those examples (for instance, learn to talk: Plunkett and Sinha (1992)). In cognitive science, neural networks have been used to model how agents actually face pattern recognition and categorization, as in Taraban and Palacios (1994), for child development, as in Elman *et al.* (1996), for animal learning, as in Schmajuk (1997), and even arithmetic learning, as in Anderson (1998). Most importantly these studies show that what networks learn in practice is also exactly what makes neural networks useful for psychological modelling. For example, a model of arithmetic learning that would

predict the absence of mistakes is unlikely to be plausible when dealing with human subjects.

As yet, relatively little has been done within game theory to capitalize on this research. This section begins the process by examining learnability results such as Hornik *et al.* (1986) and Sontag and Sussmann (1989), and applications of those results to play by a trained neural network in new games in Sgroi and Zizzo (2002). Then in the following section we go on to examine the literature on learning to play in specific games.

IV.1 Learnability

A set of results in the algorithm complexity and computer science literature exist which focus on neural network learning. One of the most well-known results comes from Hornik *et al.* (1986). Summarizing, Hornik *et al.* (1986) find that standard feedforward networks with only a single hidden layer can approximate any continuous function uniformly on any compact set and any measurable function arbitrarily well. The main result, Theorem 2.4 from Hornik *et al.* (1986) effectively concerns the *existence* of a set of weights which allow the perfect emulation of the algorithm that the neural network is attempting to learn. There is however a major area of potential worry. The network may experience inadequate learning, so the learning dynamic will fail to reach the global error-minimizing algorithm. A *learning algorithm* takes the training sample and acts on these to produce a function $h \in H$ that, provided the sample is large enough, is with probability at least $1 - \delta$ within ε of the global error-minimizing algorithm (the algorithm which picks out the Nash equilibrium actions). The final function h produced by the neural network can be thought of as representing the entire processing of the neural network's multiple layers, taking an input vector x and producing a vector representation of a choice of strategy. Over a long enough time period we would hope that C will return a set of optimal weights which will in turn produce a function which will select the Nash strategy. This

all crucially rests on the ability of backpropagation to pick out the globally error-minimizing algorithm for finding Nash equilibria. While backpropagation is undoubtedly one of the most popular search algorithms currently used to train feedforward neural networks, it is a gradient descent algorithm and, therefore as shown in Sontag and Sussmann (1989) this approach leads only to a local minimum of the error function. In fact, while:

> "...sufficiently complex multilayer feedforward networks are capable of arbitrarily accurate approximations to arbitrary mappings...an unresolved issue is that of "learnability", that is whether there exist methods allowing the network weights corresponding to these approximations to be learned from empirical observation of such mappings." White (1992, page 160).

White (1992, Theorem 3.1) summarizes the difficulties inherent in backpropagation: it can get stuck at local minima or saddle points, can diverge, and therefore cannot be guaranteed to get close to a global minimum. The problem is exacerbated in the case of our neural network C as the space of possible weights is so large. Furthermore, Fukumizu and Amari (2000) show that local minima will always exist in problems of this type and Auer *et al.* (1996) show that the number of local minima for this class of networks can be exponentially large in the number of network parameters. The upper bound for the number of such local minima is calculable, but it is unfortunately not tight enough to lessen the problem (Sontag, 1995). In fact, as the probability of finding the absolute minimizing algorithm (the Nash algorithm) is likely to be exponentially small, the learning problem faced by C falls into a class of problems known in algorithm complexity theory as NP-hard.[2] Though gradient descent algorithms

[2]The hardness of finding Nash strategies is the subject of a small literature on the complexity of computing an automaton to play best response strategies in repeated games, see Gilboa (1988) and Ben-Porath (1990). It was this earlier literature that established the hardness of computing Nash strategies under certain conditions, though automata rather than neural networks are used to model bounded rationality. Both of these papers and subsequent work have drawn attention to the fact that we need to focus on whether best responses are simply to hard to find in certain situations, and if so we need to find an alternative.

attempt to search for the minimum of an error function, and backpropagation is no exception, given the prevalence of local minima, the neural network cannot consistently find an absolute minimum. The basins of attraction surrounding a local minimum are simply too strong for a simple gradient descent algorithm to escape.[3] Sgroi and Zizzo (2002) completes this by arguing that a neural network will therefore find a decision-making algorithm that will retain some error even at the limit, so we may have to be content with an algorithm which is effective in only a subclass of games, optimizing network parameters only in a small subset of the total space of parameters.

