When Machine Learning Meets AI and Game Theory

Anurag Agrawal, Deepak Jaiswal

***Abstract*—We study the problem of development of intelligent machine learning applications to exploit the problems of adaptation that arise in multi-agent systems, for expected-long-term-profit maximization. We present two results. First, we propose a learning algorithm for the Iterated Prisoners Dilemma (IPD) problem. Using numerical analysis we show that it performs strictly better than the tit-for-tat algorithm and many other adaptive and non-adaptive strategies. Second, we study the same problem from the aspect of zero-sum games. We discuss how AI and Machine Learning techniques work closely to give our agent a 'mind-reading' capability.**

***Index Terms*—Iterated Prisoner's Dilemma, Evolution Theory, Tit-For-Tat algorithm, Intelligent Agent (IA), Re-inforcement Learning.**

I. Introduction

The interaction of learning and evolution is a topic of great interest in evolutionary computation[1],[2],[7],[8]. It plays an important role in application areas such as multi-agent systems, economics, politics, and biological modelling. Each of these studies have one common thread: they involve the study of systems of interacting autonomous individuals in a population, whether the individuals are artificially created intelligent agents, human beings or other biological organisms. Some of the questions they study are: What will be the equilibrium set of behaviours? Will there be any equilibrium? How can cooperative behaviours evolve?

Approaches to answer these questions have been proposed in last few decades[4],[5],[9],[10]. The purpose of this paper is not to study the evolution of cooperative behaviours. Instead, we take a specific example of a cooperation based game (The Prisoner's Dilemma) and propose an algorithm which does strictly better than most of the traditional evolution-based algorithms in terms of maximizing the long term expected payoff. Further, we use these ideas to develop an intellegent agent for a zero-sum game. We find that our algorithm gives positive expected returns in the long run when run against both simple and evolutionary strategies.

We report on a simulation study which explores what happens when learning and evolution interact in an evolutionary game scenario. We explore interactions in a co-evolving population of model-based adaptive agents and fixed non-adaptive agents playing Iterated Prisoner's Dilemma (IPD). The simulation environment is similar to Axelrods well-known IPD simulation study environment.

The rest of the paper is as follows. We first introduce the Iterated Prisoner's Dilemma problem and review previous learning and evolution-based approaches to its study. Next, we discuss our algorithm in detail. In the next section, we then give a slightly different version of out algorithm when applied to zero-sum games. Numerical Analysis of the results obtained is provided in the penultimate section. The paper ends with a conclusion section with some discussion of possible future work in this direction.

Anurag Agrawal (anurag07@stanford.edu) is with the Department of Electrical Engineering, Stanford University, Stanford, CA. Deepak Jaiswal (deepakjaiswal224@gmail.com) is with the Department of Civil Engineering, Stanford University, Stanford, CA. This work was conducted as a part of CS-229 Machine Learning course at Stanford University. A part of this work will also be included in a paper to be submitted to IEEE Journal on Selected Areas of Communication: Special Issue on Game Theory in Wireless communications.

II. The Iterated Prisoner's Dilemma Problem

The basic version of Prisoner's Dilemma can be presented as follows: Suppose there are two prisoners A and B, arrested for being involved jointly in a crime. The police does not have any evidence to file the charges. The police decides to offer the following deal to each of the prisoners separately: "If you agree that the other prisoner was involved in the crime, and the other prisoner remains silent, you will be set free and the other prisoner will receive a ten year sentence, and vice versa. But if both of you remain silent, both will receive a five year sentence." The prisoners are not allowed to communicate with each other. What should they do in order to minimize their losses, assuming that A is not 'learned' to B's decision making process? The answer is simple: Both should try to maximize their expected profits (minimize expected losses). In the IPD problem the same game is played K number of times, $K > 1$ is unknown to both the players. The outcome of the previous game is known to both the players before starting a new game. This gives an opportunity to learn from experience and make better decisions in later games.

