Learning to Cooperate in Normal Form Games

Steven Damer and **Maria Gini**
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455, USA

Abstract

We study the problem of achieving cooperation between two self-interested agents that play a sequence of randomly generated normal form games, each game played only once. To achieve cooperation we extend a model used to explain cooperative behavior by humans. We show how a modification of a pre-regularized particle filter can be used to detect the cooperation level of the opponent and play accordingly. We examine how properties of the games affect the ability of an agent to detect cooperation and explore the effects of different environments and different levels of conflict. We present results obtained in simulation on hundreds of randomly generated games.

Introduction

In this paper we address the problem of cooperation between two self-interested agents. Since each agent is self-interested we expect it to select the action that provides its own highest expected benefit, without regard for the opponent's outcome. However, it has been observed that agents who interact repeatedly with each other tend to be better off if they cooperate at some level. A classical example is the prisoner dilemma, a game that has been studied extensively in its iterated form (for instance, (Rapoport and Chammah 1970)), where cooperation can be achieved using Tit-for-Tat (Axelrod 1984). In iterated games players can observe the opponent's behavior and reciprocate.

For our study we use a sequence of non-zero-sum normal form games, each played only once by the same two players. Since each agent plays against the same opponent, they have opportunities to observe each other and play accordingly. However, there are two issues that make this difficult: (1) the actions in the games are not labeled as cooperative or uncooperative. This is because the games are generated randomly and labeling will require extensive manual intervention; (2) the games played are all different, so understanding if the opponent is cooperative or not is much harder than when playing the same game repeatedly.

We assume that opponents may be willing to reciprocate actions that benefit them, but they may also choose to exploit our agent. Reciprocation is an effective method to achieve cooperation without much exposure to the risk of being exploited, since the agent will stop cooperating when it detects a noncooperative opponent. We assume that agents play a Nash equilibrium of the game once they have taken into account how much the opponent cares about them (we will define this later as "the attitude of the opponent"). This is not necessarily rational, because the opponent may not be playing the Nash equilibrium, but it is convenient and limits the choices to a discrete set (i.e. one among the Nash equilibria for each game). We do not assume both agents use the same Nash equilibrium, relaxing one of the assumptions made in (Conitzer and Sandholm 2007).

The main contributions of this paper are (1) the use of a pre-regularized particle filter with discrete smoothing to learn the attitude of an opponent in this environment; (2) an in-depth examination of how the games affect the ability of an agent to detect cooperation. Specifically, we explore the effects of different levels of complexity and different levels of conflict between the agents; (3) empirical results obtained in simulation on hundreds of randomly generated games. Our results show that cooperation is possible and that an agent can detect if its opponent is reciprocating, even in the difficult environment we use.

Background on the Model of Cooperation

We extend the work presented in (Damer and Gini 2008a) where players play a sequence of different normal form games. Specifically we use randomly generated normal form games with 16 actions per player, and payoffs uniformly distributed between 0 and 1. We have found that this type of game provides opportunities for cooperation without making cooperation the only rational choice. Increasing the number of actions beyond this level increases the computational complexity without changing the effect of cooperation because the additional options are not likely to provide a superior outcome for cooperating agents. Reducing the number of actions reduces opportunities for cooperation. We have also empirically explored generating payoffs from other distributions, and have generally found that opportunities for cooperation are equivalent or inferior (Damer and Gini 2008b).

As we said earlier, we assume the agents play a Nash equilibrium. However, the Nash equilibrium is a strictly self-interested approach. It will not recognize opportunities for

cooperation where one of the agents needs to forego a potential benefit in order to provide a larger benefit to its opponent. To allow our agent to cooperate we use a model which has been used to adapt the Nash equilibrium to explain cooperative behavior in humans (Frohlich 1974). We assume agents value their opponent's payoffs in addition to their own. Specifically, each agent adopts an *attitude* towards its opponent, which determines how much weight it attaches to its opponent's payoff in relation to its own payoff.

As in (Damer and Gini 2008a), an attitude is a real number in the range [-1, 1]. An attitude of 1 means that the opponent's payoff is valued as highly as the agent's own payoff. An attitude of 0 means that the agent is indifferent to the opponent's payoff. An attitude of -1 means the agent is only concerned with how well it does compared to its opponent.

Let's call the two agents x and y, and respectively A^x and A^y their attitude. The payoffs for agent x are modified according to

$$P'^{x}_{ij} = P^{x}_{ij} + A^x P^{y}_{ij}$$

where P^{x}_{ij} is the payoff in the original game for player x when actions i and j are chosen, P^{y}_{ij} is the payoff for the opponent, and P'^{x}_{ij} is the payoff of player x in the modified game. The payoffs for agent y are calculated similarly, using y's attitude A^y.

