Learning to Behave by Environment Reinforcement

Leonardo A. Scardua*, Anna H. Reali Costa**, and Jose Jaime da Cruz***

Escola Politecnica - Universidade de Sao Paulo, Av. Prof. Luciano Gualberto,
Travessa 3, 158,
05508-900 Sao Paulo, SP, Brasil
{scardua, anna}@pcs.usp.br, jaime@lac.usp.br

Abstract. This paper describes a softbot agent capable of learning to choose its actions, in order to achieve its goal when facing an opponent in a dynamic environment. The agent uses rewards gathered from the environment to assess and improve the quality of its own behavior. A multilayer perceptron neural network is assessed regarding its adequacy as a value function approximator for state-action pairs in the robotic soccer domain.

1 Introduction

The reinforcement learning approach is a very effective alternative to the standard supervised training of artificial neural networks (ANNs). Instead of having an external teacher that indicates the correct output for each given input, reinforcement learning simply provides a reinforcement signal that indicates the quality of the network output. According to [10], one can identify the following main sub-elements of a reinforcement learning system: a *policy*, a *reward function*, a *value function* and sometimes a *model of the environment*.

A *policy* is a mapping from perceived states to actions to be taken in those states.

A *reward* function defines the goal in a reinforcement learning problem, mapping each pair formed by a state s and an action a to a single number. This single number is the reward that indicates the desirability, in an immediate sense, of choosing action a when in state s.

The *value function* indicates the benefit of selecting action a when in state s, in a long-term sense. Roughly speaking, the value of a state-action pair is the total amount of reward an agent can expect to receive if it takes action a when in state s, following a policy. The only goal of a reinforcement learning agent is to maximize the total reward it receives in the long run. Methods for determining values are the most important components of almost all reinforcement learning systems.

* Leonardo Azevedo Scardua is supported by CNPq grant number 141802/97-9.
** Anna H. Reali Costa is partially supported by FAPESP grant number 98/06417-9.
*** Jose Jaime da Cruz is partially supported by CNPq grant number 304071/85-4(RN).

M. Veloso, E. Pagello, and H. Kitano (Eds.): RoboCup-99, LNAI 1856, pp. 439–450, 2000.
Springer-Verlag Berlin Heidelberg 2000

In stochastic domains, a *model* describing the state transition probabilities and the resulting immediate rewards is usually not available, at least in many relevant domains. For this reason, the reinforcement learning agent learns through direct interaction with the environment.

The main objective of our research is to develop a team of agents capable of learning cooperative behavior solely by observing the impact of their actions in the environment. To achieve this goal we have been developing an agent capable of learning to choose its actions by observing environment rewards. The first phase of the development, described in this paper, intends to assess an (ANN) as an action evaluator in the robotic soccer domain.

The RoboCup soccer server is a simulated robotic soccer domain that allows matches between two teams of up to 11 players, where each player is controlled by a single process. The soccer server works in 100 milliseconds (this is an adjustable parameter but 100 is its default value[2]) simulation cycles. It demands that agents act in real time. The environment changes are influenced by the actions of both teams of agents. This combination of features makes it a very complex and realistic domain, where decisions are made in stages, and the output of a decision can not be fully predicted. Each decision results in some immediate reward and affects the environment, influencing the reward received by the next decision to be made.

In the following section, it is first introduced the learning agent's structure and the environment state perceived by the agent. Section 3 justifies the use of an ANN as a value function approximator. It also describes the overall approach implemented, including the reward signal and the ANN. Section 4 presents the results, which are discussed in Sect. 5. Section 6 presents some related work and Sect. 7 presents the conclusions.

2 The Learning Agent

The softbot agent is in essence a reinforcement learning agent, whose main objective is to gather the maximum reward. This goal can be achieved by choosing the action that has the best value in face of the present state signal.

The agent must learn while interacting with the environment since it has no previous knowledge about the task or the domain it is about to face. Once it is not practical to use memory tables to keep the value of each state-action pair that is experienced by the agent (Sect.2.1), we decided to use an (ANN) to approximate such values (Sect.2.2).