IV.2 Bounded Rationality

Given that backpropagation will find a local minimum, but will not readily find an absolute minimizing algorithm in polynomial time, we are left with the question, what is the best our neural network player can hope to achieve? In terms of players in a game, we have what looks like bounded-rational learning or satisficing behaviour: the player will learn until satisfied that he will choose a Nash equilibrium strategy sufficiently many times to ensure a high payoff. Sgroi and Zizzo (2002) call this a *local error-minimizing algorithm,* which can be interpreted as a case of rules of thumb that a bounded-rational agent is likely to employ in the spirit of Simon (1955) or Simon (1959). They differ from traditionally conceived rules of thumb in two ways. First, they do select the best choice in some subset of games likely to be faced by the learner. Second, they are learned *endogenously* by the learner in an attempt to maximize the probability of selecting the best outcome. The *best* outcome can be determined in terms of utility or profit maximization or a reference point, such as the Nash

[3]Other far less biologically plausible methods involving processor hungry guess and verify techniques, can produce better results. If we were to supplement the algorithm with a guessing stage, i.e. add something akin to grid search, or a subtle application of the theory of sieves, then we could hope to find the absolute minimum in polynomial time (White, 1992). However, White (1992), page 161 argues that such methods "... lay no claim to biological or cognitive plausibility", and are therefore not desirable additions to the modelling of decision-making.

equilibrium.[4]

IV.3 Learning a Method of Playing New Games

Sgroi and Zizzo (2002) presents a neural network model that plays (up to) 3×3 games, the simplest set of games in which iterated deletion of dominated strategies is anything but trivial. A neural network is constructed and trained on learning the correct best reply for a large number of test games, which corresponds to learning how to select the unique Nash strategy from the three strategies available. Having achieved a high degree of success at recognizing the correct Nash actions to play on this training sample, the constructed neural network complete with its game-solving algorithm is released to play a large number of completely new games. The results in Sgroi and Zizzo (2002) suggest a figure of around 60% success on games never encountered before based on the constructed neural network, as compared with 33% as the random success benchmark. [5] This compares well with the 59.6% experimental figure from Stahl and Wilson (1994).[6]

Solution concepts other than Nash and based on payoff dominance (similar to those methods discussed in Costa-Gomes *et al.*, 2001) get closer to explaining the simulated network's actual behaviour, and provide something close to a local-minimizing algorithm for the neural network. The neural network displays some strategic awareness, but this is not unbounded, and is decreasing in the levels of iterated deletion of dominated strategies required. The network goes for high payoff values. It takes into account potential trembles due to the temptation of the other player of deviating from Nash. It plays better in higher stakes games, particularly if there is more conflict of interests between itself and the other

[4]The finite automata literature has a different view of bounded rationality, focusing on complexity in terms of the cost of computing strategies. For example, see Rubinstein (1986) or Abreu and Rubinstein (1988).

[5]Earlier results using the same neural network model are detailed in Zizzo and Sgroi (2000) which also developed the econometric methods later used in Sgroi and Zizzo (2002).

[6]See also Stahl and Wilson (1995).

player. The trained network's behavioral heuristics carry over to a relevant degree when it faces not just new games, but new classes of games, namely games with multiple and zero pure Nash equilibria. Moreover, networks trained on different games with a unique pure Nash equilibrium are able to coordinate on the same focal solution, when encountering games with multiple equilibria.

The key conclusions from the literature on the learnability of patterns of play are that we cannot expect a neural network to achieve 100% success in new games though we can expect it to learn an algorithm capable of performing to a similar standard to observed experimental laboratory subjects. The algorithm itself is likely to based on concepts linked to iterated dominance which are strategically much easier to compute, and this will be produced endogenously during the neural network's training.

V. APPLICATIONS TO DECISION-MAKING IN ECONOMICS

This section explores the various uses within economics and especially game theory to which neural networks have been applied directly. Unlike in Sgroi and Zizzo (2002) where the aim of neural network learning was to develop a general method for playing a new game, the following papers usually focus on the neural networks ability to improve when playing a single game such as: repeated instances of the Prisoner's Dilemma in Cho and Sargent (1996), Cho (1995) and Macy (1996) and games of moral hazard in Cho and Sargent (1996), Cho (1994) and Cho (1996); heterogenous consumers in a model of monopoly in Rubinstein (1993); market entry with bounded rational firms in Leshno *et al.* (2003) and Cournot oligopoly in Barr and Saraceno (2003); and finally intergenerational learning in Hutchins and Hazelhurst (1991).