Each agent selects an action which maximizes its score in the modified game, but receives its payoff from the original game. To be able to compute the Nash equilibrium of the modified game an agent needs to know its own payoffs in the modified game and also its opponent's payoffs in the opponent's modified game. This means an agent needs to know the attitude of its opponent.

We have shown (Damer and Gini 2008a) empirically that when both agents have a positive attitude, their payoffs in the original game are higher than if they had both simply tried to maximize their individual scores.

Figure 1 shows the effect of different combinations of attitudes on a player's payoff in the original game. Naturally, the primary determiner of a player's score is the attitude of its opponent, but we can also observe a plateau of cooperation once both players reach an attitude of .2 or .3. When both agents adopt an attitude of 1, they can improve their average payoffs from .80 to .90. Even when they only adopt an attitude of .2 their payoffs improve to .87.

To select its action a player needs to know the attitude of its opponent. It needs the attitude of its opponent to construct the modified game it will use to pick its move. Since the opponent is not motivated to honestly disclose its attitude, the agent needs (1) an estimate of the attitude of its opponent, which we call *belief*. Belief, like attitude, is a real number in the range [-1, 1]; and (2) an estimate of how the opponent selects a Nash equilibrium from the modified game, which we call the *method*. Method is difficult to represent, since any function which maps games to equilibria is a potential value for method. Rather than attempt to represent that space, we use the initial parameter passed to the Lemke-Howson algorithm (the algorithm we use to calculate Nash equilibria). This allows us to represent method as

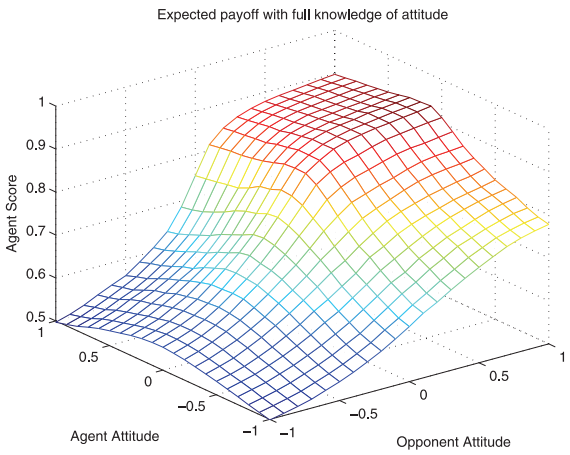

Figure 1: Payoff is affected by the attitudes of the agent and its opponent. The agent's attitude is on the left axis, going from full cooperation (1) to full selfishness (-1). The opponent attitude is on right axis, going from selfishness (-1) to cooperation (1). The payoff is plotted on the vertical axis. Results are aggregated over 1000 games with 16 actions and with payoffs drawn from a uniform distribution between 0 and 1. Results for a particular game may be quite different.

an integer.

Once those factors are known, an agent can set its own attitude to be equal to its opponent's attitude plus a reciprocation level. Hence, $A^x = B^x + R$, where B^x is the belief of agent x, i.e. its estimate of the attitude of A^y, and R is the reciprocation level that agent x has chosen. The reciprocation level can be quite low and still produce cooperation in self-play. In our experiments we have used a reciprocation level of .1. If the opponent is not cooperative this does not lead to a significant loss for the agent, but if the opponent reciprocates in a similar way this will eventually lead to full cooperation.

Learning

In every round the agent observes the nature of the interaction (the game) and the choice made by the opponent in that context (the action). From that information, it needs to update its probability distribution over the attitude, belief, and method of the opponent.

We focus on playing against a stationary opponent (one that does not change its attitude, belief, and method). This is common for algorithms that learn in repeated games (Conitzer and Sandholm 2007) and it is impossible to optimize an agent against an arbitrarily complex opponent (Powers, Shoham, and Vu 2007).

Due to the complex interactions between attitude, belief, method, and the game being played there is no probability distribution that can be updated analytically for each observation, but for a given value of attitude, belief, and method we can calculate the probability that the agent would select a particular action in a given game. This information makes

it fairly straightforward to use a particle filter to learn values for attitude, belief, and method.

Instead of representing a probability distribution parametrically, a particle filter represents it with a number of samples drawn from it. Each particle has a weight attached, and the distribution represented by the particles is a discrete distribution with probability of each particle proportional to its weight. When an observation is made, each particle's weight is updated by multiplying it by the probability assigned to the observation by that particle.

As observations are made, the relative probability of the particles changes. As the weights attached to the particles become more unbalanced, the distribution represented by the particle filter becomes simpler. At the extreme, if one particle has all the weight, the distribution is effectively represented by a single particle. To avoid this, when the effective number of particles drops below a threshold, a new set of particles are drawn by sampling particles from the existing distribution and adding noise.

