ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

Improvement in Game Agent Control Using State-Action Value Scaling

Leo Galway, Darryl Charles and Michaela Black

School of Computing & Information Engineering
University of Ulster at Coleraine
Cromore Road, BT52 1SA
United Kingdom

Abstract. The aim of this paper is to enhance the performance of a reinforcement learning game agent controller, within a dynamic game environment, through the retention of learned information over a series of consecutive games. Using a variation of the classic arcade game Pac-Man, the Sarsa algorithm has been utilised for the control of the Pac-Man game agent. The results indicate the use of state-action value scaling between games played as successful in preserving prior knowledge, therefore improving the performance of the game agent when a series of consecutive games are played.

1 Introduction

Digital games provide an interesting test-bed for machine learning research due to the characteristically non-deterministic, dynamic nature of their environments [1]. In particular, the dynamic environments presented by predator/prey style games offer the advantage of being easily decomposed into a finite set of states, each with an associated set of reward values [2]. In order to generate reactive and believable game agent behaviours the use of machine learning techniques is required. However, the effective use of such algorithms is restricted by a number of requirements including the necessity for game agent behaviours to be learned in response to a changing game environment [1], [3]. Subsequently, by incorporating prior knowledge about the learning task into the learning algorithm and knowledge representation used, the performance of the game agent may be improved [4], [5].

Although a large variety of techniques exist within the machine learning domain, reinforcement learning provides an approach to agent-based learning which focuses on an agent's interactions with its environment [6]. As such, reinforcement learning provides a learning methodology appropriate for use within digital game environments and comprises a set of algorithms and techniques which involve learning a sequence of actions in order to maximize an accumulated discounted reward received from the environment over a period of time. Subsequently, a *control policy* can be learned, through an agent's exploration and exploitation of the environment, without requiring explicit training from a domain expert [4], [6], [7], [8]. For a comprehensive discussion on reinforcement learning, please refer to [6].

Within the academic digital games research literature, reinforcement learning techniques have been applied to a variety of games in order to learn control policies for game agents. Research conducted includes the *Sarsa(λ)* generation of a near-optimal control policy for game agents in the fighting game "Tao Feng" [8], and both

Sarsa and Sarsa(λ) based game agent controllers within a variation of the game "Pac-Man" [5]. Similarly, Sarsa(λ) has been used to generate a control policy for game agent strategies in the game "Settlers of Catan" [4]. By making use of domain specific knowledge within both the value function representation and learning algorithm used, enhanced game agent control has been observed [4], [5].

Based on previous research conducted into reinforcement learning-based control of a game agent within a dynamic, 2D game environment [5], the objective of the research outlined in this paper is to improve the performance of a game agent over a series of games played consecutively by retaining information about the learning task between games. Experiments regarding the retention of state-action values between successive games have been performed, including investigations into the use of linear scaling as a mechanism for retaining learned information over a series of games. Details of the experiments will be presented, along with analysis of the results obtained by the game agent controller, discussed in terms of the game related objectives of the game agent.

2 Methodology

The game environment employed was a variation of the classic arcade game Pac-Man [9], in which the primary objective for the player is to achieve as high a score as possible by manoeuvring the game agent (*Pac-Man*) around a 2D, grid-based environment in order to consume dots while at the same time avoiding being eaten by four opponent agents (*ghosts*). Consisting of a 20x20 grid of game-dependent *features*, including *walls*, *dots*, *energizers*, *tunnels*, *inaccessible spaces* (i.e. grid cells for the starting position of the opponent agents), and *empty spaces* (i.e. grid cells where a dot/energizer has been consumed), the configuration of features establishes a single game *level*. Initially populated with 180 dots (176 dots & 4 energizers), a single level has been used throughout all experiments performed. Within the course of a game, if the game agent consumes an energizer the game state of the opponent agents temporarily changes from the default *Attack* state to the *Evade* state, during which the game agent may eat the opponent agents. The duration of the Evade state has been predefined as 300 simulation steps, which may be further reset to a maximum possible duration of 300 simulation steps each time one of the remaining energizers is consumed within the 300 simulation steps. If an opponent agent eats the game agent while in the Attack state, the game agent loses 1 out of 5 lives. Eaten agents are regenerated within the inaccessible spaces, with the game state of opponent agents being reset to the Attack state. For both the game agent and opponent agents, moves may be made in the North, South, East and West directions. A series of moves were randomly pre-generated and used during game-play for each of the opponent agents, thus preventing the learning algorithm from simply learning a deterministic set of movement patterns for the opponent agents. To allow for direct comparisons of games played, the game agent has been restricted to a maximum of 1000 moves per game. A game ends when the number of game agent's lives has reached 0, the total set of dots, including energizers, has been consumed, or the game agent has reached the maximum number of moves permitted. For the level used within the experiments presented, a maximum possible score of 2680 may be achieved by the game agent.

