Supervised Learning in Football Game Environments Using Artificial Neural Networks

Omer Baykal
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
obaykal@ceng.metu.edu.tr

Ferda Nur Alpaslan
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
alpaslan@ceng.metu.edu.tr

***Abstract*—Game industry has become one of the sectors that commonly use artificial intelligence. Today, most of the game environments include artificial intelligence agents to offer more challenging and entertaining gameplay experience. Since it gets harder to develop good agents as games become more complex, machine learning methods have started to be used in some notable games to shorten the development process of agents and to improve their quality. Popularity of machine learning applications in game environments has increased in the last decades. Supervised learning methods are applied to develop artificial intelligence agents that play a game like human players by imitating them. The imitating agents can either play the role of opponents or play on behalf of the real players when they are absent. The purpose of this study is to develop imitating agents for a popular online game, namely HaxBall. HaxBall is a two dimensional football game with fully observable, continuous, and real-time game environment.**

***Keywords*—artificial intelligence, learning systems, supervised learning, neural networks, feedforward neural networks, games**

I. Introduction

Artificial intelligence (AI) methods are commonly used in game environments. Most of today's game environments include intelligent agents and having good intelligent agents is an important criterion for a game to be successful. In general, better intelligent agents make a game more challenging and entertaining for human players. Developing intelligent agents using classical AI methods generally solves the problems for simple game environments. What is expected from those agents is to make simple decisions based on limited numbers of possibilities. However, as the decisions that intelligent agents should make become more complex, it becomes harder and more expensive to develop high performance agents using classical methods. A first-person shooter game playing agent is a good example of complex agents. It should consider many factors, such as the positions of players, the weapons picked and the obstacles in map, in order to be able to pick the best or at least better action at a time. So, developing high level intelligent agents for this kind of complex problems is not simple.

Main problem of developing intelligent agents for complex game environments using classical AI methods is the work load. Classical intelligent agents follow some hard coded rules. However, writing rules that deal with too many criteria and make good decisions is hard and it requires too much effort. Machine learning (ML) overcomes the development and performance issues related to classical intelligent game agents. ML methods can be used in complex game environments to develop better performing game playing agents. Supervised learning (SL) is used to learn how to play a game. It can be basically defined as learning from examples or past experiences. In game environments, this corresponds to learning from gameplay data. Collecting gameplay data for learning is easy and the data consists of relationships that correspond to the rules for developing an intelligent agent. However, as previously stated, finding out these relationships is not a short and easy task. With the help of SL methods, one can generate learning agents that find out these relationships and approximate their behavior to the behavior defined in the data. So, SL makes agents capable of imitating someone's gameplay, which means playing a game like a human player is playing. The imitating agents can either play on behalf of their teachers or be opponents of other players. Using SL, intelligent agents can be created by not knowing or deriving any rules about the game. Performance levels of SL agents are determined by the success of learning applications and also by performances of the players from whom the examples are collected.

In this study, SL agents for a world-wide popular online game, namely HaxBall [1], were developed. HaxBall is an online browser game. It basically is a two dimensional football game and it's game dynamics are similar to air-hockey game. HaxBall is a real-time multiplayer game. Players are divided into two teams and try to score in time or score limited matches. Game environment of HaxBall is fully observable and continuous.

II. Related Work

AI methods are being used in game environments since 1950s. Initially, traditional board and card games (e.g. checkers, chess, backgammon) were tried to be solved using AI methods. Main purpose of those studies were to develop agents that play at highest level and to compete against best human

This work was supported by Scientific and Technological Research Council of Turkey under grant number 7130138.

978-1-5386-7893-0/18/$31.00 ©2018 IEEE

opponents. The resulting agents were evaluated by comparing them against world-wide champion human opponents.

Starting from mid 1990s, popularity of personal computers increased dramatically due to the new technologies and lower costs in computer industry. Personal computers with high computational and graphical performance were spread all over the world. Due to this revolution in home computing, new forms of computer games appeared in the game industry: video games. Video games consist more complex and interactive environments compared to their counterparts. Most of the video game environments include both human controlled agents and computer controlled (AI) agents. Many studies were done for developing intelligent agents for video games and some of them include successful applications of ML methods. In this study, artificial neural networks were used to develop intelligent game playing agents. There are previous studies of developing learning agents for video games using artificial neural networks in the literature.