The agent's structure is very simple, yet efficient. It works according to the following loop:

```
Sense the environment state.
If it is time to choose the best action:
  Use the neural network to evaluate each of the available actions
   in light of the state signal.
  Choose the action that has the best-estimated value.
```

```
If it is time to choose randomly:
  Choose randomly among the set of available actions.
Execute the chosen action.
Gather the reward.
Update the value estimate of the neural network.
```

2.1 The Environment State

The learning player senses the world according to the following state signal:

1. The angle between the ball and the learning agent.
2. The distance between the ball and the learning agent.
3. The angle between the opponent and the learning agent.
4. The distance between the opponent and the learning agent.

According to [2], the soccer field size is 105 x 68 where the unit is meaningless. We use the arctan function to calculate the relative angles above. We did not use any kind of generalization, in order to reduce the size of the state space. As the Soccer Server gives us float point precision, we have an enormous state space.

2.2 Set of Actions

Each of the agents is given a set of three actions, each action is implemented by a routine that has all the information it needs to perform the action.

Get the Ball.

Pre-conditions : The ball must not be within kicking distance of the agent.

Effect : This moves the agent (A) in the direction of the ball. When it reaches the ball, A stops within kicking distance from it.

Kick Towards Adversary Goal.

Pre-conditions : The ball must be within kicking distance of A.

Effect : This kicks the ball toward the adversary goal.

Turn the Ball.

Pre-conditions : The ball must be within kicking distance and behind A, with respect to the adversary goal.

Effect : This turns the ball towards the adversary goal, keeping it within kicking distance. When the ball is in front of A with respect to the adversary goal, A kicks the ball.

2.3 Decision Cycle

In this paper, we had matches between two opposing players, without goalies. One player uses the learning structure already depicted while the other chooses its actions at random. Both agents must decide which action will be taken at each simulation cycle. The action set and the short simulation cycle allow even the random player to score consistently if it does not face opposition.

3 The Learning Approach

As the main objective of this work is to assess an ANN as a function approximator in the robbotic soccer domain, we are not dealing with delayed reward.

3.1 Why a Neural Network as Value Function Approximator

It is clear that, given the chosen state representation, it is not practical to keep the actions values in memory tables, since the number of possible states is very large. This imposes the use of an approximator for the value function.

The chosen approximation architecture must attend at least, the following conditions:

1. Once it is not possible to know beforehand the features of the value function that must be approximated, the architecture must allow for good generic function approximation for at least continuos functions.
2. It must be capable of working well even in a noisy, uncertain environment.
3. It must be able to work under real-time demand.

The Kolmogorov's Mapping Neural Existence Theorem [9] gives the mathematical justification for the use of three layer perceptron networks, as universal continuos function approximators.

3.2 Coding the Agent's Goal as a Reward Signal

In the reinforcement learning framework, the reward signal is used to encode what we want the agent to do. It is the same in the present case, where the reward following the choice of each action is used to inform the agent whether it was a "good" or a "bad" choice.

In this paper, "good" means choosing an action that has all of its pre-conditions (constraints that must be valid, given the present state signal, such that the action is applicable) satisfied by the state signal the agent perceives from the environment. In this case the reward signal is positive.

If anyone of the pre-conditions of the chosen action is unsatisfied, the reward signal will be negative. In other words, the encoded goal is to learn to choose actions that have all pre-conditions satisfied by the perceived state signal. This goal and method for calculating the reward are intended just to provide an easy yet valid mean to evaluate the proposed value function approximator.

3.3 Learning the Value of a State-Action Pair

To learn the value of a state-action pair, the (ANN) is presented with the action chosen by the agent and the perceived state signal. The network output must be equal to the reward immediately received by the agent. In other words, when we talk about values in this work, we are talking about immediate rewards (given the goal of this work, that is to assess a ANN as a function approximator in the robotic soccer domain, we are not dealing with delayed rewards). If the ANN is able to learn the immediate rewards, it will be able to learn the values of state-action pairs (in the usual reinforcement learning sense) when dealing with delayed rewards.