Certain conditions have to hold in defining a Prisoners Dilemma game[8]. Firstly, the order of the payoffs is important. The best a player can do is T (Temptation to Defect). The worst a player can do is to get the Sucker payoff, S. If the two players cooperate then the Reward for that Mutual Cooperation, R, should be better than the Punishment for Mutual Defection, P. Therefore, the following must hold. T $>$ R $>$ P $>$ S Secondly, players should not be allowed to get out of the dilemma by taking it in turns to exploit each other. That is, taking turns should not be as good an outcome as mutual cooperation. Therefore, the reward for mutual cooperation should be greater than the average of the payoff for the temptation and the sucker: R $>$ (S + T) / 2 To be definite, we will choose the commonly used values T = 5, R = 3, P = 1, and S = 0.

The question to be asked here is that: as a perfectly rational player, playing another perfectly rational player, what should you do in such a game? The Nash equilibrium solution to this problem is to defect. We discuss this briefly below.

Suppose you think the other player will cooperate. If you cooperate then you will receive a Reward of 3 for mutual cooperation. If you defect then you will receive a payoff of 5 for the Temptation to defect payoff. Therefore, if you think the other player will cooperate, you should defect, to give you a payoff of 5. But what if you think the other player will defect? If you cooperate, then you get the Sucker payoff of zero. If you defect then you would both receive the Punishment for mutual defection of 1 point. Therefore, if you think the other player will defect, you should defect as well.

So, you should defect, no matter what option your opponent chooses. Of course, the same logic holds for your opponent. And, if you both defect you receive a payoff of 1 each, whereas, the better outcome would have been mutual cooperation with a payoff of 3 each. This is the dilemma. In other games, there may not be a dominant strategy, and other notions of ”solving” the game are used. In Nash equilibrium the two players adopt a pair of strategies such that neither player can get a better payoff by deviating from their strategy. In other words, each strategy is a best response to the other. Depending on the game, there may be no Nash equilibrium, a unique one, or many equilibria. Aside from these strategies, there is another kind of strategy that can be considered, in which players are allowed to use randomness (e.g. a roll of a die) to decide their moves. In game theory these are called mixed strategies or stochastic strategies, whereas those without randomness are called pure strategies.

In contrast to the rational conclusion of mutual defection, in real-life instances of Prisoner's Dilemma, cooperation is often observed. Why is it so? One suggested explanation is that in real life, the players would have an expectation that they may meet the same opponent in the future, and he might remember a previous defection and take revenge by defecting on us next time we play.

In the iterated game, player strategies are rules that determine a player's next move in any given game situation (which can include the history of the game to that point). Each player's aim is to maximize his total payoff over the series. If you know how many times you are to play, then one can argue that the game can be reduced to a one-shot Prisoner's Dilemma[8]. The argument is based on the observation that you, as a rational player will defect on the last iteration - that is the sensible thing to do because you are in effect playing a single iteration. The same logic applies to your opponent. Knowing that your opponent will therefore defect on the last iteration, it is sensible for you to defect on the second to last one, as your action will not affect his next play. Your opponent will make the same deduction. This logic can be applied all the way back to the first iteration. Thus, both players inevitably lock into a sequence of mutual defections.

One way to avoid this situation is to use a regime in which the players do not know when the game will end. Nature could toss a (possibly biased) coin to decide. Different Nash equilibria are possible, where both players play the same strategy. Some well-known examples are:

(1) Tit-for-tat: cooperate on the first move, and play the opponents previous move after that;

(2) Grim: cooperate on the first move, and keep cooperating unless the opponent defects, in which case, defect forever;

(3) Pavlov: cooperate on the first move, and on subsequent moves, switch strategies if you were punished on the previous move.

In our version of the problem we assume that K is large, so that the opponents have enough opportunities to learn from each other. Also, the payoff matrix is given by table-1:

(C,C) $\rightarrow (3,3)$	(C,D) $\rightarrow (0,5)$
(D,C) $\rightarrow (5,0)$	(D,D) $\rightarrow (1,1)$

where $(C,C) \rightarrow (3,3)$ means that if both cooperate, both get a payoff of 3, and so on.

In 1980, Robert Axelrod staged two round-robin 'tournaments' between computer programs designed by participants to play IPD. Many sophisticated programs were submitted. In each case, the winner was Anil Rapaports submission, a program that simply played Tit-for-Tat. In 1987, Axelrod carried out computer simulations using a genetic algorithm (nowadays it would be called a co-evolutionary simulation) to evolve populations of strategies playing the IPD against each other. In these simulations, tit-for-tat-like strategies often arose as the best but it was proved that they were not optimal. In fact, Axelrod used this to illustrate that there is no 'best' strategy for playing the IPD in such an evolving population, because success depends on the mix of other strategies present in the population. Axelrods simulations illustrate a different approach to studying the IPD one in which the players are not perfectly rational, and solutions evolve rather than being deduced.

