

# Neural Networks and Bounded Rationality

Daniel Sgroi<sup>\*1</sup> and Daniel J. Zizzo<sup>#</sup>

<sup>\*</sup>Faculty of Economics and Churchill College, University of Cambridge, U.K.

<sup>#</sup>School of Economics, University of East Anglia, U.K.

## Abstract

Traditionally the emphasis in neural network research has been on improving their performance as a means of pattern recognition. Here we take an alternative approach and explore the remarkable similarity between the under-performance of neural networks trained to behave optimally in economic situations and observed human performance in the laboratory under similar circumstances. In particular, we show that neural networks are consistent with observed laboratory play in two very important senses. Firstly, they select a rule for behavior which appears very similar to that used by laboratory subjects. Secondly, using this rule they perform optimally only approximately 60% of the time.

*Keywords:* neural networks, game theory, bounded rationality, learning.

PACS codes: 02.50.Le; 02.70.c; 84.35.i

---

<sup>1</sup> Corresponding author: Faculty of Economics, University of Cambridge, Cambridge CB3 9DD, United Kingdom. E-mail: daniel.sgroi@econ.cam.ac.uk. Phone: +44 1223 335244. Fax: +44 1223 335299.

## 1. Introduction

We argue that simulated neural networks can be used to model the real-world behavior of laboratory subjects, and in particular, that they can approximate the observed bounded rationality that human subjects appear to demonstrate. We believe that neural networks will not only allow us to model human behavior, but will more importantly provide insight into why and how subjects consistently perform below the expectations of social scientists who base their predictions on rationality. The theoretical tools we use to capture the notion of rationality is Nash equilibrium and Game Theory.

Game theory is the study of strategic interaction between agents. Perhaps the most common rationality assumption in game theory is that agents play the Nash equilibrium strategy [1], which requires that each player's action is an optimal strategy given that the other player also chooses optimally. For example, if a motorist drives on the right-hand side of the road, then a motorist driving in the opposite direction should optimally drive on the left. Given that the second motorist drives on the left, the first should be content to stick to the right hand side of the road. Most Nash equilibria are unfortunately a little harder to find, but similarly have the in-built stability property that players would do badly to shift to an alternative strategy given that everyone else is playing their optimal strategy. This *should* provide a good predictive tool for assessing likely behavior when two competitors meet. However, recent research indicates that in the laboratory, people often fail to play according to Nash equilibria [2,3]. So how and why does real-world behavior fail to live up to the widely accepted standard set by Nash play?

One way to answer this is to build of model of human behavior to attempt to determine exactly what is going on. In the realm of artificial intelligence there are several forms of machine learning which might be employed by a scientist seeking to approximate

likely patterns of human behavior, and we have opted for a neural network. This choice is in part justified by the large body of related literature in cognitive science and engineering, and in part by the consistency between our results and those seen in experimental testing on human subjects [3,4,5]. Neural networks are trained (just as human children are taught), they learn primarily through example (as is often the case with humans), and they create general pattern-recognizing algorithms, learning general rules of optimal behavior, which look very close to theorized optimal rules of thumb envisaged in economics and the social sciences which go back to Simon [6,7]. Neural networks do have serious limitations, but rather than a focus on how to improve on neural network learning, it is the limitations of neural network learning and the similarity between those limitations and those seen in laboratory subjects that provides the focus for our work. For more on the use of neural networks as models of bounded-rational behavior in games see [8] and the further references therein.

To motivate this, consider how an agent would face the following challenge. She believes she is a fair player of a board game, and has to decide what odds she places on her chances of winning versus a beginner. How would she change her odds of winning if she were told that her opponent rarely plays any board games at all? How about if her opponent is a chess Grand Master, and known to be a very strong player of many other similar board games? What we need is a model of how good players can be if they have experience of similar, though not identical games. More generally, we want to model real-world agents who have faced similar, though not identical, situations in the past. A good way to go about this is to focus on learning general rules of behavior by example. By training a neural network according to the Nash algorithm, and then allowing our simulated neural network to try playing new games, using whatever rules it has learned, we can approximate a human adult who, having learned rules of conduct during childhood, must venture forth into the

economy. What we find is that a neural network trained in this way seems to make errors that are remarkably similar to those errors made by humans in the laboratory, and has similar success rates. A neural network should then provide a good tool for determining how and why humans routinely fail to live up to the standards of optimal decision-making.