Colin McRae Rally is a car racing video game series, which is being published by Codemasters beginning from 1998. In this game, neural networks were used to develop driver agents to compete against human drivers [2]. Driver agents of this game were created using gameplay data of human players. *Forza Motorsport* is also a car racing video game series. It is being published by Microsoft beginning from 2005. Objective of the developers of this game was to create human like computer opponents and they achieved it by training neural networks with human player data. However, they also added another feature to the game, which was named Drivatar. This feature made users able to create their own intelligent agents that imitate their driving styles [3]. *Soldier of Fortune* is a first-person shooter (FPS) video game, which was published by Activision in 2000. In a study by Ben Geisler [4], multilayer neural networks with backpropagation algorithm were used to develop expert game agents without detailed rule specifications. He collected the gameplay data of an expert Soldier of Fortune player, and used it to train the agents. *Motocross the Force* is a motorcycle racing video game published by JSTARLAB in 2009. Objective of its developers was to feature advanced physics simulation and artificial intelligence as well as creating an entertaining racing game. So, developers made computer controlled bikes improve as you play the game using artificial neural networks trained by backpropagation algorithm [5], [6]. *Quake II* is another FPS video game published by Activision in 1997. In Quake II, there exist multiple weapon options and players can switch the weapon they are using during the game, as it is in almost all FPS games. The default computer controlled agents in game have the capability to navigate in the map and to target and shoot their enemies. However, appropriatness of the picked weapon is not a concern for them. In a study [7], researchers collected gameplay data of human Quake II players and they used it to train neural networks that were successful decision makers to select appropriate weapon for a game situation.

III. HaxBall: The Game Environment

HaxBall was developed by an Argentinian indie game developer, Mario Carbajal, and released in 2011. It has hundreds of thousands of daily users. It is a real-time, two dimensional multiplayer football game with gameplay dynamics similar to air-hockey game. Players are divided into two teams; red team and blue team. Number of players cannot exceed 22. Just like real football matches, players have the purpose of scoring to opponent team's goal and not letting opponent players to score to theirs.

Screenshot of a HaxBall match is given in Fig. 1. Each player is represented by a circular character with his team's color and ball is the white circle. HaxBall is played with keyboard. Only five keys are available to players; four arrow keys and space key. Arrow keys make player's character move in corresponding direction and space key makes it shoot the ball. Players move the ball by pushing it with their characters and they either shoot it or not.

A. Simulated HaxBall Environment

Although the purpose was to develop learning agents for HaxBall, we could not use the real HaxBall environment for this study. The environment should have provided two crucial features. First, we had to be able to collect gameplay data from players. Second, we had to be able to observe the trained agents on the game to make experiments on them to measure their successes. Since the HaxBall environment does not have these features, a simulated HaxBall environment was developed to use it throughout this study.

The developed environment is a replica of the real HaxBall environment. Screenshot of this environment is given in Fig. 2. Although there are some visual differences between them, the simulated environment functions the same as the real environment.

B. Behavior Based Agents

In supervised learning applications, agents are made to learn by training them with labeled training data, which is gameplay data of HaxBall players in this application. One obvious way to collect data is to get it from real HaxBall players. However, considering the amount of data needed for training

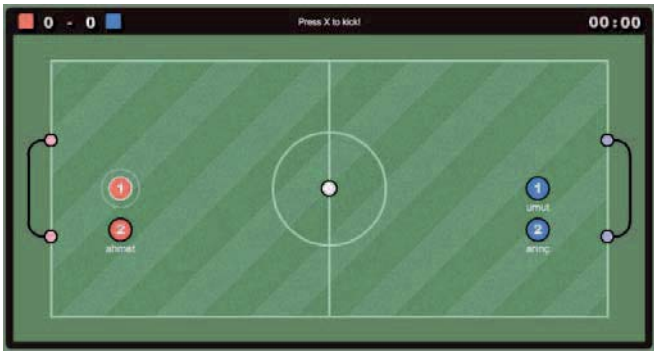

Fig. 1. Screenshot from HaxBall.

