

Strategy Inference in Stochastic Games Using Belief Networks

D. Michael Franklin

Kennesaw State University
Marietta, GA, USA

Abstract

In many gaming and real-world scenarios players try to predict the behavior of the other players. This assumes some underlying strategies that players follow, that they can be inferred, and that a reasonable player can counter them in real-time. This paper seeks to formulate a definition for strategies and their relationship with policies, determine the viability of inferring strategies, and formulate counter-strategies in real-time. If successful, players utilizing dominant strategies should defeat those using weaker strategies, the weaker strategies should be shifted to a better policy within that strategy, and this should result in improved performance. We will show that strategies offer significant performance enhancement, strategies can be recognized in real-time, and AI's utilizing strategy inference will outperform their originally superior opponents.

Introduction

In well-formed machine learning algorithms we seek to provide a mapping of which action to take in a given state. This mapping is a policy, $\pi : s \in S \rightarrow a' \in A$. Such policies are effective, but take many iterations to learn often requiring thousands of trials to learn behaviors, and exponentially more to learn complex behaviors. We introduce strategies to simplify learning by breaking down complex behaviors into several smaller policies rather than one monolithic policy. We wish to imbue these artificially intelligent agents with strategies so that they can learn and react quickly. Once learned and placed into strategies, the strategy can choose a policy to implement. The strategy monitors its performance and evaluates itself in comparison to other policies. If performing poorly, the strategy then switches to a better policy. This requires that the strategy can heuristically determine the value of a policy over time. The strategy will also compare the actions of the other intelligent agent to known policies within the set of strategies. Thus the AI can form a belief network about which policy it believes the opposing AI is using. By monitoring the policies that the opposing agents use the AI can then hypothesize about the opponent's strategy. The AI can then select the optimal policy or counter-strategy. This allows for improved performance as the game

progresses. The foundation work for these strategies, underlying policies, and the inference engine are detailed in their respective sections. We will use the games of Roshambo and RPS-LS (Rock, Paper, Scissors, Lizard, Spock) to implement these policies and strategies and to compare the performance of the AI both with and without strategy inference.

Related Works

Bowling and Veloso (2001, 2004, 2009) lay a foundation for stochastic games with multiple agents but stop short of true multi-agent strategic considerations. Kim, Cho, and Lee (1995) offer seminal insight into vectorization of features for policy extraction (used in this research). Fisher (2008) explores the application of game theory to RPS extensively. Cheng (2004) also offers insight into policy formation and reasoning. We build on these and explore multi-agent strategy, not just policies. Finally, Bersma and Spronck (2008) explore strategy in two-player turn-based games. We wish to extend this into multi-agent and non-TBS games.

Methodology

There are several steps required. First, strategy must be represented and modeled. Second, strategy can be recognized by another agent or team of agents. Third, strategy can be compared with another strategy and adjusted in real-time.

A strategy is comprised of policies that have defined states and actions (s, a) . A policy, π , is a mapping from $s \rightarrow a$, or that chooses a' , the next action, from A optimally for all $s \in S$ (1, generically). A valuation function evaluates the results of taking actions from a given state, and thus determines the optimal action. This function (2) generally determines a value by capturing the reward of an action - in Markovian or Non-Markovian fashion. While the valuation function, V^* , is left generalized here, it is defined specifically during implementation. This function is critical to the success of the policy and essential for it to have decidability.

Additionally, *w.r.t* time, the reward can be diminished by a certain factor that grows *w.r.t* time (i.e., the cumulative value is weighted in proportion to time). In a fully-explored deterministic environment choosing a' is distinct and simple. Each state s has an optimal action a mapping to the highest value reward. In non-deterministic environments these evaluations must use underlying probabilities or experimental values (3). When time constraints or complexity

create bounds then a heuristic must be used. These heuristics may range from scores or values to expected values and rewards. In this environment a' is determined by choosing the maximum value of the available choices (4). Here $E(a, s)$ is the expected value based on experience thus far.

$$\pi : \operatorname{argmax}_a V^a(s), \forall s \in S \rightarrow a' \quad (1)$$

$$\text{where} : V^*(s_t) = \max_a V^a(s_t) \quad (2)$$

$$\pi : \operatorname{argmax}_{a \in A} P(s|a) V^a(s_t), \forall s \in S \rightarrow \hat{a}' \quad (3)$$

$$\pi : \operatorname{argmax}_{a \in A} E(s, a), \forall s \in S \rightarrow \hat{a}' \quad (4)$$

These values can be determined through machine learning, experience, or domain knowledge. The agent may start by favoring exploration (i.e., favoring stochastic selection) then transition to exploitation (i.e., taking the best actions learned). Once these most-favorable choices (*w.r.t.* reward or utility) have been learned, they can be saved and re-used, forming a policy that the agent can start from without re-learning. Once learned it can be added to the set of policies Π to form a collection of policies. The agent or set of agents can then share this collection of policies as prior experience for each of them. Selecting from among these policies is the domain of strategy.