V.1 The Prisoner's Dilemma and Games of Moral Hazard

Various papers directly explore the outcome of allocating neural networks

the task of playing repeated instances of the Prisoner's Dilemma.[7]

Cho and Sargent (1996) provides a primer on the capabilities of simple perceptrons showcasing their application to various problems in economics and game theory, beginning with two-armed bandit problems and repeated play of the Prisoner's Dilemma, and eventually building a full-scale application to moral hazard problems in a model of stochastic growth.[8] Focusing on the findings linked to the Prisoner's Dilemma, Cho and Sargent (1996) and Cho (1995) show that a simple perceptron can find the optimal strictly dominant action in the Prisoner's Dilemma, since that game is solvable through linear programming methods.[9] This is in line with the finding in Sgroi and Zizzo (2002) which shows that a neural networks can learn strict dominance and variations on dominance as a solution concept which is all that is required to solve a Prisoner's Dilemma game.

Macy (1996) simulated a population playing an iterated Prisoner's Dilemma with bilateral payoffs of 3 and 1 for cooperate and defect, and unilateral payoffs of 0 and 5 for cooperate and defect (the same payoffs as in Axelrod, 1987). Each generation consisted of 50 rounds of play with random partner selection from a population of 100. The players knew the previous two moves and whether their opponents were strangers. Simulations were run from the start and from a "state of war" for 105 generations. While the main aim of the simulations in

[7]Rubinstein (1986) and Abreu and Rubinstein (1988) also offer an attempt to model bounded rational play of repeated games, especially the Prisoner's Dilemma, but instead use finite automata. Their focus is on the high computational cost of more complex strategies and so they suggest that using simple finite automata to model repeated strategies captures a player's desire to keep this cost down.

[8]Perceptrons are simpler than more general feedforward neural networks (of which they are a subset). They calculate the empirical frequency of outcomes in a given history and then classify the history according to linear functions of the calculated empirical frequency. They are often described as "classifiers" because their output is based upon whether some function of the inputs exceed a given threshold on the real number line. In this way a perceptron is capable of offering a binary yes or no decision, and is therefore especially useful for solving linear programming problems. An alternative way of visualizing feedforward neural networks is as a system of perceptrons working in parallel producing finer and finer classifications. See Cho and Sargent (1996).

[9]Cho and Sargent (1996) also shows that in the presence of imperfect monitoring (and the moral hazard problems which this implies) linear proxies can overstate the hidden variables. Cho and Sargent solved this problem only through the use of a strong law of large numbers to avoid overestimation of variable values.

Macy (1996) were to test the viability of cooperation in the repeated Prisoner's Dilemma, the paper also aimed to show the differential impact of hardwired rules (which are fixed for a given player and can only be changed with a new player in the next generation via natural selection or reproduction and were modelled using genetic algorithms) and softwired (which can be changed for the same player through social learning or reinforcement and were modelled using neural networks). Reproduction alters the frequency distribution of strategies with in the population of players, whereas reinforcement alters the probability distribution of strategies within the repertoire of each player. Hardwiring produced *"Win-Stay, Lose-Shift"* as the winning strategy with punctuated equilibria (long sequences of cooperate with sudden collapses swiftly replaced with more cooperate). Adding softwiring eliminated punctuated equilibria but retained high levels of cooperation. Increased strategic complexity reduced the prospects for cooperation under neural network learning though far less so under hardwiring since they found that latent cooperative genes could be protected from selection pressures until the rules were sufficiently widespread so they could emerge in safety. Uncertainty also hampered neural network learning, but less so hardwired genetic algorithms. Errors actually helped natural selection weed out naive cooperators. Macy also allowed direct instruction combined with imitation between players in what he describes as a process of "social influence". He found that this provided an antidote to uncertainty by increasing the probability of bilateral choices relative to unilateral ones, but produced a risk of bilateral defectors trapping each other. However, the costs of mutual defection biased the result toward coordination on both cooperate.