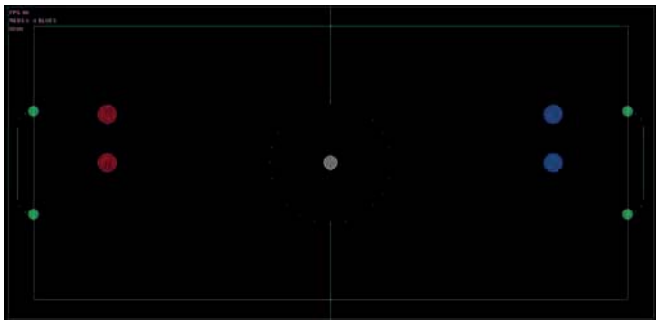

Fig. 2. Screenshot from simulated HaxBall environment.

and testing agents, it is time and resource consuming. Another problem associated with humans is that their gameplay is not standardized. A human game player is not guaranteed to play the same or even similar in two different sessions. This makes results of comparison between trained agents and human players less significant.

Considering all these problems, behavior based HaxBall agents were developed to carry out this study and getting acceptable results. These agents are both automated and standardized. They were used for both collecting training data and testing performances of trained agents. They were the baseline players in this study. The behavior based agents have behaviors like shooting, passing, dribbling, and pressing. They follow some hard coded rules to choose the behavior to execute according to the positions of the game elements (e.g. players and ball). At each time step, agents check the positions and pick a behavior.

IV. Supervised Learning

In this study, artificial neural networks were used to develop HaxBall playing supervised learning agents. Among several types of artificial neural networks with different features, multilayer feedforward neural networks were utilized and they were trained by backpropagation algorithm.

Multilayer feedforward neural networks do not have fixed structures. Although they have some strict rules for their internal structures, there are some configuration decisions to be made to create the neural network. Some of these decisions are number of input nodes, number of output nodes, number of hidden layers, and number of nodes in each layer. There exist configuration decisions also for backpropagation algorithm, like choosing an appropriate learning rate. For supervised learning applications using artificial neural networks and backpropagation algorithm, there are no strict rules in literature to pick the best configuration for the application. So in this study, learning process was repeated several times with different datasets and for each dataset with different configurations to find out the optimal network and algorithm configuration.

V. Datasets

The training data needed all along the study for supervised learning was collected from the behavior based agents developed. Many datasets with various sizes and state representations were used in this study. In general, there were two types of datasets in this study, namely the datasets with discretized values and the datasets with continuous values.

A. Datasets with Discretized Values

Although several datasets with discretized values were used in this study, they are very similar to each other. The main difference between them is the way they represent state of the game environment, i.e. what information they include and how precisely that information is collected. Datasets are composed of numerous training examples. Each training example consists of an input component and an output component. The input components describe the state of the environment at a time and output components show which actions were taken by the trainer for that state. While there is no difference between output components of all datasets, each dataset has its own kind of input components.

HaxBall players are allowed for only five actions: move left, move right, move up, move down, and shoot, respectively. Considering this, output components are described as vectors of five attributes. Each attribute corresponds to one of the actions. These attributes are either 1, which means that the action is active, or 0, which means that the action is passive. As an example, if trainer chooses to move up-right and not shooting the ball at a time, the output component of that training example is the vector $[01100]$.

An input component is a sequence of vectors, where each vector is an input item. Input items are vectors of a measure of angle and distance between two game elements. Number of items in input components of training examples in a dataset changes. It depends on the state representation decided for that dataset. For example, if a dataset has a state representation that considers ball's and teammate's positions with respect to the player, training examples in that dataset consist two item input components. An input item is also a sequence of vectors in its own. It is a combination of a vector that represents the angle value and another vector that represents the distance value. Both angle and distance values are discretized. Discretization intervals are not fixed. Each dataset has its own discretization intervals. Angle and distance values together specify the position of one game element with respect to the other one.

A sample discretization between two game elements is given in Fig. 3. In this sample, angle values are discretized with 45 degrees intervals and distance values are discretized with 4 intervals. Since the ball is in 45-90 degrees angle interval and in third distance interval, the input item for this sample is the vector $[010000000010]$. It is the combination of two one hot vectors for angle and distance values, $[01000000]$ and $[0010]$ respectively. Each attribute in angle vector corresponds to an angle interval from zero to 360. Each attribute in distance vector corresponds to a distance region from closest to farthest where last attribute is for every distance value that are not in other distance intervals.