III. The Algorithm

In any study of games involving a multi-agent system, the players can be categorized under two broad sectors - adaptive and non-adaptive agents. Consider a population of IPD Playing agents. During their lives, these agents meet and interact with each other where they may choose to cooperate or defect and receive payoffs as given by table [1]. Higher reproductive fitness is shown by those who get higher payoffs. The choices they make are prescribed by genetically determined strategies. This is the scenario Alexrod simulated in his experiment. In to this population, we introduce our own set of players (by mutation of a subset of existing set of players). These are the agents who try to model the strategies of their fellow players and use this model to maximize their own payoffs. Hence, the population now has a set of intelligent, exploitative and adaptive agents. This is the scenario simulated by us in our experiments.

It is evident that their are two kinds of interactions possible: adaptive v.s. adaptive, and adaptive v.s. non-adaptive. The objective of each (adaptive) player who interacts with another player is to first figure out whether the opponent is adaptive or non-adaptive. Next, it should make an intelligent move so as to maximize its long term profits and minimize the difference in losses. In this study, we have chosen to restrict the strategies under consideration to a class of finite memory stochastic strategies (”behavioural strategies”) that can be described in terms of fixed set of probabilities. This is general enough to represent quite complicated strategies, but it does not include, for example, some strategies defined by finite state automata. However, most of the well-known strategies for IPD fit into the framework.

We define the Probability function P_n such that $P_{n+1} = f_n(e_1, e_2, ..., e_n)$ gives an agent the probability of cooperation in the $(n+1)^{th}$ move given the outcomes $e_1, e_2, ..., e_n$ of the previous n moves of games played against a particular opponent. Here, $f(.)$ is the function that depends upon the strategies in play. In a 'pure' strategy, the value of each function $f(.)$ is either 0 or 1, otherwise the strategy is said to be stochastic. A zero-order strategy is one in which f_n is a constant. For example, a completely random strategy is a zero-order strategy for which $f_n(.) = 0.5$ for all values of n. A first order strategy is one in which f_n is a function of e_n only (and is independent of $e_1, ...e_{n-1}$). For example, the Tit-for-Tat strategy is a first order strategy in which $P_{n+1} = 1$ if the opponent cooperated in the n^{th} move, zero otherwise. In other words,

$$P_{n+1} = 1 \text{ if } e_n = (C, C) \text{ or } (D, C) \tag{1}$$

$$P_{n+1} = 0 \text{ if } e_n = (C, D) \text{ or } (D, D) \tag{2}$$

where C denotes the outcome 'Cooperate' and D denotes the outcome 'Defect'. P_1 is set to 1 by default when the experiment starts.

We now introduce the three basic steps to the generalized version of algorithm:

(1) Play a 'bad' game (i.e. Always Defect) for the first N moves against each new opponent. Similarly, for the next N moves against each opponent, play a 'nice' game. Isolate the opponents who play with zero and first order strategies using the outcomes of these $2N$ moves. Note that all the non-adaptive agents will be isolated in this step. That is, stratgies like 'Tit-For-Tat', 'Random', 'Always-Defect', 'Always-Cooperate' etc. can all be decoded in this step. Goto step 2.

(2) If the opponent is non-adaptive, play with the optimum strategy (derived in the first step) for the next M moves, where $M >> N$. Else, goto step 3. After M moves, loop back to step 1.

(3) If the opponent is adaptive, use Reinforcement Learning and adaptive-control techniques to predict its strategy and maximize your profits, for the next M moves. After M moves, loop back to step 1.