Regularization

Particle filters were originally developed to learn in environments which changed stochastically over time. Our environment does not change since we currently assume the opponent does not change its attitude, belief, and method and without a model of particle motion, particle filters have problems converging. The problem arises because a discrete distribution does not increase the effective number of particles during the resampling step if the particles remain in the same position. There is no functional difference between 1 particle with a weight of 50, and 50 particles with a weight of 1 all in the same place. If the particles move randomly, then those 50 particles will spread out, but if they don't move, then resampling does not increase the effective diversity. A regularized particle filter (Musso, Oudjane, and Legland 2001) avoids that problem by resampling from a continuous distribution instead of a discrete distribution. By adding noise to particles drawn during the resampling process, we avoid having all the samples drawn from a single particle being identical. The optimal level of noise can be estimated by observing the variance of the current particle set. It is a Gaussian distribution with 0 mean and standard deviation equal to $N^{-1/6}$ times the standard deviation of the particle set.

An additional complication in our case is that while some of the data (attitude and belief) are continuous and can be perturbed with Gaussian noise, some (method) are discrete. Learning the method the opponent uses is complicated. In theory, method could be any function which maps games to probability distributions over actions, but this is such a large space that it would be impossible to learn it. We restrict method to one of the Nash equilibria of the modified game. This reduces the space of possible equilibria, but it means that there is no distance measure between different methods. Therefore the only perturbation we can apply to methods is to change them to a random method with some probability. We find the optimal probability using a technique called Leave-One-Out. We select the probability that gives the highest likelihood of resampling the current distribution of method values from a distribution created by removing one particle from the current set. We approximate this value by testing 100 values evenly distributed over the range of possible values and using the one which gives the highest likelihood.

We start with 400 particles with attitude and belief drawn from a Gaussian distribution centered at 0 with the identity matrix as a covariance matrix, and method drawn from a uniform distribution over the list of methods under consideration. We assign each particle a weight of .0025. If the effective number of particles goes below 200 we resample.

Algorithm 1 $ParticleFilter$

Generate initial set P of N particles from prior belief about the values of attitude, belief, and method
Assign each particle a weight equal to $\frac{1}{N}$
while presented with data **do**
  Observe the game G and the opponent's move M
  Compute the effective number of particles $N_{eff} = 1/[\sum_{p \in P} p^2_{weight}]$
  if $N_{eff} >$ threshold **then**
    for $p \in$ P **do**
      p_{prob} = probability of opponent's move M in game G given p_{att}, p_{bel}, and p_{method}
      p_{weight}=$p_{weight} * p_{prob}$
    end for
  else
    Compute standard deviation Std_{att} of p_{att}
    Compute standard deviation Std_{bel} of p_{bel}
    $h = N^{-1/6}$
    Compute optimal perturbation probability pp for p_{method}
    while accepted particles $<$ total particles **do**
      Select a new particle from current particles with probability proportional to p_{weight}
      Update attitude and belief with Gaussian noise
      $p'_{att} = p_{att} + h * N(0, Std_{att})$
      $p'_{bel} = p_{bel} + h * N(0, Std_{bel})$
      $p'_{method} = p_{method}$
      with probability pp, p'_{method} = random method
      p'_{prob}= probability of M in G given p'_{att}, p'_{bel}, and p'_{method}
      Accept p' with probability p'_{prob}
    end while
  end if
end while

Performance

Figure 2 shows the error of our learning algorithm's estimate of attitude, belief, and method. Note that the error in method (the probability that the learner has not found the correct method) eventually levels off around .12, which means 12 percent of the time the particle filter fails to learn the opponent's method. This is because for some values of attitude and believe (-1,-1 for instance) method does not play a role in the move selected by the agent. In those situations it is not possible to learn the opponent's method. It is also worth not-

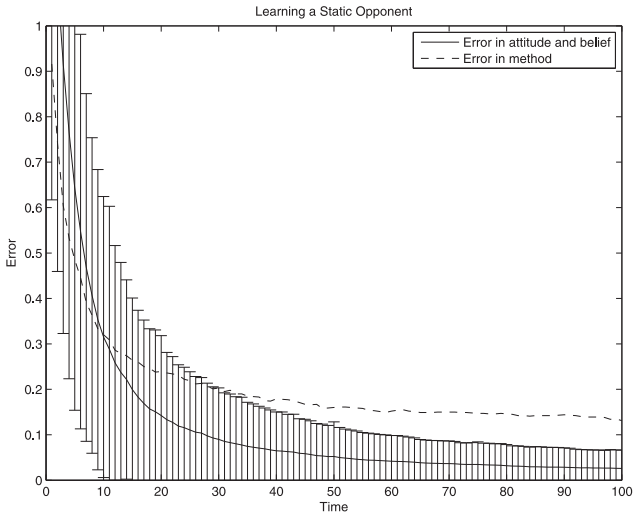

Figure 2: Accuracy of learning algorithm against a random static opponent. Results aggregated over 100 sequences of 100 games. Learning targets drawn from a Gaussian with 0 mean. Time is the number of games played. Error bars at ± 1 standard deviation

