

Neural Network-Based Strategies for the Iterated Prisoner's Dilemma

ICS 276A: Neural Networks

Winter 2002

Joshua O'Madadhain (Madden)

July 10, 2004

1 Introduction

1.1 Overview

The purpose of this paper is to report the results of my investigations into the development of adaptive neural network-based strategies for the Iterated Prisoner's Dilemma (IPD). My objectives were to develop a method by which an artificial neural network (ANN) could learn to play IPD, to define a template for an IPD-playing ANN, and to empirically determine some appropriate values for the parameters of such templates. Two specific things that I investigated were the effects of an explicit record of previous moves, and of increasing network complexity.

1.2 The Iterated Prisoner's Dilemma

The **Prisoner's Dilemma** is a classic problem in game theory, discovered in 1950 by Melvin Dresher and Merrill Flood, whose purpose is, among other things, to describe a situation in which rational behavior (defined as that behavior which will maximize one's expected benefit) can be, in a sense, self-undermining. It has become a popular problem because it is seen as fundamentally similar to real-world problems in sociology and politics.

The Prisoner's Dilemma can be described as follows: suppose that two individuals, A and B , have been arrested in connection with a crime. Each is put in a separate interview room and told that a deal may be available to them depending on whether each provides testimony against the other (defects) or keeps silent (cooperates). Each individual is assumed to want to do as well as possible for themselves *without regard to the welfare of the other player*.

This situation is formally specified by the matrix in Figure 1, subject to the following definitions and restrictions:

- T : **temptation** for unilateral defection
- R : **reward** for mutual cooperation
- P : **punishment** for mutual defection
- S : **sacrifice** for unilateral cooperation

	<i>B</i> cooperates	<i>B</i> defects
<i>A</i> cooperates	(R, R)	(S, T)
<i>A</i> defects	(T, S)	(P, P)

Figure 1: Payoff matrix for the Prisoner's Dilemma

Figure 2: MDP diagram for the iterated Prisoner's Dilemma

- $T > R > P > S$
- $2R > T + S$

The purpose of the condition $2R > T + S$ is to ensure that a consistent payoff of R is better than consistently alternating between a payoff of T and a payoff of S .

It is easy to show that it is always to a player's advantage to defect, since the payoff will be higher for that choice regardless of the other player's actions. However, it is also clear that mutual cooperation is a better outcome for each player than is mutual defection.

The Iterated Prisoner's Dilemma extends the Prisoner's Dilemma by causing the game to be played multiple times, so that each player has the opportunity to adapt his behavior based on his observations of the other player's behavior. This allows, for example, each player to discover whether the other is willing to engage in consistent mutual cooperation. As with the original (one-stage) Prisoner's Dilemma, each player is assumed to be trying to maximize their total payoff over all games, without concern for the other player's payoff. Figure 2 shows a diagram of a Markov Decision Process that models the iterated Prisoner's Dilemma (assuming an explicit history length of 1 move).

Responsive strategies for the iterated Prisoner's Dilemma take their opponents' previous moves into account when deciding what move to make. Classic examples of responsive strategies include:

- Tit For Tat (TFT): cooperate on first move, then do what opponent did on previous move [1]
- Win-Stay, Lose-Shift (WSLS): if you get T or R , repeat previous move; otherwise, switch [5]
- majority: play the move that your opponent has most often used

Nonresponsive IPD strategies, on the other hand, are those whose moves do not depend on their opponents' previous moves. Some examples of nonresponsive strategies are:

- AIIC/AIID (cooperate/defect each move)
- CD, CCD, DDC, CCCD, ... (cooperate and defect in the specified repeating pattern)
- random

An intelligently conceived responsive strategy will not tolerate being consistently exploited (cooperating while its opponent defects), so the best outcome that an opponent of such a strategy can hope for is to evoke mutual cooperation. Against a nonresponsive strategy, however, the best strategy is to defect each time. These results suggest the finding in [2], which states that no single IPD strategy plays optimally against all other strategies.