Given the goal proposed to the agent, there is no need to estimate the true value of all state-action pairs; learning the immediate rewards will be enough.

3.4 The Neural Network

The neural network shown in Fig. 1 is a three-layer fully connected feedforward perceptron, with ten hidden neurons and one output neuron.

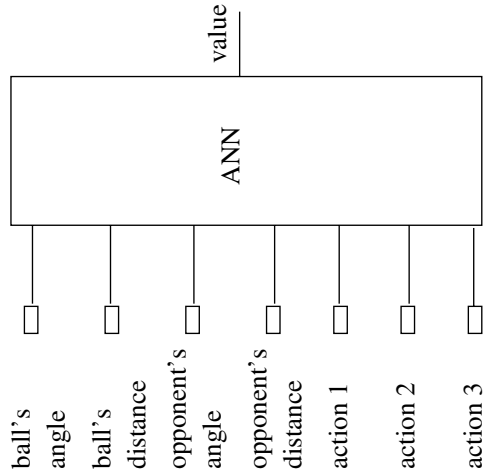

Fig. 1. ANN diagram

The hidden neurons use the hyperbolic tangent as the activation function while the output neuron just sums up all of its inputs. The training method is the backpropagation algorithm [6].

3.5 Choosing the Best Action

The choice of the best action to be taken is based solely on the state signal perceived by the agent. When it is time to choose an action, the agent feeds

the (ANN) input with the current state signal and the input corresponding to the action being evaluated is made positive, while the other action inputs are kept at zero level. The ANN output is then recorded. This procedure is repeated for each existing action and the agent chooses that one that yields the biggest numerical output of the ANN. Each action is implemented by a routine and identified uniquely by a positive number.

4 Results

To assess the effectiveness of the proposed state-action pair value approximator, we conducted experiments consisting of a series of ten-minute games. The learning agent faces an opponent that selects its actions randomly.

If the learning agent effectively learns to take actions which pre-conditions are satisfied by the state of the environment, it will be able to outscore a random selecting agent, that wastes time selecting actions that may not be effective in the current environment state.

4.1 Experiment Results

Experiment 1 - The learning agent always selected a random action and did not learn from the rewards received from the environment. Therefore, the learning player actually performed just like a random player. The goal of this experiment is to assess the performance of two random players.

The results, presented in Tables 1 and 2 and Fig. 2, show that in this case, the performance is the same for both players.

Table 1. The score of each match in experiment 1

Game Number	Learning Player	Random Player
1	5	2
2	2	5
3	3	4
4	4	4
5	3	5
6	4	3
7	3	5
8	5	4
9	5	5
10	6	4

Table 2. Some statistics referring to experiment 1

Game Number	Learning Player	Random Player
Number of goals	40	41
Average goals	4	4.1
Number of wins	4	4

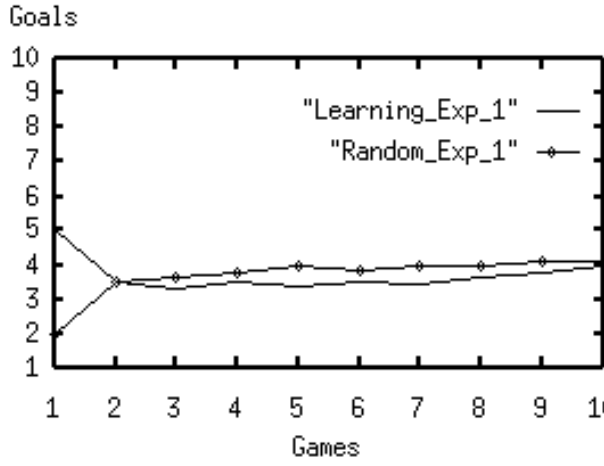

Fig. 2. Average score per game in experiment 1

Experiment 2 - At random, with probability 0.2, the learning agent selected the most valuable action, and with probability 0.8 selected randomly one action from the set of available actions.

At the beginning the ANN's weights were assigned randomly, and they changed according to the learning process throughout each game. The goal of this experiment is to assess the effect of the learning process on the player's performance.