What we find is that Nash equilibrium is just too complex a concept for a neural network to use in general in new environments. It is not difficult for the network to find Nash equilibria in specific games, but what is difficult is to learn to employ Nash as a general algorithm. We first describe the theoretical limitations to neural network learning, and then go on to show that these limitations are precisely the sort of limitations we want to see when modelling bounded rationality in human subjects, since they evolve endogenously and reflect a method of learning by example that is very loosely biological plausible [9].

## **2. The Theory**

A neural network typically learns by exposure to a series of examples or *training set*, and adjusts 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 facing problems never encountered before by learning rules of play. To provide an intuitive summary: a neural network is continuously shown and re-shown a set of games until it recognizes the optimal way to play these games via back-propagation. Recognition comes in terms of a simple measure of mean squared error. This form of learning accords with an intuitive notion of a teacher continuously correcting the behavior of a student until behavior 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 to attempt to select the Nash equilibria in games which it has seen before.

Consider a feed-forward 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 intermediate layers of further computation between input and output. Formally, let  $R$  be the set of real numbers and define the neural network as  $C = \langle \Omega, X, Y, F \rangle$  where  $\Omega$  is a set of states,  $X \subseteq R^n$  is a set of inputs,  $Y$  is a set of outputs and  $F : \Omega \times X \rightarrow Y$  is a parameterized function. For any  $\omega$  the function represented by state  $\omega$  is  $h_\omega : X \rightarrow Y$  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  $\omega$  it computes the function  $h_\omega$ , *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 for which we use back-propagation. Rumelhart *et al.* [10] is the definitive reference for the back-propagation method. More detail can also be found in Sgroi and Zizzo [11].

The situation faced by the neural network is well modeled 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 equilibrium. Actions are given by  $a_i \in A_i$ . Feasible action combinations are given by  $A = A_1 \times A_2 \times \dots \times A_N$ . Payoffs for player  $i$  are given by a von Neumann-Morgenstern [12] bounded utility function  $u_i : A_i \rightarrow R$ . We normalize payoffs to be drawn from a uniform distribution with support  $[0,1]$  and then revealed to the players before they select an action. Figure 1 provides a stylized view of a neural network in this context.

**[“Figure 1: A stylized neural network” around here]**

Hornik *et al.* [13] found that standard feed-forward networks can approximate any continuous function uniformly on any compact set and any measurable function arbitrarily well. Their main result 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. However, 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  $h_\omega \in H_C$  that is with probability of at least  $1-\delta$  within  $\varepsilon$  of the global error-minimizing algorithm (the algorithm which picks out the Nash equilibrium actions) provided the sample is large enough. The final function  $h_\omega$  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 the network will return a set of weights which will produce a function, which attempts to select the Nash strategy.

This all crucially rests on the ability of back-propagation to pick out the globally error-minimizing algorithm for finding Nash equilibria. Back-propagation is a gradient descent algorithm and will therefore lead to a local minimum of the error function [14]. The learning process can get stuck at local minima or saddle points or can diverge, and so need not get close to a global minimum. The problem is exacerbated since: the space of possible weights is large; local minima will always exist in problems of this type [15]; and the number of local minima for this class of networks can be exponentially large in the number of network parameters [16]. The upper bound for the number of such local minima is calculable, but it is unfortunately not tight enough to lessen the problem [17]. Therefore, as the probability of finding the absolute minimizing algorithm (the Nash algorithm) is likely to be exponentially small, the learning problem is intractable (technically, NP-hard). In

response, a neural network will find a decision-making algorithm that will retain error even at the limit, leaving an algorithm which is effective only in a subclass of games [11]. This imperfect game-playing algorithm can be loosely described as a local-minimizing algorithm or LMA.