Macy's simulations confirm that neural network learning can produce cooperation in a repeated Prisoner's Dilemma, though uncertainty made cooperation harder. In that sense Macy (1996) supports the theoretical findings in Cho and Sargent (1996) and Cho (1995).

These papers demonstrate that when examining repeated instances of the Prisoner's Dilemma a neural network (even a simple perceptron) is quite up to the task of finding the Nash equilibria and we can even produce something

akin to a folk theorem. For more general games the results in Cho (1994) and Cho (1996) show that a more general feedforward network, along the lines of section 2, can be sufficient unless we add complexities such as discounting and uncertainty in combination.[10] The final conclusion from these papers should be that even quite simple neural networks can handle lengthy repetitions of the same well-specified and well-understood game.

V.2 Limited Consumer Processing Ability

Rubinstein (1993) uses neural networks to capture the notion of a limit to consumer processing ability by consumers in a model of monopolistic competition. In particular this paper unlike those discussed earlier allows consumers to differ in their ability to process information and so examines the impact of differential levels of bounded rationality.

The consumers wish to purchase a single indivisible good from a monopolistic firm. The seller announces a price policy (a specification of a different lottery of prices for each of the states of nature) to which he is committed. The state of nature and the price is then determined. Finally, consumers are informed about the realization of the price lottery and decide whether to accept or reject the offer. This can be seen as a slight variation on the Stackelberg leader-followers model with the monopoly seller as leader and the consumers as follower. To this Rubinstein adds imperfection in consumers' calculations and the monopoly's ability to offer not just a simple price, but several price components. The example given is for a stereo where component prices, taxes and various service fees are all offered. Rubinstein then allows consumers to differ in their ability to interpret this information. Each consumer's ability to process this information is modelled in terms of a perceptron. The neural network receives some of

[10] In a two-person repeated game Cho (1994) shows how to recover the folk theorem for a finite perceptron with a single linear classifier without discounting, and Cho (1996) shows that perceptrons with three linear classifiers can be sufficient even under imperfect monitoring (where each player is allowed to monitor the other by computing the ordinary least-square estimator of the opponent's expected payoff).

the components of the price vector as its input and produces a real number as output, with complexity measured by the number of price vectors in the domain of the neural network. Consumers are divided into two types based on the complexity of the neural networks operating their ability to compare prices. The monopoly is shown to be able to exploit this variation in type and bounded processing power to gain additional profit. Rubinstein explains that consumers with low processing ability (using simpler neural networks) are incapable of inferring the true state of nature from the observed pricing strategy of the monopolist and will often fail to identify when it is profitable to purchase the product at a high price.

While Rubinstein does not offer any direct empirical support for his model he does offer a more general conclusion which fits casual empirical findings: sellers will aim to offer complex pricing schemes precisely to exploit bounded rational agents. Whether modelled as neural networks or not, this has the seeds of a testable hypothesis, and seems very reasonable in some complex markets. What is of greatest interest is perhaps not the conclusions of the paper but rather the fact that Rubinstein chose neural networks to model bounded rational consumers. His rationale may come from the fact that, just as in Sgroi and Zizzo (2002), it is Herbert Simon to whom Rubinstein looks for a clear definition of bounded rationality as one in terms of processing ability, and it is here that neural networks stand out.

V.3 Bounded Rational Firms

Leshno *et al.* (2003) focus on a market entry game with a well defined Nash equilibrium and find that simulations involving feedforward neural networks exhibit results which are very similar to empirically validated human behaviour in similar circumstances. One of the paper's key strengths is the importance placed on validating the results produced by neural network learning in comparison with empirical data.

Their model requires that N symmetric players decide in every time period whether or not to enter the market. Before deciding in each time period they are told the capacity, c, of the market. Payoffs are given as k for staying out of the market and $k + r(c - m)$ for entry, with m as the number of entrants and r as a publicly known scaling factor. Cost of entry is zero and the motivation for entry decreases in m.

A general feedforward neural network was used, with a finite number of units in the hidden layer. Leshno *et al.* (2003) changed the number of units in the hidden layer to roughly capture the idea of differing processor power between experimental subjects, and set about training the neural networks to forecast the number of entrants, m, given c. The results indicate that the neural networks can very closely approximate the performance of the human subjects in Sundali *et al.* (1995), which in turn closely approximates the Nash equilibria of this game. The authors chose to emphasize the relative simplicity of neural networks as compared with human decision-making and therefore argue that it is all the more surprising that a simple neural network can get so close to observed human performance.