During each game played the choice of moves for the game agent was made using the Sarsa control algorithm. By decomposing the game environment into a 20x20 grid, each grid cell was represented by the control algorithm as a state, with actions corresponding to the 4 possible moves. Due to the finite number of resulting state-action pairs, the value function, *Q(s,a)*, was represented as a look-up table. Every time a move was required for the game agent, the control algorithm was run for 100 episodes of learning, with the initial state for each episode of learning corresponding to the current position of the game agent. For each episode of learning performed, 2 steps look-ahead were used and the appropriate state-action values updated. The action corresponding to the state-action pair with the highest value for the state associated with the current position of the game agent was then chosen as the game agent's move. Throughout all episodes of learning a ε-greedy action selection policy was used with a low exploration rate ($\varepsilon = 0.1$) in order to maintain a high degree of exploitation of learned state-action pairs.

For the experiments presented in this paper two series of 20 consecutive games were played. In the first game played in each series, the initial values for the state-action pairs were generated randomly with values in the range [-0.1, 0.1]. As game-play progressed, the state-action values were updated by the Sarsa algorithm as each movement choice was made for the game agent. In the first series of 20 games played, the set of state-action values at the end of $game_n$ were used as the initial set of state-action values at the start of $game_{n+1}$, without further re-initialisation. By contrast, in the second series of 20 consecutive games played, the set of state-action values at the end of $game_n$ were linearly scaled within the range [-0.1, 0.1] before being used as the initial set of state-action values for $game_{n+1}$.

Throughout the experiments performed a number of game related metrics were recorded for each game played, including the score obtained, the number of dots consumed by the game agent, and the entropy of the moves made by the game agent over the course of a game. In particular, previous research has shown that the entropy of the moves made by the game agent may be used to measure the spatial diversity of the agent over the game environment. As such, games in which a high entropy value is measured indicate a more interesting range of moves have been made by the game agent [10]. The normalized entropy of the game agent, E_n, was obtained using the following set of equations:

$$E_r = -\sum \frac{p_i}{P} \log_2 \left(\frac{p_i}{P} \right)$$

$$E_n = \left(\frac{E_r}{\log_2 (P)} \right) \tag{1}$$

where p_i is a count of the number of times a specific grid cell has been visited by the game agent and P is the total number of moves made by the game agent during the course of a game [5], [10].

3 Results

Based on the results from both series of 20 consecutive games played, Table 1 shows the percentage of the maximum possible score obtained together with the percentage of the maximum possible number of dots consumed during all games.

	%Score		%Dots Consumed	
Num Games	**No Scaling**	**Scaling**	**No Scaling**	**Scaling**
1	47.57	47.57	99.44	99.44
2	12.50	43.28	32.22	97.78
3	26.31	55.22	68.33	100.00
4	18.47	38.99	45.00	90.00
5	10.07	43.66	30.00	98.89
6	11.94	43.84	30.56	99.44
7	9.51	51.49	23.33	93.89
8	11.01	37.87	32.78	86.67
9	14.93	47.57	39.44	99.44
10	15.11	43.28	40.00	97.78
11	17.91	46.27	37.22	95.56
12	13.43	42.35	35.00	95.00
13	14.37	39.55	26.67	91.67
14	12.31	47.57	31.67	99.44
15	10.45	43.84	31.11	99.44
16	7.84	41.98	23.33	87.78
17	1.87	43.84	5.56	99.44
18	5.04	43.66	15.00	98.89
19	5.41	41.98	16.11	87.78
20	7.65	39.74	17.78	98.33

Table 1: Percentage of Maximum Possible Score & Percentage of Maximum Possible Number of Dots Consumed

Correspondingly, Figure 1 illustrates the normalized entropy values calculated for the game agent's moves over both series of games played. From both Table 1 and Figure 1 it can be seen that the performance of the game agent over a series of consecutive games is largely improved when using state-action value scaling between games. Although the state-action values change in proportion to the total number of episodes of learning, the results indicate the learning algorithm is less effective at retaining learned information between consecutive games when no constraint is placed on the growth of the state-action values. This may be explained by the lessening impact of the set of reward values on the accumulating set of state-action values obtained during learning over successive games. By constraining the overall magnitude of the state-action values using linear scaling between games, information learned during a game, encoded by the state-action values, is no longer represented by values that may subsume the reward values used during further periods of learning.