### 3. Training and Testing

We will now focus on the results of an extensive programme of training and testing. The training set consisted of 2000 games with unique pure Nash equilibria, and the precise method of training and convergence are detailed in the Appendix. We found that the network trained to a convergence level of  $\gamma = 0.1$  played the Nash equilibria of 60.03% of the games to an error tolerance of 0.05. This is consistent with the 59.6% average success rate of human subjects newly facing 3×3 games in [4]. With an error tolerance of 0.25 and 0.5, the correct answers increased to 73.47% and 80%, respectively. Further training improves the performance on exactness (the 0.02-converged network plays the Nash equilibria of a mean 67% of the games) but *not* on ‘rough correctness’ (the 20% result appears robust). This suggests and further training of the network confirms that there is an upper bound on the performance of the network.

The network did learn to generalize from the examples and to play Nash strategies at a success rate that is significantly above chance, but significantly below 100%. While the network has been trained to recognize Nash equilibria we know that the process of calculating a Nash equilibrium is a hard one, so what simpler algorithms might be capable of describing what the network does?

For testing we used the 30 networks trained with the 30 different random seeds but with learning rate 0.5 and momentum 0 (our results are robust to alternatives). Using these

30 networks, we tested the average performance of the various algorithms on the same testing set of 2000 new games with unique pure Nash equilibria considered previously. We considered the following alternative solution algorithms in turn: (1) minmax; (2) rationalizability; (3) 0-level strict dominance (0SD); (4) 1-level strict dominance (1SD); (5) pure sum of payoff dominance (L1); (6) best response to pure sum of payoff dominance (L2); (7) maximum payoff dominance (MPD); (8) nearest neighbor (NNG). Further information about these algorithms can be found in the Appendix, but some explanations follow.

Minmax chooses the action that gives the player the maximum guaranteed payoff no matter what the other player does. 0SD chooses the strictly dominant strategy, if available; dominated strategies are those that always give a lower payoff than another strategy, no matter what the other player does. 1SD iterates the process by choosing the dominant strategy assuming that the other player is a 0SD type: after having removed the actions that are strictly dominated for the other player, the simulated player chooses the dominant action among the remaining possible outcomes. In the context of a  $3 \times 3$  game, rationalizability chooses the strictly dominant strategy assuming that the other player is 1SD. MPD corresponds to going for the highest conceivable payoff for itself; L1 to choosing the best action against a uniformly randomizing opponent; L2 to choosing the best action against a L1 player. NNG responds to new situations by comparing them to the nearest example encountered in the past, behaving accordingly.

**["Table 1: Answerable games and relationship to Nash" around here]**

We define a game as answerable by an algorithm if a unique solution exists. Sgroi and Zizzo [11] provides supporting analysis for games with multiple equilibria. Table 1



lists the percentage of answerable games out of 2000 according to each algorithm, averaged out across the 30 neural networks trained with different random seeds, learning rate 0.5 and momentum 0. The central column considers the cases where these unique solutions coincide with the pure Nash equilibrium of the game. The right column adjusts this Nash predictive success of the non-Nash algorithm by assuming that the agent randomizes over all the choices admissible according to the non-Nash algorithm.

**[“Table 2: Predictive power of algorithms” around here]**

Table 2 shows the degree to which different algorithms are successful in describing the behaviour of the trained network under different convergence rates  $\gamma$ . We find that our set of candidate LMAs typically can do better than an agent employing no rationality who simply plays randomly across all choices and games. More strategically sophisticated LMAs can do better than less strategically sophisticated ones, but at the same time, although the network has been trained on Nash, Nash is not the best performing algorithm. L2, a simple algorithm based on payoff dominance, is our best approximation for the behavior of the neural network, predicting the simulated network's behavior exactly 76.44% over the full testing set. L2 is a decision-rule that selects the best action against a player who is in turn choosing the best action against a uniformly randomizing opponent. Together with L1, which also does not perform badly in our simulations, L2 has been found to be a good predictor in lab tests using human subjects [3]. As Table 2 shows, L2 has the virtue that, while being simpler than Nash, its predictions do coincide with Nash predictions more than any other non Nash algorithm. The fact that L2 outperforms less strategic algorithms suggests that the simulated network behaves as if capable of some strategic thinking.