1.3 Neural Networks and Games

Since no strategy is capable of playing optimally against all other strategies in IPD, I thought of training a neural network to adapt to other strategies in playing IPD. There are a few peculiarities of IPD as a game that meant that my approach to doing so was somewhat unconventional in the context of other approaches that have been made to playing games using neural networks. Specifically:

- IPD has no intrinsic termination condition; there is no obvious time at which the sequence of rounds stops. Most games have an obvious place to stop (usually when one player has won).
- IPD is not zero-sum: both players can “win” in each round. (Remember that each player is simply trying to maximize their own total payoff, not get a higher payoff than the other.)
- Since each round is treated individually, there is no way to “win” a series of rounds of IPD; thus the “reward” is not delayed until the end of the game.

These facts mean that learning mechanisms such as temporal difference learning [6] (which assume that the reward is delayed until the end of the game) are not relevant to learning to play IPD. Similarly, since no single strategy can be optimal against all other strategies, I wanted to try to discover a generalized neural network architecture that would be capable of playing effectively (if not optimally) against all other strategies. Thus, it seemed inappropriate for me to separate the training and testing phases of the development of this neural network: the neural network should always be trying to adapt to its opponent.

2 Related Work

Axelrod [1] has perhaps been the most influential researcher in the area of the iterated Prisoner’s Dilemma; he was responsible for the original tournaments and “ecological” investigations that defined the methods by which IPD strategies ought to be compared (and incidentally provided the venue that led to the notoriety of TFT).

Macy [5] investigated neural networks playing IPD under a variety of circumstances (including noise in the reports of outcomes). His approach differed from mine in that he considered only the case in which neural networks were playing against other networks like themselves.

3 Tools

For this project, I created a collection of Java programs, which can be divided into the following classes:

- Implementations of the basic neural network architecture (`NeuralNetworkLayer`, `NeuralNetwork`). (These were adapted from a C implementation by Karsten Kutza, available at <http://www.geocities.com/CapeCanaveral/1624/bpn.html>.)
- An abstract class `PDPlayer` which specified the necessary behavior of any player of IPD, and various subclasses (`PDNeuralNetwork`, `PDTitForTat`, `PDMajority`, `PDPeriodic`, etc.)
- `PDReferee`, which determined the payoff to each player based on the moves submitted to it.

- `PDDriver`, which, based on a combination of parameters specified on the command line and in configuration files, created instances of `PDPlayer` of the specified types, pitted them against each other for the specified number of rounds, and then summarized the results.

All of these tools were designed, as far as possible, to be easily generalizable to (a) partial cooperation and (b) multiple players; I will discuss this further in the concluding section.

4 Methods

4.1 IPD Parameters

For this experiment, I represented the choices (moves) as floating point values in the range $[0, 1]$, where a value of > 0.5 implied cooperation. The payoff values were set as follows: $(T, R, P, S) = (1.0, 0.7, 0.3, 0.0)$.

4.2 Opposing Strategies

The strategies against which I tested each neural network-based strategy include all the responsive and nonresponsive strategies that were explicitly listed in the section on the Prisoner's Dilemma.

4.3 Neural Network Architecture

4.3.1 Structure

Inputs and Outputs

The output of the network was a single value in the range $[0, 1]$ which represented the network's move for the round. Because I used the conventional sigmoidal transfer functions for each node, the network would, of course, actually generate values other than 0 or 1.

The input for the network consists of the outcome of previous round(s), that is, the moves and payoffs for each player (4 inputs per round of explicit history). I tested networks which had 1-5 rounds of explicit history (thus, 4-20 inputs) to see whether there would be a difference in their performance versus various other strategies. I chose to use all 4 elements of the outcome, rather than simply the payoff (from which the other information could, in theory, be derived) for two reasons: the network had no explicit knowledge of the payoff matrix values, and I thought that the redundancy of information might aid the learning process.