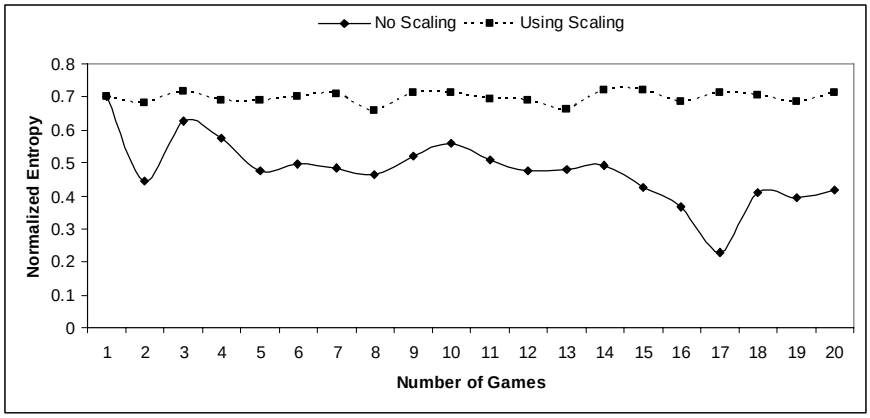

Fig 1: Normalized Entropy of Game Agent Moves

Table 1 and Figure 1 also illustrate that the results obtained from the first game in each series show a relatively high level of performance by the game agent. However, for the series of games played without state-action value scaling, Table 1 shows a drop in performance in terms of the percentage of the maximum possible score obtained and the percentage of the maximum possible number of dots consumed over the remaining games played of approximately 84% and 82% respectively. Similarly, Figure 1 shows a drop in the normalized entropy of approximately 40%. By contrast, in the series of games played which incorporate scaling between games, the performance of the game agent oscillates without a significant drop in the results obtained. Again, these results suggest that without the constraint imposed by the use of scaling on the growth of the state-action values over a series of consecutive games, the effect of the rewards becomes diminished during learning. Conversely, even when incorporating state-action value scaling, the results depicted in Table 1 and Figure 1 also indicate that the amount of new information being learned by the game agent between games is somewhat reduced after the first game in the series. Although marginal gains and loses in the performance of the game agent are shown, no overall increase in performance occurs, which would be expected if the control policy was improving over successive games. In particular, the range of normalized entropy values over the series of games, illustrated in Figure 1, suggests the control algorithm has learned the static features of the game environment, such as the walls and inaccessible spaces, as the spatial diversity of the game agent remains consistent over the 20 games played. If the dynamic features of the game environment, including the opponent agents, dots and energizers, were being repeatedly learned, an improvement in the percentage of the maximum possible score would be expected.

4 Conclusion

It has been shown that a reinforcement learning controller may be successfully used for the real-time generation of game agent moves within a dynamic digital game environment, yet the performance of the game agent may deteriorate over a series of consecutive games unless constraints are imposed on the magnitude of the state-action

values generated. Through repeated learning using a single set of state-action values for the duration of all games played, coupled with a fixed set of reward values, the accumulation of state-action values reduces the effectiveness of the environmental feedback used during learning. Subsequently, the observed results of the game agent diminish over the number of games played. One way to limit such an accumulation of the state-action values, thereby preventing degradation of the game agent's performance, is through the use of state-action value scaling between games. From the experiments presented herein, this technique has been shown to be successful in preventing the decline in game agent performance observed over consecutive games played without the use of scaling. By providing a uniform basis over the range of state-action values used, the effectiveness of the reward values is maintained.

In terms of retaining prior information between games, the results of the experiments featuring state-action value scaling have shown the preservation of information predominantly representing the static features of the game environment. Exposing a limitation of the experiments presented, a more generalized control policy is required in order to successfully learn both the static and dynamic features of the game environment. Further research should be conducted into the use of a more suitable, dynamic value function representation and into the decomposition of the learning task into separate tasks for both types of features found within a dynamic game environment.

References

[1] P. Spronck, A Model for Reliable Adaptive Game Intelligence. In *Proceedings of 2005 Workshop on Reasoning, Representation and Learning in Computer Games*, Washington (USA), 2005.

[2] T. Hartley, Q. Mehdi and N. Gough, Applying Markov Decision Processes To 2D Real Time Games. In Q. Mehdi, N. Gough and S. Natkin, editors, *Computer Games: Artificial Intelligence, Design & Education*, pages 55-59, University of Wolverhampton (UK), 2004.

[3] M. Van Lent and J. Laird, Developing an Artificial Intelligence Engine. In *Proceedings of 1999 Game Developers Conference*, pages 577-588, San Jose (USA), 1999.

[4] M. Pfeiffer, Reinforcement Learning of Strategies for Settlers of Catan. In Q. Mehdi, N. Gough, S. Natkin and D. Al-Dabass, editors, *Computer Games: Artificial Intelligence, Design and Education*, pages 384-388, University of Wolverhampton (UK), 2004.

[5] L. Galway, D. Charles, M. Black and C. Fyfe, Temporal Difference Control Within A Dynamic Environment. In M. Roccetti, editor, *Proceedings of 8th International Conference on Intelligent Games & Simulation* (GAME-ON 2007), pages 42-47, November 20-22, Bologna (Italy), 2007.

[6] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[7] C. Baekkelund, A Brief Comparison of Machine Learning Methods. In S. Rabin, editor, *AI Game Programming Wisdom 3*, pages 617-631, Charles River Media, 2006.

[8] T. Graepel, R. Herbrich and J. Gold, Learning To Fight. In Q. Mehdi, N. Gough, S. Natkin and D. Al-Dabass, editors, *Computer Games: Artificial Intelligence, Design and Education*, pages 193-200, University of Wolverhampton (UK), 2004.

[9] Namco, *Pac-Man*. 1980.

[10] G. Yannakakis and J. Hallam, Evolving Opponents For Interesting Interactive Computer Games. In *Proceedings of 8th International Conference on Simulation of Adaptive Behaviour*, pages 499-508, 2004.