#### 4. Discussion

We have attempted to analyse the nature of the errors made by neural networks in an attempt to draw conclusions about observed human-error made in laboratory conditions. Several important laboratory experiments have highlighted the type of errors made by human subjects and the success rates which they obtain. We find that a very simple specification of neural network can easily duplicate both the observed error-rates and provide insight into how and why human subjects continue to make errors which are ruled out by traditional rationality assumptions in the social sciences. As we more understand the methods used by biologically plausible neural networks, so we may better hope to understand the errors made by human subjects in the laboratory and competitors in the real world. During the learning process both humans and neural networks make progress, and both achieve significantly better than chance rates of success. The major research question to be asked is why both neural networks and humans seem content to hit simple rules which offer success around 60% of the time, and why concepts like payoff maximisation seem more important than overall optimisation. We might also wish to understand why some humans get closer to optimality than others, and how we might model this by using different specifications of neural networks. Much work is still to be done, but by borrowing a well-understood model from the physical sciences the social sciences are already beginning to understand better how people behave in the real-world, a perfect example of cross-fertilisation between disciplines.

- [1] J.F. Nash, Proceedings of the National Academy of Sciences 36 (1950) 48-49.
- [2] A.E. Roth and I. Erev, American Economic Review 88 (1998) 848-881.
- [3] M. Costa-Gomes, V.P. Crawford, B. Broseta, Econometrica 69 (2001) 1193-1235.
- [4] D.O. Stahl and P.W. Wilson, Journal of Economic Behavior and Organization 25 (1994) 309-327.

- [5] D.O. Stahl P.W. Wilson, *Games and Economic Behavior* 10 (1995) 218-254.
- [6] H.A. Simon, *Quarterly Journal of Economics* 69 (1959) 99-118.
- [7] H.A. Simon, *American Economic Review* 49 (1959) 253-283.
- [8] D. Sgroi in D.J. Zizzo (ed) *Transfer of Knowledge in Economic Decision Making*, Palgrave Macmillan, London, U.K. (2005) 128-147.
- [9] D.J. Zizzo, *Social Choice and Welfare* 19 (2002) 477-488.
- [10] D.E. Rumelhart, R.J. Hinton, R.J. Williams, *Nature* 323 (1986) 533-536.
- [11] D. Sgroi and D.J. Zizzo, *Cambridge Working Paper in Economics* 0207, Cambridge University, U.K. (2002).
- [12] J.V. Neumann and O. Morgenstern, *The Theory of Games and Economic Behavior*, Princeton University Press (1944).
- [13] K. Hornik, M.B. Stinchcombe, H. White, *Neural Networks* 2 (1989) 359-366.
- [14] E.D. Sontag and H.J. Sussmann, *Complex Systems* 3 (1989) 91-106.
- [15] K. Fukumizu and S. Amari, *Neural Networks* 13 (2000) 317-327.
- [16] P. Auer, M. Herbster, M.K. Warmuth in D.S. Turetzky, M.C. Mozer, M.E. Hasslemo (eds), *Advances in Neural Information Processing Systems* 8, MIT Press, Cambridge, Mass. and London (1996) 316-322.
- [17] E.D. Sontag, *Advances in Computational Mathematics* 5 (1995) 245-268.
- [18] R.A. Reilly, *Connectionist exploration of the computational implications of embodiment. Proceedings of the Swedish Conference on Connectionism*. Hillsdale: Lawrence Erlbaum (1995).

### **Acknowledgements**

Daniel Sgroi is grateful to AEA Technology plc for financial support and both authors would like to thank Michael Bacharach, Vincent Crawford, Huw Dixon, Glenn Ellison, Jim Engle-Warnick, Meg Meyer, Michael Rasmussen, Andrew Temple and seminar participants in Cambridge, Copenhagen and Oxford.

## Appendix

This appendix provides supporting material. Further details on the testing and development of a simulated neural network can also be found in SgROI and Zizzo [11], including findings related to games with multiple equilibria.