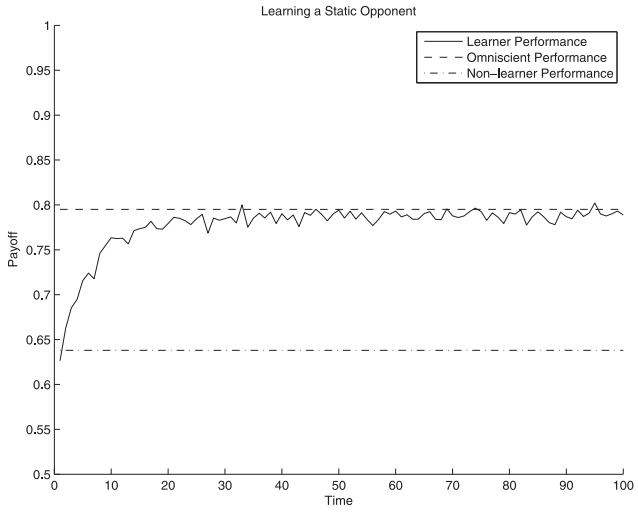

Figure 3: Performance of learning algorithm against a random static opponent. Results aggregated over 100 sequences of 100 games. Learning targets drawn from a Gaussian distribution with 0 mean. Time is the number of games played.

ing that the variance in the error for the estimate of attitude and belief (the Euclidean distance between the estimated attitude and belief and the true attitude and belief) remains high. For some values of attitude and belief (1, 1 for instance) small changes in attitude and belief do not result in changes in behavior. Therefore the accuracy achieved can vary significantly with the value of the target being learned.

Figure 3 shows that the algorithm can perform well against a random static opponent, even though it may not succeed in learning the exact parameters of the model used by that opponent. The agent uses the particle filter to predict a distribution over the actions of its opponent, and chooses a move which is the best response to that distribution. Omniscient performance is what would be achieved by an agent already aware of the true attitude, belief, and method of the opponent. Non-learner performance is what is achieved by an agent which plays according to its prior distribution over the opponent. After 10-20 interactions the agent is performing close to the theoretical maximum against its opponent.

A general concern is the learning speed. In human interactions people typically can discern the intentions of others from a few interactions. In our case, because the interactions are very complex it takes 20 or more games before a conclusion about the opponent can be made. This is still a small number when compared to the number of games that typically are played to learn in repeated games (Powers, Shoham, and Vu 2007).

Results

We have identified a number of parameters that affect the games and modified them to assess the robustness of our learning algorithm.

Distribution of payoffs

Generating payoffs randomly is a logical way to generate many different normal form games but then the question arises of how to pick the random distribution. We generally use a uniform distribution, but we have verified that the algorithm works for other distributions. Figure 4 shows the effect of the following distributions on the speed of learning:

Gaussian. Payoffs are drawn from a Gaussian distribution. In this case the agent's actions have strongly interacting effects, but with more outliers.

Gaussian with higher standard deviation. Payoffs are drawn from a Gaussian distribution with a standard deviation 10 times as high as the standard deviation of the uniform distribution we use.

Strongly Move Correlated. Instead of drawing payoffs for each combination of actions, payoffs are drawn from a uniform distribution for each action individually. The payoffs for each combination of actions are found by summing the payoffs of each individual action. This means that the results of a player's action depend only on the action it chose, and not on the action the other player chose.

Weakly Move Correlated. This is a combination of a strongly correlated game, and a uniform distribution. The agents choices interact, but there is also an independent factor.

Figure 4 shows the effect of different distributions on the speed of learning. The top two graphs show that a Gaussian distribution, regardless of its variance, doesn't have a significant effect on the speed of learning. The bottom two graphs show that when payoffs are associated with individual actions instead of combinations of actions, learning can be affected. When an agent doesn't need to take its opponent's action into account when choosing its action, the only