The values of M and N should satisfy $M \geq 200N$. We now define the parameters for our Markov Decision Model which will be used in Step-3 of our algorithm. For each move, there are 4 possible outcome states (C, C), (C, D), (D, C) and (D, D) as depicted in table [1]. We denote this set by S. The set of actions A consists of two elements: $\{C, D\}$. $R : S \times A \rightarrow \mathbf{R}$ is the Reward function. The values taken by the reward function are discussed in the previous section. P_{sa} are the state transition probabilities. Given a fixed policy $\pi : S \rightarrow A$, the value function V^π gives the total expected reward in following that strategy:

$$V^\pi(s) = R(s) + \gamma \sum P_{s\pi(s)}(s') V^\pi(s') \tag{3}$$

where γ is the discount parameter and the the summation is over the set of all states s' belonging to S. Note that the description of the Value function is not yet complete. Because we do not know what our opponent's current move is going to be, we cannot complete determine the state s we want to visit next. For instance, if we decide to cooperate, our next state will be of the form (C, X) where X is random variable taking values in $\{C, D\}$. Hence, we take the expected value of the reward we expect to receive over the two possible states. We define the optimal value function according to:

$$V^*(s) = \max_\pi E[V^\pi(s)] \tag{4}$$

where expectation is taken over the two possible states. The optimal policy $\pi^* : S \rightarrow A$ is given by

$$\pi^* = \arg\max_\pi \, E[V^\pi(s)] \tag{5}$$

We solve the Bellman equations numerically to compare the performance of our algorithm with other algorithms run in similar experimental environment. Numerical analysis is presented in the 'Numerical Results' section.

IV. Zero-Sum Games

We now discuss some similar situations that arise in zero-sum games. A zero-sum game is a mathematical representation of a situation in which the sum of profits (may be positive or negative) of all the participants is equal to zero. In other words, the profits are exactly balanced by the losses. The Nash equilibrium solution for a two player zero-sum problem can be easily obtained as a solution to a convex optimization problem. We consider a problem of an iterated zero-sum game. One of the greatest temptations of designers in the gaming industry has been to create a false impression of 'learning' and, until recently, machine learning hasn't been used in many games. In complex zero-sum games like Poker, decision making is influenced not only by the scenario of the current game, but also by a learning agent that is composed of a few fundamental parts like: a learning element, a curiosity element, a performance element and a performance analyzer. The learning element is the one responsible for modifying agent's behavior on each iteration of the underlying algorithm. Here we are assuming that the participants do not have any information about their opponents' playing styles before the start of first game. The curiosity element is one that alters the behavior predicted by the learning element to prevent the agent from developing bad habits or biases. The performance element is the one that decides the action based on output of the curiosity element. The performance analyzer is the one responsible for analyzing the outcome of the decision made in the previous iteration and feeding it back to the learning element in the current iteration. Decision trees, neural networks and belief propagation networks are all goods methods for modeling 'learning' and 'reasoning'. The most common pitfall encountered in designing a Learning AI using this approach is that the computer program is taught badly if the human player is not familiar with the game and/or plays the game stupidly. Our algorithm, as discussed above, considers a Reinforcement Learning approach to model the learning element. For zero-sum games, we add the curiosity element of our algorithm to make sure that we do not get trapped in the above pitfall.

The problem is as follows: Two players are dealt one card each. The cards have either 0 or 1 written on them with equal probability. If you happen to receive a card which says '1', it implies you have a strong hand, and if it says 0 it means that your hand is weak. Same is the case with your opponent. Now both of you are required to declare the strengths of your hands, but both have an option of 'bluffing'. The objective is to make the best guess of the opponent's hand and also to try that your opponent is not able to guess your hand correctly. If your guess is right and your opponent's guess is wrong, you get a score of 20 whereas your opponent gets -20, and vice versa. If both are correct or both are wrong, both get a score of 0. Thus, in simple words, our objective is just to maximize our own score. If the game is played exactly once, the solution is to make a guess of opponent's hand by flipping a coin, and to bluff about your own hand hoping that your opponent will believe you. We call this problem the 'Bluff-Catcher's' problem.

Now, we deal with the iterated version of this problem. Assuming that you will play this game with your opponent a large number of times, is it possible to make more educated guesses of your opponent's hand to determine wether or not the opponent is bluffing? The answer is Yes. We again consider a similar experiment environment. The difference is that now 'Defecting' takes the role of 'Bluffing' B and that 'Cooperating' takes the role of 'Telling Truth' (T). The steps of the modified version of the algorithm are as follows:

(1) Play a 'bad' game (i.e. always bluff) for the first N moves against each new opponent. Similarly, for the next N moves against each opponent, play a 'nice' game. Isolate the opponents who play with zero and first order strategies using the outcomes of these $2N$ moves. Note that all the non-adaptive agents will be isolated in this step. That is, strategies like 'Tit-For-Tat', 'Random', 'Always-Bluff', 'Always-Tell-Truth' etc. can all be decoded in this step. Goto step 2.