### Training

Training was random with replacement subject to the unique Nash equilibrium condition, and continued until the error  $\varepsilon$  converged below convergence level  $\gamma \in \{0.1, 0.05, 0.02\}$ . Three values of  $\gamma$  were used for the sake of performance comparison, and convergence was checked every 100 games. The initial connection weights and order of presentation of the games were generated according to a random seed given at the start of training. To check the robustness of the analysis, the neural network was trained 360 times, that is once for every combination of: 3 learning rates  $\eta \in \{0.1, 0.3, 0.5\}$  that give the speed of the adjustment of the connection weights when the network fails to play following Nash equilibrium behavior; 4 momentum rates  $\mu \in \{0, 0.3, 0.6 \text{ and } 0.9\}$  which introduce auto-correlation in the adjustments of the connection weights when successive examples are presented; and 30 random seeds. Convergence was always obtained, at least at the 0.1 level, except for a very high momentum rate. More details on convergence can be found in the next section. The simulated network was tested on a new set of 2000 games with unique Nash equilibria which it had never played before. We considered an output value to be correct when it is within some range from the exact correct value. If both outputs are within the admissible range, then we can consider the answer “correct” [18].

**[“Table 3: Percentage of Correct Answers” around here]**

Table 3 displays the average performance classified by  $\gamma$ ,  $\eta$  and  $\mu$ . The numbers given under ‘At least 1 Correct Output’ are the % of cases in which at least 1 of the two output nodes is correct. The numbers given under ‘Correct Answer Given’ are the % of cases in which both output nodes are correct. The table shows that the network trained until  $\gamma = 0.1$  played the Nash equilibria of 60.03% of the games within the 0.05 range.

### Convergence

Table 4 lists the number of networks (of 30) trained with different random seeds that achieved various levels of convergence for different learning and momentum rates. Convergence was achieved at all but a combination of high  $\eta$  and  $\mu$  ( $\eta = 0.5$  and  $\mu = 0.9$ ). A high momentum rate was particularly detrimental. A low learning rate ( $\eta = 0.1$ ) slowed learning, particularly with  $\mu \leq 0.3$ , but did not prevent convergence at  $\gamma = 0.1$  or better. A maximum of 600,000 rounds was used to try to achieve convergence, though in practice,  $\gamma = 0.1$  convergence was usually reached in about 100,000 to 200,000 rounds.

**[“Table 4: Number of neural networks (of 30) achieving convergence” around here]**

### Solution Algorithms

This section provides some detail about alternative solution algorithms. Consider the game  $G$  and the trained neural network player  $C$ . Index the neural network by  $i$  and the other player by  $j$ . The neural network’s minmax value (or reservation utility) is defined as  $r_i = \min_{a_j} [\max_{a_i} u_i(a_i, a_j)]$ . The payoff  $r_i$  is literally the lowest payoff player  $j$  can hold the network to by any choice of  $a_j \in A_j$ , provided that the network correctly foresees  $a_j$  and plays a best response to it. Minmax therefore requires a particular brand of pessimism to have been developed during the network’s training on Nash equilibria.

Rationalizability is widely considered a weaker solution concept than Nash equilibrium, in the sense that every Nash equilibrium is rationalizable, though every rationalizable equilibrium need not be a Nash equilibrium. A dominated strategy for  $i$  produces a lower payoff against all possible strategies for  $j$ . Rationalizable sets and the set which survives the iterated deletion of strictly dominated strategies are equivalent in two player games: call this set  $S_i^n$  for player  $i$  after  $n$  stages of deletion.  $S_i^n$  is the set of player  $i$ 's strategies that are not strictly dominated when players  $j \neq i$  are constrained to play strategies in  $S_j^{n-1}$ . So the network will delete strictly dominated strategies and will assume other players will do the same, and this may reduce the available set of strategies to be less than the total set of actions for  $i$ , resulting in a subset  $S_i^n \subseteq A_i$ . Since we are dealing with only three possible strategies in  $G$ , the subset can be adequately described as  $S_i^2 \subseteq A_i$  with player  $j$  restricted to  $S_j^1 \subseteq A_j$ .

