Soccereus 2D Simulation Team Description Paper

Shayan Salehian[1], Ehsan Imani[2] Anahita Hosseini[3], Sahand Mozaffari[4], and Mohammad Ali Baghershemirani[5]

[1] salehian@ce.sharif.edu, Sharif University of Technology, Tehran
[2] eimani@ce.sharif.edu, Sharif University of Technology, Tehran
[3] anahosseini@ucla.edu, University of California, Los Angeles
[4] smozaffari@ce.sharif.edu, Sharif University of Technology, Tehran
[5] bagher@ce.sharif.edu, Sharif University of Technology, Tehran

Abstract. Soccer simulation is a ground where one can utilize theoretical concepts of Artificial Intelligence in a practical field. Majority of teams participating in this international league give their complete attention to enhance high level and more common actions through a learning process. In this paper we employ several methods of machine learning to improve functionalities that has received less attention in the past years. We propose three AI based algorithms to improve the shooting skill of the agent, increase probability of scoring a goal from corner kicks, and block the attacker in penalty shootouts.

Keywords: artificial intelligence, genetic algorithm, neural networks, reinforcement learning, soccer simulation, shoot, corner kick, penalty shootout

1 Introduction

Soccereus [1] is a 2D soccer simulation team that was established in 2011. The team was the outcome of cooperation of a group of students, majoring in software engineering at Sharif University of Technology who had found themselves absorbed in Artificial Intelligence area and most of them had had a previous experience in 2D soccer simulation leagues and had achieved several awards in national leagues. Soccereus has participated in three national leagues in the past year, and has ended second in Sharif Cup and PNU National contests and fifth in Iran Open league.

After some basic development of high level actions on Agent2D base, by H. Akiyama [2], teams goal changed to developing a multi-aspect intelligent agent. The first action that drew our attention was shooting and consequently we developed a neural network based shooting method. Then we pondered that corner kick is a great opportunity for every team to score a goal. However, most existing teams do not have strategic and specific plans for this critical situation. They usually start with a long pass or in some cases they utilize static decision makings. Along with these, we used reinforcement learning to train the goalie for penalty shootouts.

2 Corner Kicks

To enable the agents to perform the corner kicking at their best, an algorithm was needed to decide how the players should position themselves and also to determine the sequence of passes that gives our team the most chance of scoring a goal. To accommodate this need, a continuous genetic algorithm [3] was utilized to plan a strategy for scoring from corners. When the first corner kick is awarded to our team,

the coach observes the positions of the opponent defenders and the goalie. Then he runs the algorithm and informs the involved players of their positions and the pass sequences to be used in next corner kicks.

In our data representation, each gene consisted of 5 positions for offensive players. Each position contained the radius and angle relative to the players position in the formation. Also, the cost of each gene was the risk of losing possession of the ball. To compute the cost function, a directed graph was defined in which, each vertex corresponded to a player. For each i and j, the edge from vertex v_i to vertex v_j had a cost relative to the chance of losing possession of the ball when the i^{th} player passed the ball to the j^{th} player. There was also an extra vertex v_g corresponding to the opponents goal. For each i, the edge from v_i to v_g had a cost relative to the chance of failing to score when the i^{th} player shot the ball towards the goal.

Then a run of Dijkstras algorithm [4] was executed on the graph to find the least risky path from the corner kick taker to the goal. The output of cost function was computed by the cost of the path and the number of players participating in the pass sequence.

The initial population in our model, consisted of genes with random angles between 0 and 2 and random radii between 0 and 9. The algorithm used rank selection method to choose two parents and single-point crossover to generate two children from the parents. The positions before the crossover point were inherited from the first parent and the ones after the point were inherited from the second parent. The position whose index equaled to the crossover point was inherited in the following manner:

$$P_{new1} = P_{ma} - \beta[P_{ma} - P_{da}] \tag{1}$$

$$P_{new2} = P_{da} + \beta[P_{ma} - P_{da}] \tag{2}$$

where P_{new1} and P_{new2} are the positions in the crossover points of the first and second children, respectively. P_{ma} is the position in the crossover point of the first parent, P_{da} is the position in the crossover point of the second parent and is a random real number between 0 and 1.

To avoid premature convergence to a local optimum, in each generation, 5 percent of the population had a chance to be mutated i.e. a single position in each of them would be replaced by a new position with an angle between 0 and 2 and a random radius between 0 and 9. The parameters of the algorithm (number of generations and population) had to be set so that the algorithm could achieve a reasonable cost in a short time. Therefore, we ran the algorithm to find a safe path among the defenders of FC Persepolis 2013 [5], which had a very dense defensive formation and could be a challenging test for our algorithm. Then we analyzed the results based on the parameters. The results are shown in figure 1. As it can be seen, the algorithm converged after almost 50 generations and taking 1000 as the population led to a reasonable cost. Therefore we chose 50 for the number of generaions and 1000 for the population.