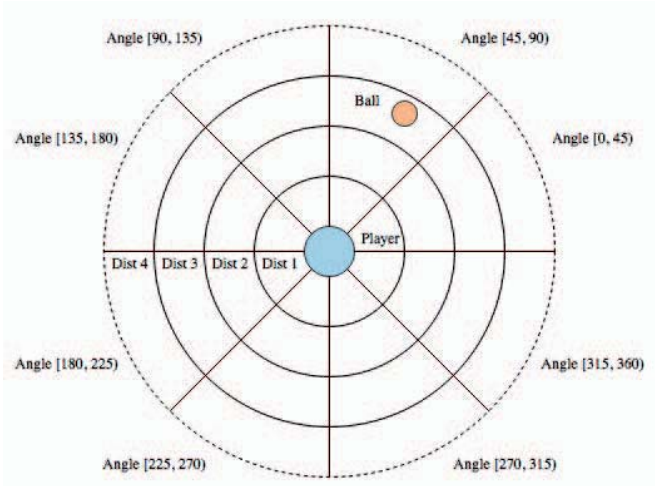

Fig. 3. Sample discretization between a player and the ball.

B. Datasets with Continuous Values

Datasets with continuous values are very simple compared to discretized datasets. Each training example in such datasets consists of an input component and an output component. There are no differences between output components of two types of datasets. The input components are also very similar. The input components of continuous datasets are composed of input items that hold information about game elements. However, instead of discretized and relative angle and distance values, each input item includes two real numbers that are coordinates of the game elements in the game environment.

VI. Experiments

In a HaxBall match, number of players in each team is not fixed. Each team can have at least 1 and at most 11 players. Football is a team sport, not an individual sport. If there are more than one players in a team, there needs to be collaboration among them. Unless teammates collaborate with each other, there occurs a chaos in the game. Defining rules to develop collaborative football agents is not an easy task and it gets harder as the number of players to collaborate increases. The behavior based agents developed in this study are capable of playing HaxBall matches between teams of at most two players. This means that they can either play solo or play in collaboration with an only teammate. Considering these limitations, two kinds of HaxBall playing agents were developed in this study. First one is solo playing agents that challenge against single opponents and the second one is collaborative playing agents that collaborate with single teammates and challenge against two opponents.

Performances of created HaxBall agents were evaluated according to two successive criteria. First one is mean squared errors (MSE) of neural networks trained for the agents on test data. Each dataset used in experiments was divided into two subsets, namely training data and test data. Training data was used for training the network and test data was used for measuring the performance of the network. The performance of a network was determined by computing the MSE on test data. Although training process was repeated several times with various configurations for each dataset, not all trained networks were allowed for following evaluations. They were eliminated by picking the one with lowest error on test data and only allowing that one for future evaluations.

Second criterion is gameplay performances of the agents. Gameplay performances were measured by arranging time limited matches between created HaxBall agents and behavior based HaxBall agents, which were baseline agents for this study. Results of these matches (win, draw, or lose) were noted and overall performances of trained agents were determined according to their success against baseline agents.

A. Experiments of Solo Playing Agents

The behavior based agents used throughout this study decide on actions by controlling nothing but the positions of all game elements. Each time, the agent perceives the current state of the game by checking the positioning of the game elements on the pitch and decides what action to do for that state of the game. For a solo playing behavior based agent, only criteria are the player's own position, the ball's position and the opponent's position. So, they were the only information included in the experiments done to develop solo playing agents.

In order to develop solo playing agents for HaxBall, four experiments were done in this study. Data required for experiments was collected from one to one matches between behavior based HaxBall agents. The dataset had 100000 samples. Three-fourth of it was used for training and the rest was for testing. Although information included in training data was same for all experiments, representations of the states (of the data) were different. Four different state representations were used in the experiments.

In the first experiment, state was represented by six real numbers that respectively were coordinates of player's position, ball's position, and opponent's position. As stated before, learning process was repeated several times with different configurations and only the one that provides lowest MSE was used for future evaluations. The details of the best configuration for this experiment is given in Table I.