Finally, using similar simulation methods, Barr and Saraceno (2003) show that firms modelled as neural networks are able to converge to the unique Nash equilibrium of a Cournot game when facing a linear demand function with stochastic parameters, and they also estimate the optimal industry structure for different environmental variables.

To summarize, in Leshno *et al.* (2003) and Barr and Saraceno (2003) neural networks perform extremely well when facing repetitions of the same market-based game. As with Sgroi and Zizzo (2002) the results in Leshno *et al.* (2003) are shown to be similar to observed laboratory behaviour by human subjects, though both neural networks and laboratory subjects perform better under repeated instances of the same game (as in Leshno *et al.,* 2003) rather than when

facing a series of new games (as in Sgroi and Zizzo, 2002).

V.4 Intergenerational Learning with Neural Networks

A final quite different application comes from Hutchins and Hazelhurst (1991), which takes the novel approach of implementing intergenerational learning using neural networks to model society's use of culture as a means of transferring information across generations. Culture is seen as "a process that permits the learning of prior generations to have more direct effects on the learning of subsequent generations." Hence Hutchins and Hazelhurst (1991) argue that culture enables a population over many generations to be able to learn what a single individual cannot learn in a lifetime (such as a useful regularity in the environment). Parts of solutions may be learned by one generation and transmitted to the next. One means of transmission is to represent what has been learned in a form readily usable by later generations through the development certain elements of culture such as myths, tools, beliefs, practices, artifacts, understandings, practices, classifications, and crucially, language, which they group together as "artifactual media".

Hutchins and Hazelhurst (1991) demonstrate this with a set of computer simulations of feedforward neural networks (two "language" neural networks and one "task" neural network) designed to show how a population can learn the relationship between the phases of the moon to tide states. This they cite as important for an early population wishing to correctly decide whether or when to relocate near the ocean in order to benefit from the easy fishing of shellfish at low tide. They first simulate the learning of a language, then direct learning from the environment and finally mediated learning which transforms encoded symbolic descriptions gained through language into vicarious experience of the events for which they stand.

They find that the system does learn, with individuals in later generations using the same learning protocol able to perform much better at predicting

environmental regularity than those in earlier generations, concluding:

> "In later generations virtually all of the individuals are able to predict the environmental regularity almost perfectly even though the individuals in later generations have no greater innate learning abilities than those in earlier generations. Clearly this phenomenon results from the retention of "successful" knowledge in the artifactual media."

They also find that unlucky choices of artifacts to transfer information between generations can slow down learning. They conclude that their neural network model displays considerable similarities to observed patterns of cultural intergenerational learning as motivated by their tidal phase example.

Relating this back to the other papers various features remain the same, such as the neural networks good ability to improve its play in an unchanging environment (a single game), but rather than specify an arbitrarily large number of training sessions Hutchins and Hazelhurst (1991) instead attempt to capture a lifetime's worth of learning and then pass this to the next generation. Clearly the biological plausibility of neural network learning is of paramount importance for their work.

VI. CONCLUSION

Neural networks are artificial intelligence models using methods analogous to those employed by the human brain. They possess strengths and weakness, and those weaknesses can be highlighted as a benefit not a liability, since they may correspond to the natural limits of bounded rational behaviour. We may know the right answer to a problem and by suitably adding further levels of biological implausibility to a neural network model, we can force it to get things right a very high percentage of the time. However, the fact that a neural network can mimic the imperfect problem-solving ability of economic agents, and does so endogenously, is exactly what makes it so appealing to anyone wanting to model

bounded rational decision-making. In particular in well-understood problems which are often repeated a neural network can become arbitrarily good at producing optimal solutions, especially where the optimal solution can be found using linear methods or through dominance arguments. When the problem is to find a general method applicable to all manner of new problems success rates are more limited, just as we would expect with human subjects.