The results, presented in Tables 3 and 4 and Fig. 3, show that in this case, the learning player is becoming superior as the number of games grow.

Table 3. Some statistics referring to experiment 2

Game Number	Learning Player	Random Player
1	5	5
2	9	3
3	2	7
4	3	5
5	6	2
6	6	2
7	2	6
8	6	3
9	7	2
10	5	2

Table 4. Some statistics referring to experiment 2

Game Number	Learning Player	Random Player
Number of goals	51	37
Average goals	5.1	3.7
Number of wins	6	3

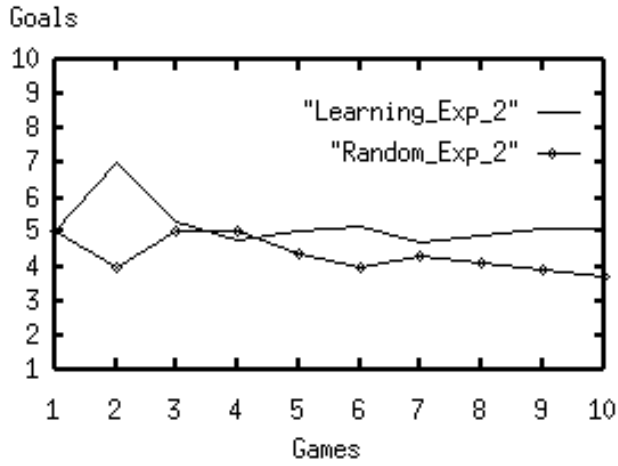

Fig. 3. Average score per game in experiment 2

Experiment 3 - At random, with probability 0.8, the learning agent selected the most valuable action, and with probability 0.2 selected randomly one action from the set of available actions.

At the beginning the ANN's weights were assigned randomly, and they changed according to the learning process throughout each game. The goal of this experiment is to assess the effect of the learning process on the player's performance.

The results, presented in Tables 5 and 6 and Fig. 4, show that in this case, the learning player becomes much superior as the number of games grow.

Table 5. Some statistics referring to experiment 3

Game Number	Learning Player	Random Player
1	6	4
2	7	2
3	5	4
4	10	2
5	7	3
6	7	3
7	7	2
8	8	4
9	9	4
10	10	2

Table 6. Some statistics referring to experiment 3

Game Number	Learning Player	Random Player
Number of goals	76	30
Average goals	7.6	3.0
Number of wins	10	0

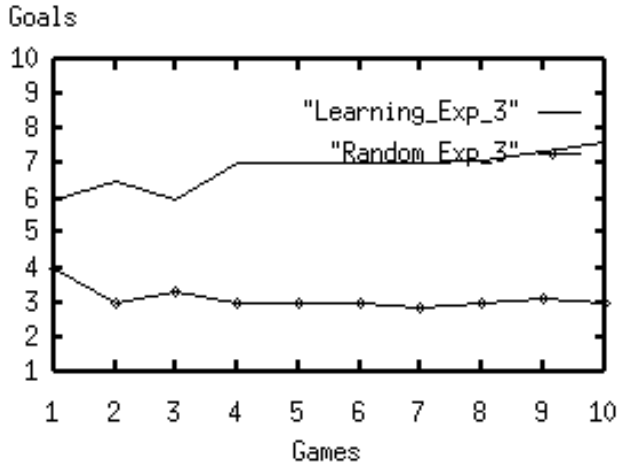

Fig. 4. Average score per game in experiment 3

5 Discussion

The simulation results show that:

1. The average number of goals per game of the random selecting player consistently decreases as the probability of selecting the most valuable action for the learning agent increases.
2. The average number of goals per game of the learning agent consistently increases as the probability of selecting the most valuable action increases.
3. The number of wins of the learning agent consistently increases, reaching 100%, as the probability of selecting the most valuable action increases.

Since the only difference between the three experiments is the learning agent's probability of choosing the best-estimated action, it is clear that this is what caused the above changes in the experiments' statistics. Because of the fact that the increase in this probability improved the performance of the learning agent and hindered the performance of the random player, it is clear that the learning agent has learned something useful. This allowed the learning agent to consistently outscore the random selecting agent.