(2) If the opponent is non-adaptive, play with the optimum strategy (derived in the first step) for the next M moves, where $M >> N$. Else, goto step 3. After M moves, loop back to step 1.

(3) If the opponent is adaptive, use Reinforcement Learning and adaptive-control techniques to predict its strategy and maximize your profits, for the next M moves. Before making a move, turn to the curiosity element. With probability δ, the curiosity element will inform you to switch to a sub-optimal strategy. If that indeed happens, keep a separate record of the reward you got after playing this move. Feed it back to the curiosity element. The curiosity element will adjust the value of δ by comparing the reward you got in this move to the rewards in previous moves. After M moves, loop back to step 1.

The reinforcement learning model used is quite similar, and hence is not discussed in detail again. Thestarting value of δ

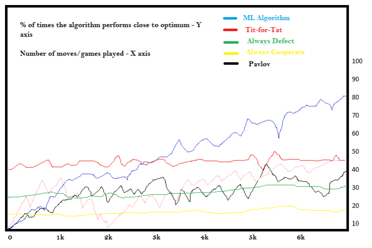

Fig. 1. % of times algorithm performs close to optimum versus the number of games played.

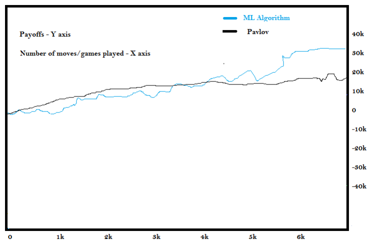

Fig. 4. Expected Score versus Number of games played. (IBC problem)

was chosen to be 0.01. The zero and first order strategies were easily cracked, hence their plots are not included again. Some interesting observations were obtained when we compared the performance of our strategy to the Pavlov-type strategy. We observed that the n^{th} order Pavlov strategy performed considerably well against our algorithm in the initial phase of the simulatiion. The overall score it earned was high (and sometimes even more than the learning algorthm) but the rate of increase of it's score always showed a decreasing trend. Brief comparison between the two strategies are made in the Numerical Analysis section.

V. Numerical Results

Bellman equations are very hard to solve analytically even with small state space as is considered in the current problem. We employed the method of undetermined equations to solve a few special cases of the Bellman equations. But most of the other (general) cases were solved numerically. We obtain some useful insights related to the performance of our algorithm in our simulation environment as compared to that of Tit-for-Tat, Pavlov and a few special zero-order strategies. Fig. 1 shows the % of times different algorithms performs close to optimum. Here 'optimum' refers to the score that would have been achieved if you would have predicted each move of your opponent correctly and hence would have chosen the best possible series of actions. Here 'close' means that the score obtained was at least 85% of the optimum score. We see a clear distinction between the performances of our algorithm and that of Tit-for-Tat when the number of moves is close to 8000. Each algorithm was run against all the other algorithms and the average performace was plotted as a functon of number of moves. We observe that the algorithms which do not learn with time (for ex. the 'Always Cooperate' algorithm) were punished badly by our learning algorithm and hence showed

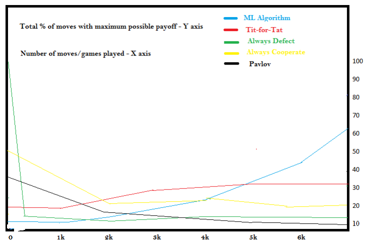

Fig. 2. % of moves for which maximum possible payoff is achieved versus number of games played.

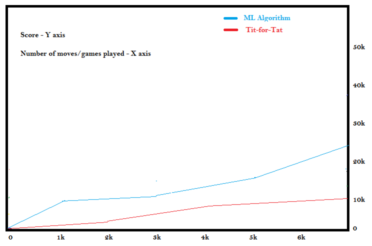

Fig. 3. Total Score achieved by an Algorithm versus Number of games played.

poor performances. Tit-for-Tat did well against some adaptive algorithms (including our own algorithm) but did not exploit the 'learning' benifit against zero-order algorithms.