It is important to note that the actual value generated by the network for its output was used as one of its inputs, for two reasons. First, this causes it to be explicitly recognized as a recurrent network. Second, if only the move itself (C (1.0) or D (0.0)) had been used as the input, then the information about just how equivocal the previous decision had been would have been lost, and I suspected that this information would be important to the performance of the network.

Hidden Units

One of the questions which I hoped to answer by these investigations is whether explicit history (in the form of extra inputs) could be replaced or supplemented by implicit history (in the form of hidden units and attendant connections). As such, I tested several different architectures with a single hidden layer of 4 or 8 units, as well as several different perceptron architectures.

Each node in each layer was connected to each node in any adjacent layer(s), with weights originally uniformly distributed in the range $[0, 1]$.

4.3.2 Learning Rate

Because the network would not be given a separate training period for each strategy, I decided that it was appropriate to use a relatively high learning rate ($\eta = 1.0$) so that the network could adapt quickly.

(While I did not investigate this in detail, I did do some preliminary testing on different settings of the momentum parameter. It did not appear that they had any significant effect on the speed or accuracy with which the network adapted to any of the opposing strategies.)

4.4 Target Values

One of the most difficult problems to solve in the course of my investigation was to decide how the target value for the neural network should be determined. Essentially, this follows from the fact that the network has no way of knowing *a priori* what kinds of moves will cause it to get a good payoff, since we have already established that the best strategies in the presence of responsive opponents are different from those appropriate for nonresponsive ones.

Eventually I decided to investigate two heuristics. The first was the **cooperativeness** heuristic: if the other player cooperated, repeat the previous move; otherwise, play the opposite of the last move. The motivation for this was that, since a player always does better if the other cooperates, one could view the network's task as that of inducing cooperation in its opponent. This heuristic corresponds to a kind of supervised learning, in that the target value is directly specified based on the outcome, and is based on the Win-Stay Lose-Shift strategy.

The second was the **gratitude** heuristic: the higher the payoff, the higher the inclination to cooperate. In this specific context, I simply used the value of the payoff (which is, like the move itself, a value in the range $[0, 1]$). This is similar to reinforcement learning, in that the feedback is based on the payoff itself, rather than on a retrospective analysis of what the move "should have been".

My initial experiments determined that the cooperativeness heuristic alone tends to be overly optimistic vs. nonresponsive strategies, since it assumes that its decisions will have some effect on the other strategy's decisions. (The experimental results indicate that WSLS has a similar problem with nonresponsive strategies, especially ones which cooperate less than they defect.) On the other hand, the gratitude heuristic tends not to evoke cooperation from responsive strategies when its first move is a defection.

This suggested to me that I should try a linear combination of the two with parameter $f \in [0, 1]$:

$$t = (t_c)f + (t_g)(1 - f)$$

where t_c , t_g , and t were the target values given by the cooperativeness heuristic, gratitude heuristic, and combination, respectively.

I determined empirically that f should be approximately 0.3. After some analysis, I determined why this was the case: this value of f yields a target value of 0.51 (marginally more than that required to cooperate) in the presence of mutual defection. This simultaneously allows cooperation to emerge from mutual distrust (which is essential with strategies such as TFT) and nevertheless minimizes exploitation by nonresponsive strategies, (in the sense that while the neural network will still try cooperation every once in a while in the presence of consistent defection, it will not do so enough to harm it very much).

Of course, the specific value of f that should be used to evoke this behavior will depend on the values of the payoffs in general, and the value of P in particular. Since $t_c = 1.0$ in the presence of mutual defection, we have:

$$(1.0)(f) + (P)(1.0 - f) > 0.5$$

$$\implies f + P - Pf > 0.5$$

$$\implies f(1 - P) > 0.5 - P$$

$$\implies f > \frac{0.5 - P}{1 - P}$$

which yields a minimum value for f of $\frac{2}{7}$ for this value of P .