Various papers have used neural networks to model decision-making.[11] This paper summarizes the learnability results distributed across several papers such as Hornik *et al.* (1986) and Sontag and Sussmann (1989), and applications of those results to play by a trained neural network in new games in Sgroi and Zizzo (2002). It also reviews various direct applications, such as: repeated play of the Prisoner's Dilemma in Cho and Sargent (1996), Cho (1995) and Macy (1996) and games of moral hazard in Cho and Sargent (1996), Cho (1995) and Cho (1996); heterogenous consumers in a model of monopoly in Rubinstein (1993); bounded rational firms considering market entry in Leshno *et al.* (2003) and Cournot oligopoly in Barr and Saraceno (2003); and intergenerational learning in Hutchins and Hazelhurst (1991). What each of these papers delivers is an alternative to standard game theoretic methods of modelling human behaviour, with a direct focus on the computational limitations of economic agents and the endogenous development of bounded rationality.

REFERENCES

Abreu, D. and Rubinstein, A. (1988), "The Structure of Nash Equilibria in Repeated Games with Finite Automata", *Econometrica*, 56, 1259–1282.

Anderson, J.A. (1998), "Learning Arithmetic with a Neural Network: Seven

[11] There is also a large literature beyond the scope of this paper which details how neural networks have been used in the past in the broader scientific and psychological literature, for example, learning to see patterns in Taraban and Palacios (1994), learning to talk in Plunkett and Sinha (1992), learning arithmetic in Anderson (1998), learning by children in Elman et al. (1996) and even learning in animals in Schmajuk (1997).

Times Seven is about Fifty", in *Methods, Models and Conceptual Issues: An Invitation to Cognitive Science, Volume 4* (D. Scarnborough, and S. Sternberg, eds.), MIT Press, Cambridge, Mass. and London, 255–299.

Anthony, M. and Bartlett, P. L. (1999), *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, Cambridge.

Auer, P., Herbster, M. and Warmuth, M. K. (1996), "Exponentially Many Local Minima for Single Neurons", in *Advances in Neural Information Processing Systems 8* (D. S. Turetzky, M. C. Mozer, and M. E. Hasslemo, eds.), MIT Press, Cambridge, Mass. and London, 316–322.

Axelrod, R. (1987), "The Evolution of Strategies in the Iterated Prisoner's Dilemma", in *Genetic Algorithms and Simulated Annealing* (L. David, ed.), Pitman, London, 32–41.

Bandura, A. (1977), *Social Learning Theory*, Prentice-Hall, Englewood Cliffs.

Barr, J. and Saraceno, F. (2003), "Cournot Competition, Organization and Learning", Mimeo, Dept. of Economics, Dartmouth College, Hanover.

Ben-Porath, E. (1990), "The Complexity of Computing a Best Response Automaton in Repeated Games with Mixed Strategies", *Games and Economic Behavior*, 2, 1–12.

Cho, I.K. (1994), "Bounded Rationality, Neural Networks and Repeated Games with Discounting", *Economic Theory*, 4, 935–957.

Cho, I.K. (1995), "Perceptrons Play the Repeated Prisoner's Dilemma", *Journal of Economic Theory*, 67, 266–284.

Cho, I.K. (1996), "Perceptrons Play Repeated Games with Imperfect Monitoring", *Games and Economic Behavior*, 16, 22–53.

Cho, I.K. and Li, H. (1999), "How Complex are Networks Playing Repeated Games?", *Economic Theory*, 13, 93–123.

Cho, I.K. and Sargent, T.J. (1996) "Neural Networks for Encoding and Adapting in Dynamic Economies", in *Handbook of Computational Economics, Volume 1* (H.M. Amman, D.A. Kendrick, and J. Rust, eds.), Elsevier, North Holland, 441–470.

Costa-Gomes, M., Crawford, V. P. and Broseta, B. (2001), "Cognition and Behavior in Normal-Form Games: An Experimental Study", *Econometrica*, 69, 1193–1235.

Elman, J.L., Bates, E.A., Johnson, M. H., Karniloff-Smith, A., Parisi, D., and Plunkett, K. (1996), *Rethinking Innateness: A Connectionist Perspective on Development*, MIT Press, Cambridge, Mass.

Fukumizu, K. and S. Amari (2000), "Local Minima and Plateaus in Hierarchical Structures of Multilayer Perceptrons", *Neural Networks*, 13, 317–327.

Gilboa, I. (1988), "The Complexity of Computing Best Response Automata in Repeated Games", *Journal of Economic Theory*, 45, 342–352.