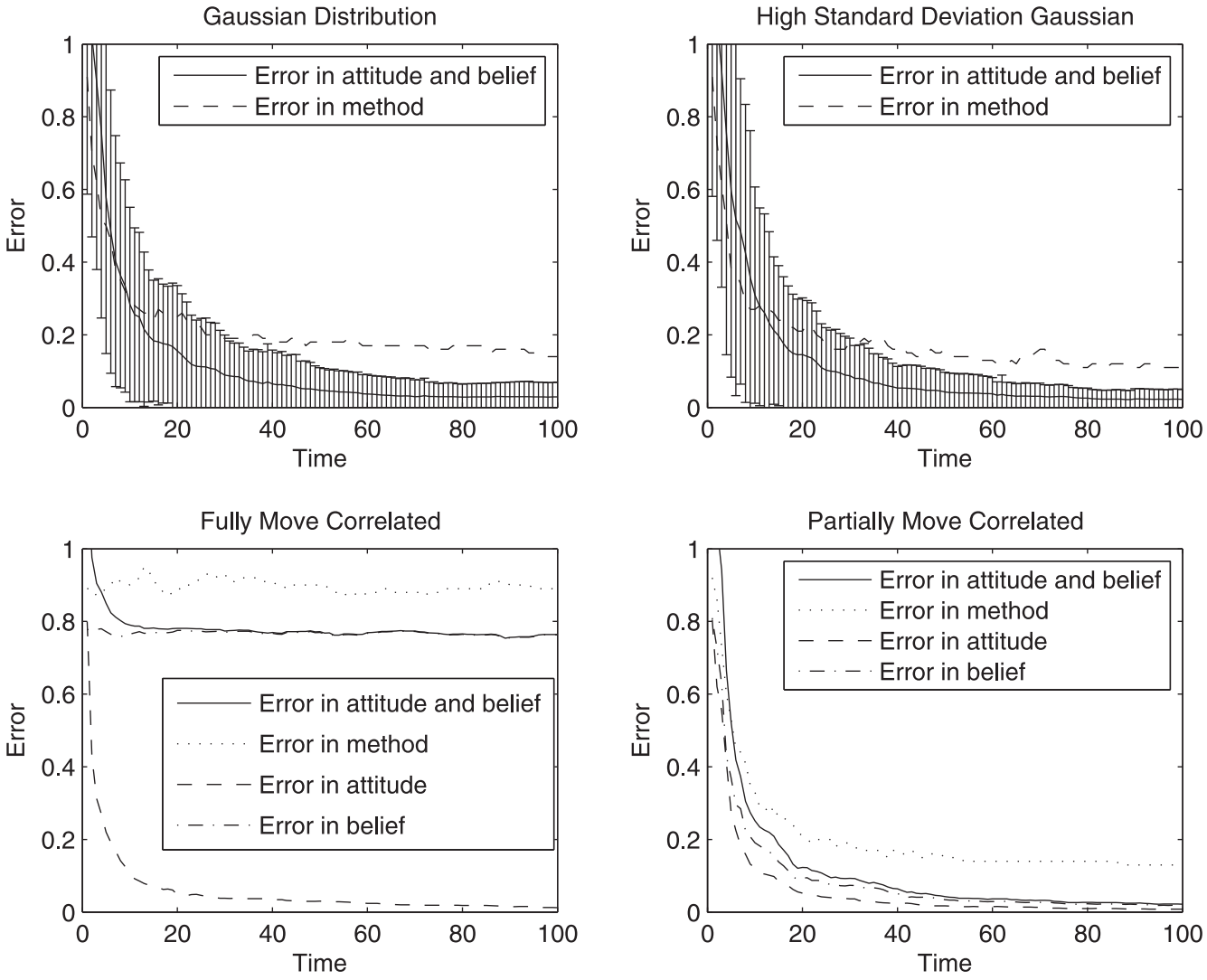

Figure 4: How learning is affected by the distribution from which payoffs are drawn. Results aggregated over 100 sequences of games. Learning targets drawn from a 0 mean Gaussian. Time is the number of games played.

thing that affects its choice is its own attitude - belief and method are irrelevant. Therefore the agent cannot learn belief and method. If we look only at the accuracy of the attitude estimate, the particle filter converges very quickly. This is a logical consequence of the removal of the noise added by the need to take belief and method into account. The final graph shows that even when payoffs are only partially correlated to the particular combination of actions, this provides enough information to learn the opponents belief and method.

Complexity

The number of actions in each game affects the performance of the particle filter. Figure 5 shows the performance in games with different numbers of actions. As the number of actions increases, the complexity of the game increases, which increases the time needed to analyze the game. At 32 moves, the time taken to learn has become prohibitively expensive, which implies that this technique is not suitable for cases with a large number of distinct moves. Note that in games with 32 moves, even though the learner is able to achieve a high level of model accuracy (the Euclidean distance between its estimate of the attitude and belief of the opponent and the true attitude and belief of the opponent), its prediction accuracy (the Jensen-Shannon divergence between its prediction of the opponent's action and the actual probability distribution the opponent used to select an action) suffers due to the unpredictability of the environment. In contrast, with 2 moves, the prediction accuracy is very good, but the model accuracy is poor.