In the second, third, and fourth experiments, datasets with discretized values were used. Each state consists of the relative positions of the ball, the player's goal, the opponent's goal, and the opponent. Since the player's relative position to itself does not provide information about the player's real position, goals' relative positions were added to state representation to make it include same information with others. These three experiments are the same experiments with different levels of discretization, namely low, medium, and high. The information about applied discretizations and the details of the best configuration for each of them are given in Table I.

B. Experiments of Collaborative Playing Agents

Training data required to develop collaborative playing agents was collected from two versus two matches between

TABLE I
NETWORK CONFIGURATION AND TRAINING DETAILS FOR THE EXPERIMENTS TO DEVELOP SOLO PLAYING HAXBALL AGENTS

Agent	Discretization Details	Network and Training Configuration	Results
Solo-1	-	Input layer: 6 linear nodes Output layer: 5 linear nodes Hidden layers: Two layers with 6 and 6 sigmoid nodes Learning rate: 0.5	MSE on test data: 0.1144 Correct classification rate on test data: 69%
Solo-2.1	90 degrees angle intervals 5 distance intervals	Input layer: 36 linear nodes Output layer: 5 linear nodes Hidden layers: Two layers with 10 and 10 sigmoid nodes Learning rate: 0.5	MSE on test data: 0.1483 Correct classification rate on test data: 58%
Solo-2.2	45 degrees angle intervals 5 distance levels	Input layer: 52 linear nodes Output layer: 5 linear nodes Hidden layers: Two layers with 10 and 10 sigmoid nodes Learning rate: 0.5	MSE on test data: 0.1261 Correct classification rate on test data: 61%
Solo-2.3	30 degrees angle intervals 5 distance levels	Input layer: 68 linear nodes Output layer: 5 linear nodes Hidden layers: Two layers with 10 and 10 sigmoid nodes Learning rate: 0.5	MSE on test data: 0.1157 Correct classification rate on test data: 64%
Coll-1	-	Input layer: 10 linear nodes Output layer: 5 linear nodes Hidden layers: Two layers with 20 and 20 sigmoid nodes Learning rate: 0.5	MSE on test data: 0.1434 Correct classification rate on test data: 63%
Coll-2.1	90 degrees angle intervals 5 distance intervals	Input layer: 54 linear nodes Output layer: 5 linear nodes Hidden layers: Two layers with 20 and 20 sigmoid nodes Learning rate: 0.5	MSE on test data: 0.2186 Correct classification rate on test data: 46%
Coll-2.2	45 degrees angle intervals 5 distance levels	Input layer: 78 linear nodes Output layer: 5 linear nodes Hidden layers: Two layers with 20 and 20 sigmoid nodes Learning rate: 0.5	MSE on test data: 0.1835 Correct classification rate on test data: 55%
Coll-2.3	30 degrees angle intervals 5 distance levels	Input layer: 102 linear nodes Output layer: 5 linear nodes Hidden layers: Two layers with 20 and 20 sigmoid nodes Learning rate: 0.5	MSE on test data: 0.1634 Correct classification rate on test data: 59%

behavior based HaxBall agents. The dataset had 200000 samples. Three-fourth of it was used for training and the rest was for testing. Collaborative playing agents have two more game elements compared to solo playing agents. They are the teammate and an extra opponent. Since a behavior based agent controls positions of all game elements for action selection, the teammate's and the other opponent's position information is also required for the experiments to develop collaborative playing agents besides the information used for solo playing agents.

Similar to developing solo playing agents, four different experiments with same information but different representations were done in this study for developing collaborative playing agents. State representations used in four experiments were also very similar to ones for solo playing agents. The only difference was the extra information included. The details of the best configuration for each experiment are given in Table I.

C. Evaluation of Agents

Finally, all of the agents developed so far were compared with behavior based agents. The behavior based HaxBall agents were the teachers of the supervised learning agents developed so far and also they were the baseline agents for evaluating performances of new agents. In order to test solo playing agents, they were compared to single behavior based agents. On the other hand, collaborative playing agents were tested by putting two of them in a team and comparing them to two behavior based agents. For testing each trained agent, 1000 matches were simulated. Table II shows statistics about the results of the 3 minute matches arranged between trained agents and baseline agents.