3 Shoot

The result of a game almost specifically relies on the scoring ability of agents and this makes shooting skill one of the most important high level skills. At first, we implemented this action by considering balls motion model, opponents defenders and goalie. This approach was similar to what Riton [6] had done in extracting possible

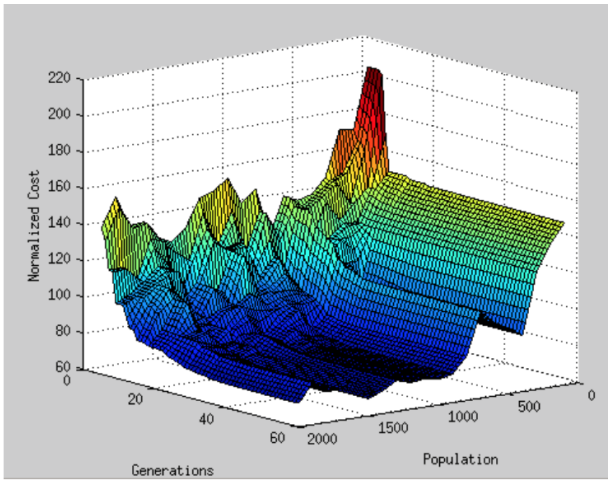

Fig. 1. The result of the genetic algorithm based on the number of generations and population against FC Persepolis 2013

shoot points. Since all the factors involved in scoring a goal are practically unfeasible to be taken into account, we reached to a limit in enhancement of that method. Therefore, we decided to adapt a new approach. Since Artificial Neural Networks [7] has been a popular option for data classification we employed its concepts to train the agent to choose the best parameters for a shoot.

3.1 Designing the training set

One fundamental step is designing a training set that covers all situations likely to happen. To this end we placed the under-training agent, uniformly in a continuous environment in the penalty area, and the opponent goalie was placed randomly in active penalty area.

3.2 Structure of the network

In construction of our neural network input, following criteria were taken into account.

- Angle between goalie and ball path
- Angle between goalie body and ball path
- Distance between goalie and ball
- Distance from ball to goal in balls direction
- Balls initial velocity

The single output of this network is a Boolean determining if a goal is scored or not and based on the result, sample data is classified into two classes. The network architecture is composed of an input layer with five inputs as described above, one hidden layer containing five neurons, and an output layer of a single neuron. The activation function for all the neurons in hidden layer are linear and output layer employs the Binary Sigmoid function.

In figure 2, we compare the results of this method to results of the earlier method we used. The two charts demonstrate the outcome of 500 rounds of shooting.

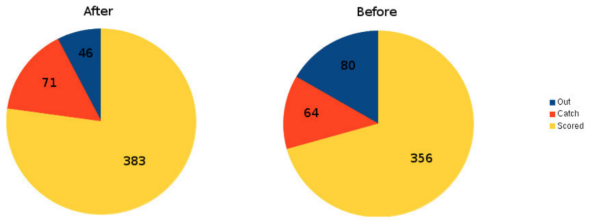

Fig. 2. Comparison of results from the old shoot algorithm with the neural networks based one

4 Penalty Shootouts

Penalty shootout is the tie-breaker in the knockout stage of the tournament. Improving the performance of the agent in this scenario demands considerable attention as it directly determines the winner. In this section, we focus on training the goalie for a penalty shootout scenario. We model it as a reinforcement learning [8] problem and propose a solution based on the recent advancements in deep reinforcement learning.

4.1 Deep Reinforcement Learning

In a reinforcement learning problem, the aim is to find a policy that maximizes the expected future rewards. Q-learning achieves this goal by finding an optimal action-value function denoted by $Q^*(s, a)$. This function shows the usefulness of taking action a in state s assuming that the subsequent actions are chosen by the optimal policy. A common approach in reinforcement learning is to estimate the optimal action-value function using Bellman equation:

$$Q^*(s, a) = E_{s' \sim \epsilon}[r + \gamma \max_{a'} Q^*(s', a')|s, a] \tag{3}$$

in which r is the reward resulted from taking action a in state s, ϵ is the environment, and γ is a discount factor.

Mnih et al. [9] proposed a method for end-to-end training of Deep Q-Network (DQN) that approximates the optimal action-value function. The idea is based on neural fitted Q-learning (NFQ) which uses resilient backpropagation and a loss function derived from the Bellman equation [10]. However, DQN is trained with experience replay. In this technique, at each time-step, the agent picks an action using ϵ-greedy policy, stores the reward and state transition in a dataset, and then applies minibatch updates on the dataset to approximate the Q-function. Off-policy learning with experience replay breaks the correlation between the training samples and makes them i.i.d., thus considerably improving the stability of the learning. Receiving raw pixels as input, this approach achieved a level of performance comparable to expert human players on many Atari 2600 games [9].