The algorithm OSD checks whether all payoffs for the neural network player  $i$  from playing an action are strictly higher than those from the other actions, so no restriction is applied to the action of player  $j \neq i$ , and player  $i$ 's actions are chosen from  $S_i^0 \subseteq A_i$ . 1SD allows a single level of iteration in the deletion of strictly dominated strategies: the row player thinks that the column player follows OSD, so chooses from  $S_j^0 \subseteq A_j$ , and player  $i$ 's action set is restricted to  $S_i^1 \subseteq A_i$ . Both of these algorithms can be viewed as weakened, less computationally demanding versions of iterated deletion.

L1 and MPD are different ways of formalizing the idea that the agent might try to go for the largest payoffs, almost independently of strategic considerations. Following MPD, the neural network player  $i$  (the row player in a normal form game) simply learns to spot the highest conceivable payoff for itself, and picks the corresponding row, hoping that  $j$  will pick the corresponding column to ensure the highest payoff for  $i$ . In the case of L1, an

action  $a_i = a_{L1}$  is chosen by the row player according to  $a_{L1} = \arg \max_{a_i \in A_i} \{ a_i \mid a_j = M_j \}$ , where  $M_j$  is defined as a perfect mix over all available strategies in  $A_j$ . Put simply,  $a_{L1}$  is the strategy which picks a row by calculating the payoff from each row, based on the assumption that player  $j$  will randomly select each column with probability  $1/3$ , and then chooses the row with the highest payoff calculated in this way. This corresponds to Stahl and Wilson's definition of 'level 1 thinking' [5]. A player that assumes that his opponent is a 'level 1 thinker' will best-respond to L1 play, and hence engage in what we can label L2 (for 'level 2') play [3,4].

Finally, the NNG algorithm is based on the idea that agents respond to new situations by comparing them to the nearest example encountered in the past, and behave accordingly. The NNG algorithm examines each new game from the set of all possible games, and attempts to find the specific game from the training set (which proxies for memory) with the highest level of 'similarity'. We compute similarity by summing the square differences between each payoff value of the new game and each corresponding payoff value of each game of the training set. The higher the sum of squares, the lower the similarity between the new game and each game in the training set. The NNG algorithm looks for the game with the lowest sum of squares, the *nearest neighbor*, and chooses the unique pure Nash equilibrium corresponding to this game. Since, by construction, all games in the training set have a unique pure strategy Nash equilibrium, we are virtually guaranteed to find a NNG solution for all games in the testing set. The only potential (but unlikely) exception is if the nearest neighbor is not unique, because of two (or more) games having exactly the same sum of squares. The exception never held with our game sample. A unique solution, or indeed any solution, may not exist with algorithms such as rationalizability, OSD and 1SD. A unique solution may occasionally not exist with other algorithms, such as MPD, because of their reliance on strict relationships between payoff values.