Fig. 2 shows the % of moves for which maximum possible payoff was achieved. As expected, the ML agorithm showed a monotonically increasing graph as a function of the number of moves. It implies that our algorithm gives better predictions as the number of moves increases and hence increases the total success %. On the other hand, the Tit-for-Tat algorithm gives a mediocre performance which is almost independent of the number of moves. The performances of the Pavlov and the zero-order strategies in this regard are also independent of the number of moves. In fact, they tend to degrade with time because the adaptive learning algorithm tends to screw them up more often.

Fig. 3 shows the total average score achieved by our learning algorithm and the Tit-for-Tat. Each of these algorithms were run against the other four algorithms and the score obtained were recorded as a function of moves. The average of these four scores were plotted for both the algorithms. An interesting observation is that when our algorithm is sufficiently 'learned' (i.e. number of moves $\geq$ 1000) the rate at which the score increases is itself an increasing function of the number of moves. Of course, this is a local phenomenon because once our algorithm is sufficiently 'learned' it cannot show this trend anymore. On the other hand, although Tit-for-Tat also shows an increasing trend, the rate of increase of score tends to decrease after about 3000 moves.

Fig. 4 compares the performances of the ML Algorithm with the Pavlov-type learning algorithm for the Iterated Bluff-catcher's problem. Both of these algorithms were run against the other four algorithms and the score obtained were recorded as a function of moves. The average of these four scores were plotted for both the algorithms. One can see that Pavlov performes better than our learning algorithm initially. This is because of the sub-optimal performance given by our algorithm in the initial $2N$ moves. Both the curves show a decreasing score-rate, however a considerable difference in the scores can be noted when the number of moves $\geq$ 6000.

VI. Conclusion

A machine learning approach to the Iterated Prisoner's dilemma problem has been studied. When the agents involved use finite memory stochastic strategies, it has been shown that the learning algorithm performs strictly better than the Tit-for-Tat algorithm. The same algorithm, when applied to the Bluff-Catcher's Problem (with minor variations including an introduction of a curiosity element), performs well against the n^{th} order Pavlov-type adaptive strategy. It also completely exploits an opponent running a first or second order stochastic strategies. For future work, we expect to apply modifications of our algorithm to many complex zero-sum games that we encounter in daily life. One such example is Poker. We are currently in middle stages of an developing AI agent for Poker.

VII. Acknowledgements

We thank Prof. Andrew Ng and CS-229 course staff at Stanford University for their continuous support with the project work and course material. We also thank Gowtham Kumar of Electrical Engineering department for several brainstorming discussions. Although we tried our best to remove any error, we understand that since our paper has not been reviewed by peers, it is likely that we overlooked some logical/grammatical/diction errors.

References

[1] G. Weiss, "Distributed Artificial Intelligence Meets Machine Learning"

[2] M.Singh and M. Huhns, "Challenges for Machine Learning in Cooperative Information Systems"

[3] Sklansky, David 2005, "The Theory of Poker (Fourth ed.). Las Vegas: Two plus two"

[4] Rabin, Steve, "AI Programming Wisdom", Charles River Media, INC. 2002

[5] http://ai-depot.com/GameAI/Learning.html

[6] Axelrod, R. and D'Ambrosio, L., "Annotated Bibliography on the Evolution of Cooperation"

[7] Axelrod, R., "The evolution of strategies in the iterated prisoner's dilemma", in Genetic Algorithms and Simulated Annealing (L. Davis, Ed.), Pitman, 1987

[8] Philip Hingston, Graham Kendall, "Learning versus Evolution in Iterated Prisoners Dilemma"

[9] Fudenberg, D. and Levine, D. The Theory of Learning in Games. Cambridge, MA: MIT Press, 1998.

[10] Fogel, D.B. Evolving Behaviours in the Iterated Prisoners Dilemma, Evolutionary Computation, Vol. 1:1, pp 77-97, 1993.

[11] Jehiel, P. and Samet, D. Learning to Play Games in Extensive Form by Valuation, NAJ Economics, Peer Reviews of Economics Publications, 3, 2001.