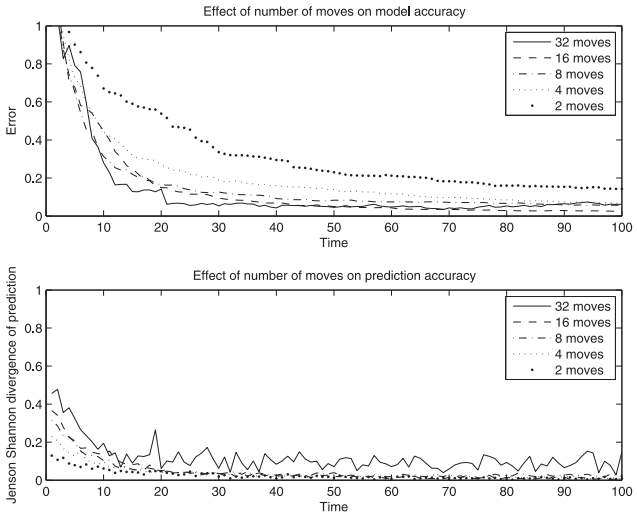

Figure 5: Performance is affected by the number of actions available in the game. Results aggregated over 100 sequences of 100 games except for the 32 move case which is only aggregated over 10 sequences. Learning targets were drawn from a 0 mean Gaussian. Time is the number of games played.

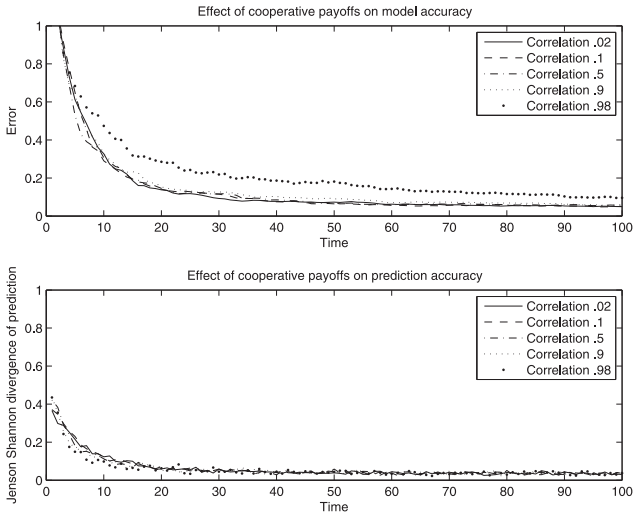

Figure 6: Effect on performance when agents payoffs are positively correlated. Results aggregated over 100 sequences of 100 games. Learning targets drawn from a 0 mean Gaussian. Time is the number of games played.

Conflict and Cooperation

The environment can have a significant effect on the ability of agents to cooperate. Agents in a zero-sum game have no opportunity to cooperate whatsoever, since any gain for one agent is an equivalent loss for the other. Agents with identical payoffs have no opportunity for conflict, since their interests are identical. Learning in those environments is impossible, since a change in attitude doesn't change behavior. However, we can explore how less extreme environmental influences affect the ability of our agent to cooperate.

We characterize the degree of conflict in terms of the correlation between agents payoffs. In a zero sum or constant sum game, the correlation is -1. When agents payoffs are identical the correlation is 1. We create games with correlated payoffs by generating payoffs as a sum of two uniformly distributed variables, one of which is shared by both agents. We vary the degree of correlation by changing the relative magnitude of the variables.

Figure 6 shows the effect of positive correlation between agents scores. The degree of correlation doesn't significantly effect the prediction accuracy, but with a high degree of correlation it becomes more difficult to learn the model parameters (attitude and belief).

Figure 7 shows the effect of negative correlation between agent scores. Again, prediction accuracy is good, but when scores are highly negatively correlated it is more difficult to learn the model parameters. This effect is more pronounced than when the scores are positively correlated.

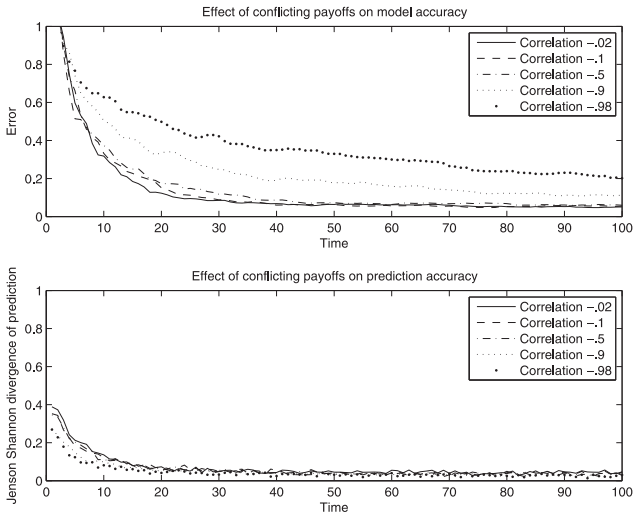

Figure 7: Effect on performance when agents payoffs are negatively correlated. Results aggregated over 100 sequences of 100 games. Learning targets drawn from a 0 mean Gaussian. Time is the number of games played.