One effect that this choice of target values has is that it makes this neural network-based strategy nonexploitative (that is, it will not exploit consistent cooperation for very long), as well as giving it a partial immunity to exploitation (as noted above).

4.5 Evaluation

The metric that I used to compare the performance of the different neural networks was a measurement of the average payoff over the entire series, which I fixed at 1000 rounds. The figure of 1000 rounds was chosen to be short enough so that I could see the effect on the average payoff of the time required to learn, but long enough that I could get an idea of the network's performance in the long term.

I interpreted the results (including the trace of the actual outcomes in addition to the average performance metric) in light of the nature of the opposing strategy: for responsive strategies, I checked to see whether it learned to induce consistent cooperation?, and for nonresponsive strategies, I tried to determine whether it managed to learn an appropriate response to the pattern (if there was one).

I included the results from testing two different responsive strategies (Tit for Tat (TFT) and Win-Stay Lose-Shift (WSLS)), and for the "optimal" strategy (a strategy that knows which strategy it's playing against, and chooses the best move accordingly), so as to provide a basis for comparison for the performance of the neural network strategies.

5 Results

The results shown below are all based on experiments in which the neural network started by defecting. This choice was made because when the neural network started by cooperating, the neural network-based strategy immediately evoked mutual cooperation from all the responsive strategies, which is optimal but perhaps overly optimistic for a true neural network (which, since the weights are initialized to random values, may initially either cooperate or defect). (Of course, it doesn't matter what the neural network's first move is if its counterpart is nonresponsive.)

The neural networks are identified by the number of nodes in each layer, from input to output, so "ANN(4,8,1)" is an (artificial) neural network that has 4 input nodes (corresponding to an explicit history of length 1), 8 hidden nodes, and 1 output node, and "ANN(16,1)" refers to a neural network that has 16 input nodes (history length 4) and one output node.

Average Payoff (1000 iterations)

Strategies	self	TFT	WSLS	majority	AIID	CD	CCD	DDC	CCCD	DDDC	random
optimal	-	0.70	0.70	0.70	0.30	0.65	0.77	0.53	0.83	0.48	0.65
TFT	-	0.70	0.70	0.70	0.30	0.50	0.56	0.43	0.60	0.40	0.50
WSLS	-	0.70	0.70	0.70	0.15	0.50	0.62	0.33	0.68	0.33	0.50
ANN(4,1)	0.68	0.66	0.69	0.29	0.29	0.49	0.53	0.43	0.54	0.44	0.52
ANN(8,1)	0.67	0.68	0.69	0.29	0.29	0.49	0.54	0.41	0.58	0.40	0.53
ANN(12,1)	0.67	0.67	0.69	0.28	0.29	0.49	0.56	0.42	0.57	0.37	0.52
ANN(16,1)	-	0.65	0.69	0.28	0.28	0.48	0.56	0.41	0.59	0.40	0.52
ANN(20,1)	-	0.64	0.69	0.28	0.28	0.48	0.56	0.41	0.59	0.38	0.52
ANN(4,4,1)	0.33	0.32	0.68	0.29	0.29	0.48	0.53	0.48	0.53	0.43	0.51
ANN(4,8,1)	-	0.38	0.70	0.28	0.28	0.48	0.53	0.45	0.53	0.43	0.51
ANN(8,8,1)	-	0.33	0.70	0.28	0.28	0.49	0.53	0.41	0.56	0.42	0.51
ANN(12,4,1)	-	0.32	0.69	0.29	0.29	0.49	0.55	0.48	0.54	0.41	0.50
ANN(12,8,1)	-	0.33	0.70	0.28	0.28	0.49	0.55	0.42	0.56	0.38	0.51