These results make clear that a feedforward multilayer perceptron is able to approximate the value function for a reinforcement learning agent, in the robotic soccer domain, in a fast and reliable way.

6 Related Work

The work presented in this paper is inspired by Tesauro's TD-Gammon [11], an (ANN) that is able to teach itself to play backgammon solely by playing against itself and learning from the results.

The domain of backgammon, like robotic soccer, is a domain where decisions are made in stages, with a huge number of possible states, making it impossible to use a supervised learning approach for the neural network. Both domains involve playing against an unknown opponent.

Many researchers have been using learning methods in the robotic soccer domain. Among the more common methods are genetic algorithms [7], genetic programming [1], case-based reasoning [5] and reinforcement learning [3].

7 Conclusion and Future Work

Our agent learned to choose its actions using solely the rewards it obtained while interacting with the environment. It is important to stress that it did not have any previous knowledge about the robotic soccer domain in any form. It is an (ANN) that learned to play just by observing the rewards gathered while playing.

One important aspect to be emphasized is that the learning is naturally focused on the state trajectory followed by the learning agent. This greatly

reduces the space state complexity, explaining the fast learning that allowed our agent to win every match in the third experiment, even having started with random weights in its neural network.

The first phase of the development of our research, described in this paper, intended to assess an (ANN) as an action evaluator in the robotic soccer domain. The results are encouraging. Our research continues with the development of a team of eleven agents that learn by being rewarded solely when they score a goal and being punished when the opponent team scores a goal. In the next step, the agents will be controlled by a version of Richard Sutton's Sarsa algorithm [10], improved with an augmented version of the ANN used in the present work, to approximate the value function.

Robotic soccer demands that the agents learn and act in real time. The output of each action is not fully predictable due to random noise imposed by the RoboCup Soccer Server. The environment changes are influenced by the actions of both teams of agents. This combination of features makes this a very complex and realistic domain. This complexity allows us to assert that the technology we have been developing can be applied to real tasks that demand learning of a control policy in dynamic huge state space environment. Good examples of real tasks are dynamic channel allocation in cellular telephone systems and elevator dispatching[10]. These two applications show the importance of a good value-function approximation architecture.

References

1. Andre, D., Teller, A: Evolving Team Darwin United. Proceedings of the Second RoboCup Workshop, Paris (1998) 317–323
2. Andre, D., Corten, E., Dorer, K., Gugenberger, P., Joldos, M., Kummeneje, J., Navratil, P., Itsuki, N., Riley, P., Stone, P., Takahashi, T., Yeap, T: Soccerserver Manual. (1999) www.dsv.su.se/ johank/RoboCup/manual
3. Andou T: A RoboCup Team wich Reinforces Position Observationally. Proceedings of the Second RoboCup Workshop, Paris (1998) 361–363
4. Bertsekas, D. and Tsitsiklis, John: Neuro Dynamic Programming Athena Scientific, Belmont, MA, (1996)
5. Burkhar, H., Wendler, J., Gugenberger, P., Schroder, K., Kuhnel, R: AT-Humboldt in RoboCup-98. Proceedings of the Second RoboCup Workshop,Paris (1998) 331–337
6. Haykin, S.: Neural Networks: a comprehensive foundation. 2nd ed., Prentice Hall, (1999)
7. Kuzuaki, E., Sadaharu, I., Yamaguchi, H., Nobui, I. and Yoshiyuki, K: Team Description for Donguri. Proceedings of the Second RoboCup Workshop, Paris,(1998), 305-308
8. Matsumura T: Description of Team Erika. Proceedings of the Second RoboCup Workshop, Paris (1998) 309–315
9. Hetch-Nielsen, R: Neurocomputing. Addison Wesley Publ. Co., New York,(1990)
10. Sutton, R. and Barto, A: Reinforcement Learning: an introduction. MIT Press, (1998)
11. Tesauro, G: Temporal Difference Learning and TD-Gammon. Communications of the ACM, (1995) Vol 38 No.3 58-68