A strategy, σ , selects the optimal (or desired) policy, π , or set of policies, Π , to accomplish a goal, g , or meta-set of goals, G (5). All σ are in Σ , the set of all strategies. As seen generically in (5), this selection is putting in force the best policy. The method for such selection, however, is not so easily contrived. There is a similar notion of expectation that will be used. This expectation function will examine the actions $a \in A$ taken by the current policy π in force and those of each other known policy $\pi \in \Pi$. The *argmax* of these values is then taken, and this may indicate that either the current policy π is the best policy to have in place or there is a needed shift in policy. Further, a threshold ϵ is considered before a policy change occurs in order to provide some momentum to the current policy and avoid unnecessary vacillations in policy change (6). If the δ between the current π and π' is less than ϵ , the current policy, π , stays in place; otherwise, a change is initiated. This mechanism and relevant valuations are considered within the experiments.

$$\sigma : \operatorname{argmax}_{\pi} V^{\pi_a}(s_t), \forall \pi \in \sigma \rightarrow \pi' \quad (5)$$

$$\sigma : \operatorname{argmax}_{\pi} E^{\pi}(s, a) - \epsilon, \forall \pi \in \sigma \rightarrow \pi' \quad (6)$$

Strategy Representation

There are many forms to represent σ . In Roshambo the strategy can be viewed as a set of policies, each with weighted views of the previous actions of the focus agent, the actions of the other observable agents, and the predicted future actions of all agents. Each player chooses rock (a_r), paper (a_p), or scissors (a_s). The other player chooses similarly. Eq. 7 shows that σ_m selects the best policy (Eq. 8) from σ

by choosing the policy that produces the maximum value (by Eq. 9). This policy leads to the optimal choice of the next action (Eq. 10). If the current policy is not producing optimal results, the strategy can shift policies via Eq. 7. The policy is made up of all relevant considerations of moves and time. (Eq. 9). Here LB and LA represent the number of steps to look ahead and look back, respectively, a the action, with r the reward and E the expected reward, and w , the weights for each.

$$\sigma_m : \max_{\pi} V^{\pi_n}(s_t), \forall \pi \in \sigma \rightarrow \pi_n' \quad (7)$$

$$\pi : f_{\pi}(LB, w_{LB_k}, LA, w_{LA_l}, w_r, w_p, w_s, E_r, E_p, E_s) \quad (8)$$

$$V^{\pi_n} = \max_a \left[\left(\sum_{k=0}^{LB} w_{LB_k} r_{a_{t-k}} \right) + \left(V^a(\max_a(w_a r_a)) \right) + \left(\sum_{l=0}^{LA} w_{LA_l} E(a_{t+l}) \right) \right] \quad (9)$$

$$\pi_n' \rightarrow a' \quad (10)$$

The first elements are the individual likelihoods, L_x , where x is the particular move rock, paper, or scissors. These likelihoods are the weighted and discounted values of choosing x by looking backwards LB moves and predicting forward LA moves (Eq. 11, $x \in \{R, P, S, L, K\}$). In these formulae the individual likelihoods, L_x , are cumulative values of rewards received from the past moves and future moves with weighting (per π). Thus the policy customizes these formulae to suit the strategy σ .

$$L_x = \left[\left(\sum_{k=1}^{LB} \frac{1}{k} \alpha_{LB} w_{LB_x} r_{a_{t-k}} \right) + \left(\sum_{k=1}^{LA} \frac{1}{k} \alpha_{LA} w_{LA_x} E(a_{t+k}) \right) \right] \quad (11)$$

The proportionate probability of choosing each move is derived by considering each with respect to the total (Eq. 12).

$$P_x = \frac{L_x}{(L_r + L_p + L_s)} \quad (12)$$

Then a' is determined by the maximum value from the inverse weighted proportions (Eq. 13).

$$a' = \max_a ((1 - P_{a_r})w_r, (1 - P_{a_p})w_p, (1 - P_{a_s})w_s) \quad (13)$$

Thus, the aggregate probabilities, based on the past and future selections of that same move and tempered by a learning rate, α , form the probabilities for each move. Anytime

reward is considered for future moves under uncertainty (i.e., in the absence of a model or an approximate formed through experience) it can be replaced with an expectation of reward, as shown in Reinforcement Learning (RL). The resultant move calculation (Eq. 13) could produce an ordered vector of moves with their probabilities. This ordered vector would rank all moves, above some threshold θ , to resolve a' . This θ provides the ability to have a default override in the cases where there is no clear cut 'winner' for the best move. This enhances the strength of the policy as it allows for probabilistic drive when the choices are clear, but an underlying set of choices to rule the behavior otherwise.