Self Play

We have explored the effect of using our learning algorithm in self-play. Each agent attempts to learn the attitude, belief, and method of its opponent, but it no longer chooses a best response. Instead, it selects a move using its own attitude, the attitude it learned for its opponent, and the method it learned for its opponent. It chooses an attitude .1 higher than the attitude it believes its opponent is using, with a minimum of 0 and a maximum of 1. This policy allows agents to achieve cooperation without taking a big risk if the opponent doesn't cooperate. Since the opponent is no longer stationary, we have added a simple model of particle motion for this test: an agent's attitude and belief are assumed to

drift according to a Gaussian distribution with 0 mean and .1 standard deviation. This does not accurately reflect the actual changes in the opponent's attitude and belief, but it is sufficient to allow an agent to track the attitude and belief of a learning opponent.

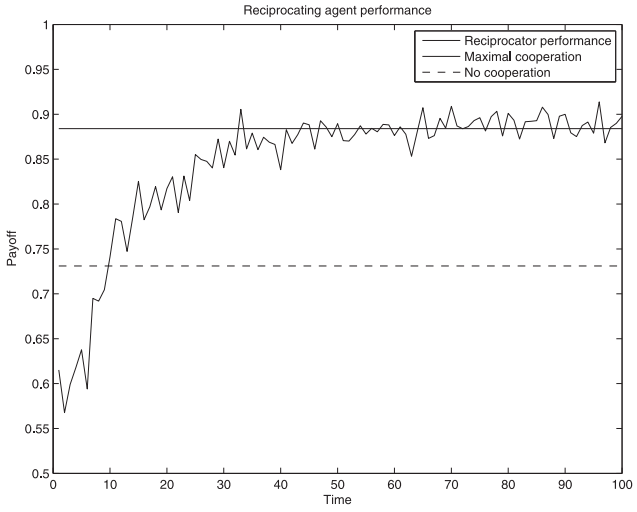

Figure 8: Performance of two learning agents, each reciprocating the attitude of its opponent with a small bonus. Results aggregated over 10 sequences of 100 games. Time is the number of games played.

Figure 8 shows the performance achieved by two reciprocating learning agents. Cooperation takes about 30 iterations to achieve, but eventually reaches performance roughly equivalent to two fully cooperative agents. The primary factor in the time taken to achieve cooperation is the magnitude of the reciprocation agents practice. We use .1, but with a higher value full cooperation would occur more rapidly.

Related Work

Typically actions in games are labeled as cooperate/defect. Labels affect how human participants play, producing more cooperation compared to situations where there are no labels (Zhong, Loewenstein, and Murnighan 2007). We do not label actions since the games are generated randomly and we want the agents to be able to decide on their own which actions are cooperative.

Our model of cooperation is based on models developed to explain human cooperation in normal form games. (Valavanis 1958) proposed the modification of a normal form game to reflect an agent's preferences over its opponent's utility. (Frohlich 1974) pointed out that this can lead to an ill-defined utility function, and proposed restricting an agent's preferences to its opponent's consumption instead of its opponent's utility. (Fitzgerald 1975) introduces a utility which is linear in the opponent's payoff, and points out that positive attitudes will not necessarily reduce the level of contention between agents. With attitude values above 1 the game can become a contest to make your opponent accept a higher payoff. It is for this reason that we have chosen attitudes between -1 and 1. There has been a lot of research on human behavior in the context of game theory. (Bolton 1998) provides an overview of work on bargaining and dilemma games, which are the games most concerned with cooperative behavior. (Camerer 1997) provides a broad overview of the many different ways in which human behavior does not conform to game theoretic predictions. We use a simple model of linear altruism towards the opponent's payoffs, which is sufficient to provide a basis for cooperative behavior.

Efforts have also been made to describe human behavior without developing a specific model. (Altman, Bercovici-Boden, and Tennenholtz 2006) shows that people's behavior in one game can be used to predict their behavior in different games. Since the games they used could all be described as cooperative games, their prediction success would seem to indicate that cooperation is a valid abstraction to use.

Reciprocation is an effective way to motivate an opponent to cooperate. (Axelrod 1984) describes a tournament among agents of repeated Prisoner's Dilemma. Tit-for-Tat was the most effective strategy in that tournament - it is a reciprocating strategy which simply copies the move chosen by the opponent in the previous round. Research on learning for agents which play normal form games has focused on repeated play of a single game against a stationary opponent with the goal of finding either an equilibrium or a Pareto-optimal outcome in self-play. (Fudenberg and Levine 1998) provides a good overview of fictitious play, which explores the effects when agents attempt to learn their opponents actions and then choose the best response.