The methods discussed above are limited in that they only work on discrete action spaces. If the set of actions is finite and a good approximation of Q-function is available, the optimal policy can be achieved by a simple maximization over the actions. However, in many tasks, like 2d soccer simulation, actions can be high-dimensional and continuous and this optimization might be infeasible in such spaces. Silver et al. [11] proposed Deterministic Policy Gradient (DPG) algorithm which handles this

problem by training two parameterized functions simultaneously: i) an actor $\mu(s)$ that gives the action that should be taken in the state s and ii) a critic $Q(s, a)$ that approximates the Q-function and is trained using the Bellman equation. Roughly speaking, this method uses the critic function to compute the gradient of the expected future rewards w.r.t. the parameters of the actor function, thus finding a direction in which the actor parameters should change to achieve the optimal policy. Deep DPG (DDPG) combines this actor-critic approach with the ideas in DQN to solve problems with high-dimensional action and state spaces [12]. Similar to DQN, DDPG uses experience replay to train the Q-network with minibatch updates on the dataset of experiences. The method also uses soft target updates [12], batch normalization [13], and Ornstein-Uhlenbeck process [14] (for exploration) to further stabilize the learning process.

4.2 Modeling Our Problem

We model the performance of the goalie in a penalty shootout scenario as a reinforcement learning problem with continuous action and state spaces and train the agent with DDPG algorithm. Since the goalie and an attacker are the only present agents in the environment, the state can be described by these features:

- Position of the goalie
- Position of the attacker
- Position of the ball
- Velocity of the ball
- Body angle of the goalie
- Velocity of the goalie

The action that should be taken is specified by the following features:

- Angle of turn
- Power of dash (at 8 different angles)

The neck angle of the agent is heuristically set to follow the ball in order to avoid further complication of the action space. We also devise a reward function that is used in the training algorithm. A serious challenge in 2d soccer simulation is the extremely delayed reward. The naive reward function gives $+1$ and -1 only when a goal is scored and conceded respectively. The agent of our problem, whose aim is to block the attacker in a penalty shootout, is only concerned with conceded goals. Therefore the naive reward function can be described as a function of the distance between the ball and our goal:

$$R(x) = 2 * (H(x) - 1) \tag{4}$$

in which $H(x)$ refers to Heaviside step function. This function can be interpreted as a severe punishment that suddenly appears as the distance reaches zero. We use the alternative definitions of Heaviside step function to derive a "soft" version of this reward function:

$$R'(x) = 2 * (1/(1 + e^{-x/t}) - 1) \tag{5}$$

where t is a hyper-parameter that controls the "softness" of our function (note that $R'(x)$ behaves like R as $t \to 0$). Intuitively, the agent's aim is to keep the ball away from the goal but the punishment is not hard as long as the opponent is wandering outside the penalty area. While this formulation alleviates the problem of delayed reward, it should be noted that sufficient stochasticity in exploration is essential for the algorithm to converge.

5 Conclusion

In this paper we proposed several methods for improving different agent actions, in which we found room for improvement. They included actions such as shooting, corner kicking, and defending our goal in penalty shootouts. All of the implemented actions had a reasonable result and showed an improvement in comparison to our former methods. Our team intends to focus on improving other actions by means of Artificial Intelligence methods.

References

1. S. Mozaffari, M. baghershemirani, A. Hosseini, E. Imani, S. Salehian. Soccereus 2D simulation - Team Description Paper, 2014.
2. Akiyama, H.: Agent2D Base Code. http://www.rctools.sourceforge.jp
3. Randy L. Haupt, Sue E. Haupt. "Practical genetic algorithms", 2nd ed., Wiley-Interscience, 2004.
4. Cormen, Thomas H. Leiserson, Charles E. Rivest, Ronald L, Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". Introduction to Algorithms (Second ed.). MIT Press and McGrawHill. pp. 595601. ISBN 0-262-03293-7.
5. Amir Tavafi, Vahid Khodabakhshi, Nima Nozari, Michael Shaghelani, Hosseinali Zare, Marziye Hashemian. FC Perspolis 2013 Binary. http://chaosscripting.net/files/competitions/RoboCup/WorldCup/2013/2DSim/binaries/fcperspolis.tar.gz
6. M. Alavi, M. Falaki Tarazkouhi , A. Azaran , A. Nouri , S. Zolfaghari. RoboCup 2012 Soccer 2D Simulation Team Description Paper (Riton) [online], Available: http://www.socsim.robocup.org/ files/2D/tdp/RoboCup2012/.
7. Martin T. Hagan, Howard B. Demuth, Mark Beale. "Neural Network Design", Boston, 2002.
8. Richard S. Sutton, Andrew G. Barto. "Reinforcement Learning: An Introduction", MIT Press, Cambridge, 1998.
9. Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." Nature 518, no. 7540 (2015): 529-533.
10. Riedmiller, Martin. "Neural fitted Q iterationfirst experiences with a data efficient neural reinforcement learning method." In Machine Learning: ECML 2005, pp. 317-328. Springer Berlin Heidelberg, 2005.
11. Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. "Deterministic policy gradient algorithms." In ICML. 2014.
12. Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015). Harvard
13. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).
14. Uhlenbeck, George E., and Leonard S. Ornstein. "On the theory of the Brownian motion." Physical review 36, no. 5 (1930): 823.