The strategy can also, for the purpose of a move based game like Roshambo, be represented in policies that contain the weights of each move and the α for each. The weights could be proportioned to favor irregularity (avoid repetition), favor regularity (repeat favorite moves), or follow predetermined patterns. The formulation for the influence of α can alter Eq. 11 by step k .

In the experiments the Roshambo models were developed. As indicated in Eq. 8, the strategy contains policies (a vector of the various features). In this iteration of the game there are equal rewards for each move. The two player game was run with two independent policies and compared. These experiments and their results are found in the next section.

The addition of a higher level strategy is put in place to help select from among these policies. For example, if the 'LB3Rock' policy was performing poorly, the strategy in place, σ_1 , may select another policy, say 'LB3Paper'. In this manner, the strategy is monitoring performance and making a decision about which policy should be in place. The more information the strategy has the better this selection will be. As history is accumulated, and the strategies of the other player are considered, the selection of the best policy from the set can happen more quickly and accurately which will be shown in the experiments.

Experiments

To prove the viability of strategic modeling, the Roshambo experiments were implemented to establish a baseline of the gameplay for comparison, implement policies to measure their impact, and create and test strategies that utilize these policies. Next, the goal was to test strategy inference, policy switching, and evaluate this shifting. The games has two players, 10,000 rounds, using different strategies. Strategies were tested against each other in sets (e.g., RockPref vs. ScissorPref).

The results are shown in Table 1. The normal vs. normal games showed the expected distribution where any one player wins about 33% of the time. It is also noteworthy to consider the not-lose percentage (non-negative outcome). For each experiment with normal vs. normal the not-lose percentage was around 67%, as expected from standard probability distribution. In LB3RockBias vs. LB3PaperBias, paper-bias won 43.78% with a not-lose percentage of 71.74%. Other match-ups show similar results. These comparisons reveal that mismatched strategies show an improvement over random strategies and that the strategy models are

functional. One challenge is that with only three choices, random guessing will succeed about 33% of the time.

Game	Bias	Player1 Win	Player1 Not Lose
RPS	N vs. N	33.29%	66.33%
RPS	P vs. R	43.78%	71.74%
RPSLS	N vs. N	40.13%	60.22%
RPSLS	P vs. R	62.43%	87.47%

Table 1: Results of RPS and RPSLS with Strategies

To counter this the number of choices was increased from 3 to 5, shifting the game from Roshambo to Rock, Paper, Scissor, Lizard, Spock (RPSLS) where there are two ways to win and two ways to lose (and one way to tie). As such, the results of random guessing should be 40% wins, 40% losses, and 20% ties based on standard probability. The results, shown in Table 1, show that the experiments supported the probability hypothesis with the wins for the normal play being 40.13% with a not-lose of 60.22%. Updated strategies were used, such as RockBias vs. PaperBias, with policies LB3RockBias vs. LB3PaperBias, respectively. This showed a significant improvement in win percentages (from 43.78% with three choices to 62.43% with five) and in not-lose percentages (from 71.74% to 87.47%). This shows that having more choices allows for strategy to be more fully implemented and results in significant differences in strategy vs. non-strategy. This is further evidence of the veracity of the strategy model presented herein. It is also leads to a follow-up research question of understanding the minimal point at which strategy is applicable. The evidence suggests that the influence of strategy use is visible only beyond a point of complexity wherein those agents following a strategy routinely outperform agents following a base policy, like random guessing. In this experiment, the number of choices was indicative of such a threshold. This claim will be verified via additional research.

The larger claim of strategy inference needed to be tested next. To this end, an inferior strategy was chosen for the target player and a superior strategy given to the other player. This set up the target player to lose initially. The target player should then cycle through its policies within its strategy to select the best performing policy. This is the first result that was desired to be shown, that the player could find the optimal policy from within its current strategy. Next, the goal was expanded to allow for a meta-strategy that could choose from among the available strategies which will perform best.