(Littman 2001) describes friend or foe Q-learning, which is capable of dealing with hostile and cooperative opponents, finding a minimax solution in the case of a hostile opponent and a cooperative solution with a friendly opponent. However, it is not capable of detecting a hostile opponent on its own – it needs to be told which approach to use. (Crandall and Goodrich 2005) offers another modification of reinforcement learning which detects cooperation by examining its performance. This approach is provably not exploitable, and is successful in achieving cooperation in self play. (Powers, Shoham, and Vu 2007) describe an algorithm which learns against stationary opponents in repeated games. The algorithm can be extended to non stationary opponents by limiting the history the opponent can use. Learning requires playing thousands of games. AWESOME (Conitzer and Sandholm 2007) is the first algorithm guaranteed to learn to play optimally against stationary opponents and to converge to a Nash equilibrium in self play. It also learns to play optimally against opponents that eventually become stationary. To guarantee convergence in self-play, it assume all agents play the same Nash equilibrium.

Particle filters have multiple uses in multiagent settings for opponent modeling, including (Doshi and Gmytrasiewicz 2009) who use particle filters to compute approximate policies for finitely nested POMDPs to map agents' beliefs to policies, and (Bard and Bowling 2007) who model opponents in Kuhn poker.

Conclusions

In this paper we have described an algorithm for an agent to learn to cooperate when playing a sequence of different normal form games with the same opponent. We have shown that achieving cooperation is beneficial and that learning how to respond to the opponent is possible. We have tested the algorithm in many situations and found that it is fairly robust and effective. Our approach provides a basis for using reciprocation to cooperate in environments where cooperative behavior is not immediately evident.

We have focused on learning against a stationary target and in self-play. Next we will explore two related questions. Firstly, given an estimate of attitude, belief, and method, how should an agent act to cooperate effectively? Secondly, how should the agent learn if the opponent is not stationary?

References

Altman, A.; Bercovici-Boden, A.; and Tennenholtz, M. 2006. Learning in one-shot strategic form games. In *Proc. European Conf. on Machine Learning*, 6–17. Springer.

Axelrod, R. M. 1984. *The evolution of cooperation*. Basic Books.

Bard, N., and Bowling, M. 2007. Particle filtering for dynamic agent modelling in simplified poker. In *Proc. of the Nat'l Conf. on Artificial Intelligence*, 515–521.

Bolton, G. E. 1998. Bargaining and dilemma games: From laboratory data towards theoretical synthesis. *Experimental Economics* 1:257–281.

Camerer, C. F. 1997. Progress in behavioral game theory. *The Journal of Economic Perspectives* 11(4):167–188.

Conitzer, V., and Sandholm, T. 2007. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning* 67(1–2):23–43.

Crandall, J. W., and Goodrich, M. A. 2005. Learning to compete, compromise, and cooperate in repeated general-sum games. In *Proc. of the Int'l Conf. on Machine Learning*, 161–168. New York, NY, USA: ACM.

Damer, S., and Gini, M. 2008a. Achieving cooperation in a minimally constrained environment. In *Proc. of the Nat'l Conf. on Artificial Intelligence*, 57–62.

Damer, S., and Gini, M. 2008b. A minimally constrained environment for the study of cooperation. Technical Report 08-013, University of Minnesota, Dept of Computer Science and Engineering, Minneapolis, Minnesota.

Doshi, P., and Gmytrasiewicz, P. J. 2009. Monte Carlo sampling methods for approximating interactive POMDPs. *Journal of Artificial Intelligence Research* 34:297–337.

Fitzgerald, B. D. 1975. Self-interest or altruism. *Journal of Conflict Resolution* 19:462–479.

Frohlich, N. 1974. Self-Interest or Altruism, What Difference? *Journal of Conflict Resolution* 18(1):55.

Fudenberg, D., and Levine, D. K. 1998. *The Theory of Learning in Games*. MIT Press.

Littman, M. L. 2001. Friend-or-foe Q-learning in general-sum games. In *Proc. 18th International Conf. on Machine Learning*, 322–328. Morgan Kaufmann.

Musso, C.; Oudjane, N.; and Legland, F. 2001. Improving regularized particle filters. In Doucet, A.; de Freitas, N.; and Gordon, N., eds., *Sequential Monte Carlo Methods in Practice. New York*, number 12, 247–271. Statistics for Engineering and Information Science.

Powers, R.; Shoham, Y.; and Vu, T. 2007. A general criterion and an algorithmic framework for learning in multi-agent systems. *Machine Learning* 67(1–2):45–76.

Rapoport, A., and Chammah, A. 1970. *Prisoner's dilemma: A study in conflict and cooperation*. University of Michigan Press.

Valavanis, S. 1958. The resolution of conflict when utilities interact. *The Journal of Conflict Resolution* 2(2):156–169.

Zhong, C.-B.; Loewenstein, J.; and Murnighan, J. K. 2007. Speaking the same language: The cooperative effects of labeling in the prisoner's dilemma. *Journal of Conflict Resolution* 51(3):431–456.