Figure for “Neural Networks and Bounded Rationality” by D. Sgroi and D. J. Zizzo

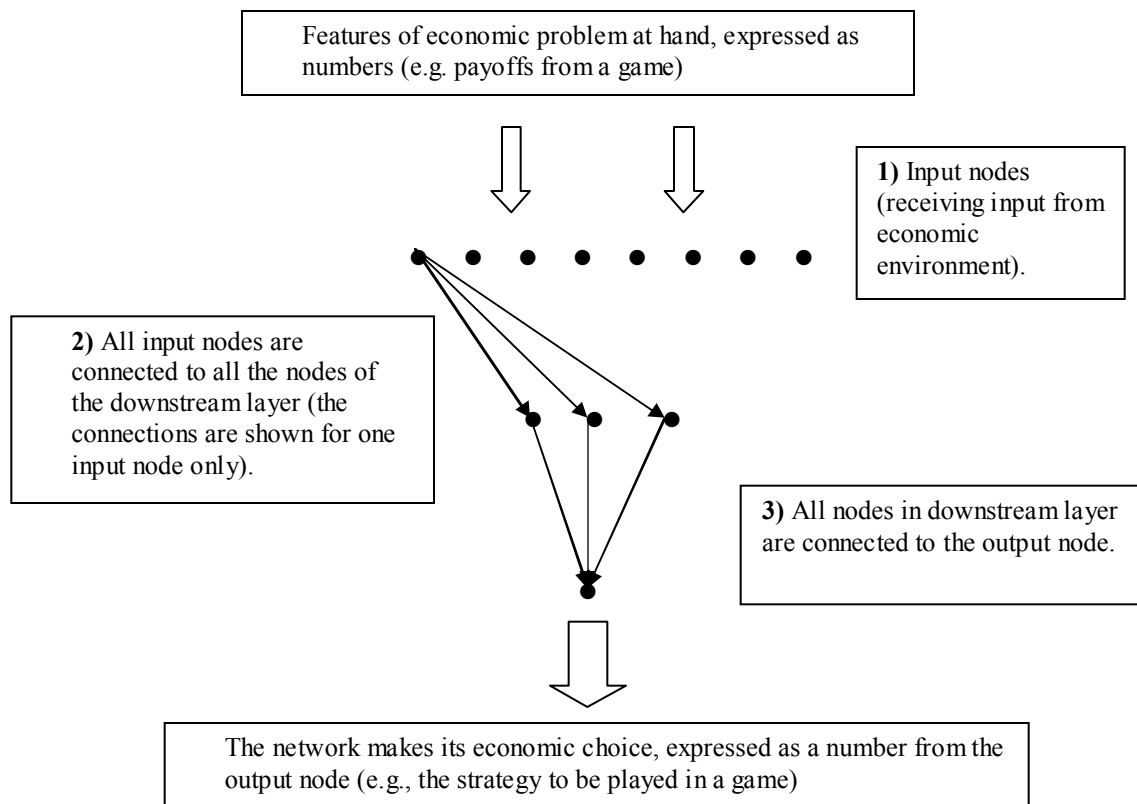


Figure 1: A stylized neural network



Tables for “Neural Networks and Bounded Rationality” by D. Sgroi and D. J. Zizzo

<b>Non Nash Algorithm</b>	<b>Answerable Games (%)</b>	<b>Uniquely predicted Nash equilibria (%)</b>	<b>Expected performance in predicting Nash (%)</b>
<b>0SD</b>	<b>18.3</b>	<b>18.3</b>	<b>46.97</b>
<b>1SD</b>	<b>39.75</b>	<b>39.75</b>	<b>62.19</b>
<b>Rationalizability</b>	<b>59.35</b>	<b>59.35</b>	<b>74.78</b>
<b>L1</b>	<b>100</b>	<b>67</b>	<b>67</b>
<b>L2</b>	<b>99.75</b>	<b>88.4</b>	<b>88.48</b>
<b>MPD</b>	<b>99.6</b>	<b>62.1</b>	<b>62.23</b>
<b>Minmax</b>	<b>99.75</b>	<b>58.55</b>	<b>58.63</b>
<b>Nearest Neighbor</b>	<b>100</b>	<b>62.8</b>	<b>62.8</b>