Considering the information in Table II, the most successful solo playing HaxBall agent trained was *Solo-1*. *Solo-1* was trained by using continuous state representation. Although *Solo-1* agent imitates the baseline agents, it achieved to win against them in more than half of the matches. The success of other solo playing agents trained (*Solo-2.1*, *Solo-2.2*, and *Solo-2.3*) was also not bad for imitating agents. To develop these agents, state representations with different levels of discretization were used. The results revealed that as the level of discretization increases the success of resulting agents also increases. The reason of this inference is that as the level of discretization increases the information loss compared to continuous state representation decreases. *Solo-2.3* was the most successful agent obtained by using discretized state and it achieved to win in quarter of the matches and also achieved not to lose in almost half of the matches.

Experiments done for testing collaborative playing agents were similar with the experiments for solo playing agents and the results were also similar. As it was for solo playing agents,

TABLE II
EVALUATION OF ALL AGENTS

Agent	Percentage of Wins	Percentage of Draws	Percentage of Losses	Average goals scored	Average goals conceded
Solo-1	54%	14%	32%	3.4	2.8
Solo-2.1	4%	5%	91%	0.7	3.5
Solo-2.2	20%	15%	65%	1.7	2.7
Solo-2.3	24%	21%	55%	2.5	3.8
Coll-1	46%	6%	48%	3.2	3.7
Coll-2.1	2%	4%	94%	1.9	5.5
Coll-2.2	18%	27%	55%	3.4	4.2
Coll-2.3	33%	11%	56%	3.5	4.1

the most successful collaborative playing agent trained was the one that trained by using continuous state representation, namely *Coll-1*. *Coll-1* succeeded to win almost half of the matches and also did not lose in more than half of the matches. The collaborative agents trained by using discretized state space also showed results similar to their counterparts. The most successful one of them was *Coll-2.3* which had the highest level of discretization compared to others. *Coll-2.3* won more than quarter of the matches and did not lose in almost half of the matches.

VII. CONCLUSIONS

The results of these experiments reveal that using artificial neural networks with backpropagation algorithm is an effective solution to develop supervised learning HaxBall agents. The preferred machine learning methods succeeded to overcome the difficulties of HaxBall environment caused by nonlinearity of the gameplay data and the continuous state space of the game.

Although HaxBall allows up to 11 players in a team, the behavior based agents developed for this study could not play in teams of more than two players. Considering the encouraging results obtained in this study, agents that collaborate with multiple players can be developed and the data collected from them can be used to train more talented agents as future work. The behavior based agents decide on the actions by regarding only the positions of the game elements. However, there exist other factors that can be considered too. For example, assuming that the ball has velocity and the player wants to catch the ball, it would be better if the player goes to the location where the ball will be after a short time instead of going to its current location. Although the behavior based agents of this study do not consider those factors, human players do. So, if we can create better baseline agents in future, we can get better supervised learning agents by imitating them.

To sum up, learning agents were developed in this study for a game with fully observable, continuous, and real-time game environment, namely HaxBall. The main purpose was to create agents that imitate human players and can either play the role of opponents or play on behalf of human players when they are absent. The results of the experiments reveal that the purpose has been successfully achieved in this study.

REFERENCES

[1] (2018) Haxball website. [Online]. Available: http://www.haxball.org/

[2] L. Galway, D. Charles, and M. Black, "Machine learning in digital games: a survey," *Artificial Intelligence Review*, vol. 29, no. 2, pp. 123–161, 2008.

[3] (2018) Video games and artificial intelligence. Microsoft Research. [Online]. Available: https://www.microsoft.com/en-us/research/project/video-games-and-artificial-intelligence/

[4] B. Geisler, "Integrated machine learning for behavior modeling in video games," in *Challenges in game artificial intelligence: papers from the 2004 AAAI workshop. AAAI Press, Menlo Park*, 2004, pp. 54–62.

[5] B. Chaperot and C. Fyfe, "Motocross and artificial neural networks," in *Game Design And Technology Workshop 2005*, 2005.

[6] ——, "Improving artificial intelligence in a motocross game," in *Computational Intelligence and Games, 2006 IEEE Symposium on*. IEEE, 2006, pp. 181–186.

[7] C. Bauckhage and C. Thurau, "Towards a fairn square aimbot–using mixtures of experts to learn context aware weapon handling," in *Proc. GAME-ON*, 2004, pp. 20–24.