Hornik, K., Stinchombe, M. B. and White, H. (1989), "Multi-layer Feedforward Networks are Universal Approximators", *Neural Networks*, 2, 359–366.

Hutchins, E. and Hazelhurst, B. (1991), "Learning in the Cultural Process", in *Artificial Life II* (C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, eds.), Addison-Wesley, Redwood City, California, 689–705.

Leshno, M., Moller, D., and Ein-Dor, P. (2003), "Neural Nets in a Group Decision Process", *International Journal of Game Theory*, 31, 447–467.

MacLeod, P. and Plunkett, K. and Rolls, E.T. (1998), *Introduction to Connectionist Modelling of Cognitive Processes*, Oxford University Press, Oxford.

Macy, M. (1996), "Natural Selection and Social Learning in Prisoner's Dilemma: Co-adaptation with Genetic Algorithms and Artificial Neural Networks", *Sociological Methods and Research*, 25, 103–137.

Plunkett, K. and Sinha, C. (1992), "Connectionism and Development Theory", British Journal of Developmental Psychology, 10, 209–254.

Roth, A.E. and Erev, I. (1995), "Learning in Extensive-Form Games: Experimental Data and Simple Dynamic Models in the Intermediate Term", *Games and Economic Behavior*, 8, 164–212.

Roth, A.E. and Erev, I. (1998), "Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria", *American Economic Review*, 88, 848–881.

Rubinstein, A. (1986), "Finite Automata Play the Repeated Prisoner's Dilemma", *Journal of Economic Theory*, 39, 83–96.

Rubinstein, A. (1993), "On Price Recognition and Computational Complexity in a Monopolistic Model", *Journal of Political Economy*, 101, 473–484.

Rumelhart, D. E., Hinton, R. J. and Williams, R. J. (1986), "Learning Representations by Back-Propagating Error", *Nature*, 323, 533–536.

Sgroi, D. and Zizzo, D.J. (2002), "Strategy Learning in 3×3 Games with Neural Networks", DAE Working Paper No. 0207, Department of Applied Economics, University of Cambridge.

Schmajuk, N.A., (1997), *Animal Learning and Cognition: A Neural Network Approach*, Cambridge University Press, Cambridge.

Simon, H. (1955), "A Behavioral Model of Rational Choice", *Quarterly Journal of Economics*, 69, 99–118.

Simon, H. (1959), "Theories of Decision-Making in Economics and Behavioral Science", *American Economic Review*, 49, 253–283.

Sontag, E. D. (1995), "Critical Points for Least-Squares Problems Involving Certain Analytic Functions with Applications to Sigmoidal Nets", *Advances in Computational Mathematics*, 5, 245–268.

Sontag, E. D. and Sussmann, H. J. (1989), "Backpropagation Can Give Rise to Spurious Local Minima Even for Networks with No Hidden Layers", *Complex Systems*, 3, 91–106.

Sundali, J.A., Rapoport, A. and Seale, D.A. (1995), "Coordination in Market Entry Games with Symmetric Players", *Organizational Behavior and Human Decision Processes*, 64, 203–218.

Stahl, D. O. and Wilson, P. W. (1994), "Experimental Evidence on Players' Models of Other Players", *Journal of Economic Behavior and Organization*, 25, 309–327.

Stahl, D. O. and Wilson, P. W. (1995), "On Players' Models of Other Players: Theory and Experimental Evidence", *Games and Economic Behavior*, 10, 218–254.

Taraban, R. and Palacios, J.M., (1994), "Exemplar Models and Weighted Cue Models in Catagory Learning", in *Categorization by Humans and Machines* (G.V. Nakamura, R. Taraban, and D.L. Medin, eds.), Academic Press, San Diego, 91–127.

White, H. (1992), *Artificial Neural Networks: Approximation and Learning Theory*, Blackwell, Cambridge and Oxford.

Zizzo, D J. (2000), "Implicit Learning of (Boundedly) Rational Behavior", *Behavioral and Brain Sciences*, 23, 700–701.

Zizzo, D.J. (2002), "Neurobiological Measurements of Cardinal Utility: Hedonimeters or Learning Algorithms?", *Social Choice and Welfare,* 19, 477–488

Zizzo, D. J. and Sgroi, D. (2000), "Bounded-Rational Behavior by Neural Networks in Normal Form Games", D.P. 2000-W30. Nuffield College, University of Oxford.