6 Analysis

6.1 Responsive Opponents

All the ANNs were able to evoke mutual cooperation from WSLS in fairly short order (ANN(4,1) did so within 15 rounds). However, while the perceptron-based ANNs successfully induced TFT to mutual cooperation (within 50 rounds for ANN(4,1)), the more complex ANNs were, in general, unsuccessful in doing so. Since ANN(4,8,1) did succeed in establishing mutual cooperation by the end of the series, I suspect that the main problem here was that the addition of the hidden nodes made the network complex enough that its learning rate was too slow for most practical purposes. (The apparent decline in performance vs. TFT of the perceptron-based networks as the size of the history increased is probably attributable to the same cause.)

None of the ANNs were able to evoke mutual cooperation from the majority strategy. The reason for this is simply that it takes too long (generally 10 turns or more) for the ANN to try cooperation in the face of persistent mutual defection, and by that time, it would require as many rounds of unrequited cooperation in order for the ANN to evoke cooperation from the majority strategy. Since the ANN is unwilling to repeatedly try cooperation given repeated defections, mutual cooperation is never achieved.

6.2 Nonresponsive Opponents

Since the ANNs prefer to cooperate if they believe that their opponent will do so as well, they did not evolve an AIID strategy against nonresponsive opponents. Generally speaking, the mechanisms that I instituted for target values caused the ANNs, when presented with a periodic strategy, to converge on a pattern that approximates the one with which it is presented (so the ANNs evolve to CCD given a CCD strategy, etc.) Thus, they were successful to the extent that their approximations were accurate.

All the ANNs were successful at evolving a workable response to the shorter patterns (AIID and CD). (In fact, the results vs. AIID were considerably better than those for WSLS.) However, the results for the longer patterns were more ambiguous, and should be supplemented with results from additional experiments in order to increase my level of confidence in the following analysis.

It appears to be the case that a larger history size is more helpful to ANNs that are attempting to adapt to periodic strategies that are primarily cooperative, and that in such cases, increasing the history length to at least the length of the period tends to increase the ANN's performance. A larger history size does not appear to be of much use when in trying to adapt to a largely uncooperative periodic strategy. In any case, neither adding more history nor more complexity allows an ANN to surpass the performance of either TFT or WSLS against such opponents.

Contrariwise, adding a hidden layer seems to be of more utility to those ANNs that are playing against primarily uncooperative strategies, although both the perceptron-based neural nets and those with a hidden layer outperformed WSLS against the period 3 strategies, and both TFT and WSLS against those of period 4. As with responsive strategies, however, adding too much complexity seems to degrade the performance.

Interestingly, all the ANNs seemed to do fairly well versus the random strategy—better than either TFT or WSLS. I conjecture that the unpredictability of the random strategy causes the ANN to be somewhat prejudiced towards defection rather than cooperation, since its overtures of cooperation will in general not be rewarded.

6.3 Weight Patterns

I did not rigorously investigate the patterns of weights for all ANN designs, but I did do some spot checking to see whether there were any interesting patterns.

The weights for ANN(4,1) tended to differ based on whether the network had adapted to a responsive strategy or a nonresponsive one. In response to responsive strategies, it tended to have weights in the range $[-.15, .65]$, with a slight tendency to weight outcomes from the other player higher than its own. After having adapted to a nonresponsive strategy (DDC), however, all weights were in the range $[-.25, .05]$, with no such bias.

The weights for ANN(8,1) when presented with a period 4 sequence were somewhat more defined. Specifically, the moves were weighted much more strongly than the payoffs ($\pm 0.9 - 3.0$ vs. $\pm 0.05 - 0.30$) and its own moves were weighted positively, while the other player's moves had strongly negative weights.

7 Conclusions

It appears that a very simple ANN can adapt with reasonable success to a number of different strategies. TFT and WSLS tend to perform better against other responsive opponents, especially if the game is short (the fact that they have a prefabricated policy essentially eliminates the ramp-up time required by ANNs). However, an ANN generalized learning strategy can be better than TFT and WSLS against (predominantly uncooperative) nonresponsive strategies.