Table 1: Answerable games and relationship to Nash

<b>Algorithm</b>	<b>% of Correct Answers</b>			<b>% of Correct Answers</b>			<b>Expected Performance</b>		
	<b>Over Answerable Games</b>			<b>Over Full Testing Set</b>			<b>Over Full Testing Set (%)</b>		
	$\gamma=0.5$	$\gamma=0.25$	$\gamma=0.05$	$\gamma=0.5$	$\gamma=0.25$	$\gamma=0.05$	$\gamma=0.5$	$\gamma=0.25$	$\gamma=0.05$
<b>Nash</b>	<b>80.38</b>	<b>75.7</b>	<b>66.53</b>	<b>80.38</b>	<b>75.7</b>	<b>66.53</b>	<b>80.38</b>	<b>75.7</b>	<b>66.53</b>
<b>0SD</b>	<b>90.83</b>	<b>87.79</b>	<b>80.98</b>	<b>16.62</b>	<b>16.06</b>	<b>14.82</b>	<b>43.91</b>	<b>43.35</b>	<b>42.12</b>
<b>1SD</b>	<b>87.63</b>	<b>84</b>	<b>76.25</b>	<b>34.83</b>	<b>33.39</b>	<b>30.31</b>	<b>54.51</b>	<b>53.15</b>	<b>50.19</b>
<b>Rationalizability</b>	<b>86.44</b>	<b>82.57</b>	<b>74.36</b>	<b>51.3</b>	<b>49.01</b>	<b>44.13</b>	<b>64.16</b>	<b>61.94</b>	<b>57.18</b>
<b>L1</b>	<b>67.71</b>	<b>63.48</b>	<b>55.97</b>	<b>67.71</b>	<b>63.48</b>	<b>55.97</b>	<b>67.71</b>	<b>63.48</b>	<b>55.97</b>
<b>L2</b>	<b>85.11</b>	<b>82.23</b>	<b>76.54</b>	<b>84.9</b>	<b>82.03</b>	<b>76.35</b>	<b>84.98</b>	<b>82.11</b>	<b>76.44</b>
<b>MPD</b>	<b>61.08</b>	<b>57.02</b>	<b>49.95</b>	<b>60.84</b>	<b>56.79</b>	<b>49.75</b>	<b>60.84</b>	<b>56.79</b>	<b>49.76</b>
<b>Minmax</b>	<b>57.75</b>	<b>53.83</b>	<b>46.89</b>	<b>57.6</b>	<b>53.7</b>	<b>46.77</b>	<b>57.67</b>	<b>53.77</b>	<b>46.85</b>
<b>Nearest Neighbor</b>	<b>62.97</b>	<b>58.78</b>	<b>51.48</b>	<b>62.97</b>	<b>58.78</b>	<b>51.48</b>	<b>62.97</b>	<b>58.78</b>	<b>51.48</b>

Table 2: Predictive power of algorithms

Convergence Level $\gamma$	At Least 1 Correct Output			Correct Answer Given		
	Error tolerance criterion			Error tolerance criterion		
	0.05	0.25	0.5	0.05	0.25	0.5
0.1	85.12	91.76	94.31	60.03	73.47	80
0.05	87.26	92.24	94.31	64.12	74.75	80.09
0.02	88.52	92.51	94.25	66.66	75.47	79.96

Learning Rate $\eta$	At Least 1 Correct Output			Correct Answer Given		
	Error tolerance criterion			Error tolerance criterion		
	0.05	0.25	0.5	0.05	0.25	0.5
0.1	86.88	92.3	94.42	63	74.48	80.22
0.3	86.75	92.06	94.25	63.3	74.42	79.89
0.5	86.81	92.04	94.2	63.66	74.54	79.94

Momentum Rate $\mu$	At Least 1 Correct Output			Correct Answer Given		
	Error tolerance criterion			Error tolerance criterion		
	0.05	0.25	0.5	0.05	0.25	0.5
0	86.87	92.36	94.5	62.86	74.63	80.47
0.3	86.77	92.27	94.42	62.89	74.49	80.22
0.6	86.91	92.09	94.23	63.73	74.53	79.9
0.9	86.6	91.52	93.8	64.05	74.05	79.04

Table 3: Percentage of Correct Answers

Convergence		M o m e n t u m R a t e $\mu$			
Level $\gamma = 0.1$		0	0.3	0.6	0.9
Learning	0.1	30	30	30	30
Rate $\eta$	0.3	30	30	30	25
	0.5	30	30	30	0

Convergence		M o m e n t u m R a t e $\mu$			
Level $\gamma = 0.05$		0	0.3	0.6	0.9
Learning	0.1	27	30	30	30
Rate $\eta$	0.3	30	30	30	11
	0.5	30	30	30	0

Convergence		M o m e n t u m R a t e $\mu$			
Level $\gamma = 0.02$		0	0.3	0.6	0.9
Learning	0.1	0	1	30	30
Rate $\eta$	0.3	30	30	30	0
	0.5	30	30	30	0

Table 4: Number of neural networks (of 30) achieving convergence