To infer the most likely strategy that the opponent is using the agent examines each action taken by that opponent. By creating a mapping of possible moves predicted by each strategy and the actual move taken by the opponent a belief network is created. The nodes in this network accumulate votes as the game progresses. In this context, a vote is a match between the predicted move for a given strategy and the actual move taken by the opponent. As play continues the most likely strategy emerges. This tally can then be used to select a proper counter-strategy for the most likely opponent strategy in place. Table 2 shows the various trials of the

Opponent Strategy	Initial Player Strategy	Recognition Steps 1	2	3	Average
RockBias	ScissorBias	18	22	12	17.3
ScissorBias	PaperBias	9	8	9	8.3
PaperBias	RockBias	15	6	19	13.3
	Average	14	12	13.3	12.97

Table 2: Strategy Inference Results

Opponent Strategy	Initial Player Strategy	Without Inference			With Inference		
		Opponent	Player	Ties	Opponent	Player	Ties
RockBias	ScissorBias	4844	3415	1741	2584	6109	1307
RockBias	ScissorBias	4808	3466	1726	2595	6002	1403
RockBias	ScissorBias	4815	3444	1741	2525	6121	1354
	Average	4822.3	3441.7	1736	2568	6077.3	1354.7

Table 3: Counter Strategy Results

experiment with the prediction accuracy.

Table 2 also shows the initial strategies for the opponent and the player. Initial strategies were chosen in a biased fashion to create a disadvantage for the Strategy Inference engine. The third, fourth, and fifth column show the total number of rounds that passed before the Strategy Inference engine correctly recognized the strategy the opponent was using. Three trials were done with each of the selected disadvantaged strategies (shown in each column) and then averaged. The overall average is 12.97 rounds. In less than 13 rounds, on average, the agent was able to correctly select the opponents strategy. This confirmed that the Strategy Inference engine was able to correctly select the strategy in play from the complete list of available strategies ($n = 21$). This is not the same as determining a strategy in general without any foreknowledge, admittedly, but it is foundational to the concept of Strategy Inference.

To see how the ability to determine the opponent’s strategy through inference would impact gameplay further experimentation was implemented. This added a strategy selection element into the code. Now the agent can infer the most likely strategy of its opponent and change its policy to counter such a strategy. To elaborate, what is observed is a policy. The inference of the strategy, then, is based on knowing the list of policies within a given strategy. There could be such a policy that it is included in multiple strategies, so the Strategy Inference engine would have to observe policy shifting to determine the opponent strategy. The results of this experimentation are show in Table 3.

These results show significant improvement. Now the win rate (percentage) went from 34.4% without inference to 60.7% with inference. The *not lose* percentage jumps from 51.7% to 74.2%. It is noteworthy that the experiments with inference break even with the random strategy. Thus inference can help against random strategies without significant penalty. With such quick strategy recognition there is very little variation in the scores.

While it is not surprising that choosing a better policy produces a better result, recall that the goal was to prove that it is possible to infer the strategy of the opponent and thus counter it with real-time analysis during play. This speed of

recognition (shown in Table 2) shows that this is possible. This result proves that, as framed, inference is credible and counter-strategy selection practical.

Conclusions

We have shown that stochastic games can be described in terms of policies. Those policies sharing common traits can be aggregated into strategies. Those strategies can determine the next action and infer the next actions of other agents in the game. Thus an artificial intelligence can adapt it own behavior by selecting a better policy from within its strategy to overcome the strategy in use by its opponents, all within the confines of real-time.

References

- Bowling, M. and Veloso, M. 2001. Rational and convergent learning in stochastic games. *International Joint Conference on Artificial Intelligence*, volume 17, pages 1021–1026. Lawrence Erlbaum Associates, LTD.
- Bowling, M. and Veloso, M. 2004. Existence of multiagent equilibria with limited agents. *J. Artif. Intell. Res. (JAIR)*, 22:353–384.
- Kim, Chang Wook and Cho, Seongwon and Lee, Choong Woong. 1995. Fast competitive learning with classified learning rates for vector quantization. *Signal Processing Image Communications*, 6:499–505.
- Chernova, S. and Veloso, M. 2009. Interactive policy learning through confidence- based autonomy. *Journal of Artificial Intelligence Research*, 34(1)
- Fisher, Len. 2008. Rock, paper, Scissors: Game Theory in Everyday Life. *Basic Books, First Edition*.
- Cheng, Shih-fen et al., 2004. Notes on Equilibria in Symmetric Games. *Proceedings of 6th International Workshop on Game Theoretic and Decision Theoretic Agents (GTDT)*, 71–78.
- Bergsma, Maurice and Spronck, Pieter, 2008. Adaptive Spatial Reasoning for Turn-based Strategy Games. *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, 161–166.