While there is certainly room for improvement in the learning mechanism that I chose, it seems that a hybrid learner which uses both reinforcement and supervised learning may perform better than either of them in isolation.

Finally, while there may be some utility to explicit history in learning primarily cooperative periodic strategies, and some advantage to including hidden units in adapting to mostly uncooperative ones, the results are inconclusive, and warrant further investigation.

8 Future Work

There are several things related to this problem that I would like to pursue, some of which have to do with modifications to the neural network, and some with the Prisoner's Dilemma.

8.1 Neural Network

One of the shortcomings that the neural network has is that it tends to take perhaps too much time to adapt to most strategies. There are a few ways that I can think of that might help with this:

- a higher learning rate, perhaps in conjunction with momentum
- removing some inputs (in theory, all inputs can be calculated given one player's payoff, if the payoff matrix is known).
- giving the network an explicit representation of the payoff matrix, perhaps by adding an extra output layer to the network with fixed weights

There are also modifications to the learning scheme that might be useful, such as enhancing the neural network to predict other quantities in addition to calculating the move. These quantities could include other players' moves (although it's not obvious what should be done with this information) and conditional probabilities (such as the probability that the other player will cooperate next turn if the ANN cooperates this turn). Another related modification would be to use time-discounted sums to determine appropriate target values, so that the cooperativeness and gratitude heuristics would take into account previous rounds' results as well as the current one.

Finally, while I was able to suggest a lower bound for f , it would be advisable to also determine the analogous upper bound.

8.2 Prisoner's Dilemma

In order to more rigorously determine the fitness (relative performance) of ANN-evolved strategies, it would be a good idea to test them against a wider array of responsive strategies.

One way to generalize the iterated Prisoner's Dilemma to multiple players is to give each player only one move, which is used as each player's move against each other in all possible pairwise contests. As with the original version, the greater the number of cooperators, the greater the benefit for all players...but unlike the original version, it is not possible to retaliate directly against those that defect. This is an interesting model for sociological interactions (and in fact is a pretty good model of the Tragedy of the Commons). I would expect it to be difficult for cooperation to take hold if no further modifications were made.

An additional generalization that might allow an atmosphere of cooperation to establish itself would be to allow each player to choose a level of cooperation as a real value in the range $[0, 1]$. Partial cooperation would allow individuals to show a willingness to cooperate without exposing themselves too much to the risk of others' defection. This generalization could, of course, also be applied to the original two-player game.

Finally, I am not certain whether anyone has developed a method for doing sensitivity analysis on Prisoner's Dilemma tournament results, but clearly, such results are not always stable under changes in the payoff structure, and such a method would be useful for further analysis of tournaments.

References

- [1] Robert Axelrod. *The Evolution of Cooperation*, Basic Books, 1984.
- [2] Robert Boyd and Jeffrey P. Loberbaum. No Pure Strategy is Evolutionarily Stable in the Repeated Prisoner's Dilemma Game. *Nature*, 327:58-59, 1987.
- [3] Jean-Paul Delahaye, Philippe Mathieu, and Bruno Beaufils. Iterated Prisoner's Dilemma. <http://www.lifl.fr/IPD/ipd.frame.html>
- [4] Douglas Hofstadter. The Prisoner's Dilemma Computer Tournaments and the Evolution of Cooperation. *Metamagical Themas: Questing for the Essence of Mind and Pattern*, Basic Books, 1985.
- [5] Michael Macy. Natural Selection and Social Learning in Prisoner's Dilemma: Coadaptation with Genetic Algorithms and Artificial Neural Networks. *Sociological Methods and Research*, 25(1):103-137, 1996.
- [6] Gerald Tesauro. Temporal Difference Learning and TD-Gammon. *Communications of the ACM*, 38(3), 1995.