Presented at the DLSU Research Congress 2015
De La Salle University, Manila, Philippines
March 2-4, 2015

Learning in Competitive Environments through Observation: A Data Mining Approach

Remedios de Dios Bulos, Mauro F. Arce, Allen Guarnes, Marvin J. Limson, and Michelle Ormoc

De La Salle University
remedios.bulos@dlsu.edu.ph or remedisdedios@yahoo.com

Abstract: In this study, we investigated the problem of learning about other agents in a competitive environment. For our application domain, we used a two-player Snake Game. We modeled the agent learner by observing the actions of a benchmark agent, a hard coded Snake Agent player which is designed and programed to be goal-oriented and whose main purpose is to win the game. Through the use of data mining algorithms, our learning agent determines the function or the underlying decision model. Data models are automatically extracted from the data collected. The data mining framework that we used in this study consists of three main modules, namely Data Observation, Model Extraction and Model Interpretation. The framework is designed to build a learning agent that behaves similarly and whose game performance is at par with the benchmark agent. We constructed several models of agent learners by applying various classification techniques. Particularly, we focused our investigation on classification techniques that produce decision trees (J48, SimpleCart, BFTree, REPTree and RandomTree) and rules (JRip and PART) because the decision trees and rules produced via WEKA can be easily parsed and automatically translated to C# program codes. Our experiments show that we have modeled an agent that learns from the actions of a benchmark agent through observation and using data mining techniques. Although overall the results indicate that the benchmark agent performed better than any of the algorithms investigated, the BFTree agent learner very closely follows its performance. On head to head matches, BFTree agent learner won more games than the benchmark agent.

Key Words: data mining; classification; machine learning; observational learning; WEKA

1. INTRODUCTION

Games have long been investigated in AI as models of multi-agent systems. While more research have been devoted to multi-agents learning in cooperative environments, research on learning in competitive settings has also continuously garnered some attention, and in this area, every now and then, games are being used as application domains or testbeds. Of late, there has been a surge of interest in using games as sources of new and challenging research problems (such as [Gorman, B., 2009]), especially in the research areas of Machine Learning and Data Mining. Our research attempts to explore

Data Mining as an approach towards building competitive agents that learn through observation.

Our research is motivated on the premise that agents adapt their behavior (or learn), based on the observations they make on the environment and on the actions of other agents. According to [Hu, J. et al., 2001], the effectiveness of learning depends not only on the learning method, but also on how much information is available to the agent. In the absence of observations, an agent has to rely on indirect evidence to formulate his decisions and guide his actions. Thus, learning becomes more difficult as well as limited when there is merely partial observation or incomplete information.

As defined in [Princeton University, n.d.,], observational learning is a type of learning that occurs as a function of observing, retaining and replicating novel behavior executed by others. It can hasten the learning process by making use of the data harvested from demonstrations of a given task [Gorman, B., 2009]. It differs from imitation learning since it does not require a duplication of the behavior exhibited by the model. For example, an agent learner which observes an unwanted behavior and its subsequent consequences will refrain from that behavior [Wikipedia, 2012]. Grounded on rational decision making, an agent leaner assumes that every agent is seeking to maximize its payoff. Consequently, it can be said that there exists a functional relation between the agent's actions and its local states. Through observation, an agent leaner can discover an underlying decision model that determines another agent's action. [Hu, J. et al., 2001]

In this study, we defined and solved our research problem of learning through observation by using a data mining framework. We used computer games as our application domain. Games provide a rich platform for observational learning. They generate vast quantities of raw behavioral data, which is easy to acquire and entirely devoid of noise as well. The data that can be harvested contain the exact locations of all agents and entities in the game world, their motions, internal states and all external influences. [Gorman, B., 2009] In particular, we used the two-player Snake Game as our testbed. In our competitive game environment, our agent learner is capable of observing other agents' local states and history of actions in terms of inputs and outputs. Through the application of data mining techniques, our agent learns the underlying function or decision model that maps those inputs to the outputs.

In the following sections, first, we introduce an instance of our application domain — a two-player Snake Game, which is a single-shot simultaneous-move game, where two interacting agent/snake players are selfish utility maximizers. We then discuss a summary of the salient findings of our previous work that motivated us to seek better ways of improving the performance of our learning agent. Next, we present the data mining framework solution to our research problem. We investigated a series of Data Mining classification algorithms that are appropriate to our application domain as well as possibly optimize the performance of our agent learner. We also performed some experiments to evaluate the learning performance of each algorithm. The results together with an analysis are presented in the tests and results section.

2. TWO-PLAYER SNAKE GAME

The modified two-player snake game is a two dimensional game, which consists of a walled rectangular playing arena and has a limited playing time. The game starts with the two snakes (players) placed opposite each other at both ends of the wall facing upwards. A snake is represented as a 1 pixel wide line with a head and tail; its size may increase or decrease, depending on the type of food it eats. Four types of food pellets with varying effects on the snake players randomly appear in the game arena. [Bulos, et al., 2007; Bulos, et al., 2009]

To win the game, a player has two objectives to meet: (1) stay alive and (2) accumulate points by eating the right type of food or performing some feats (series of risky moves). A player is destroyed when its size is reduced down to zero or when it is bumped by the opponent. Bumping occurs when a snake player's head collides with the opponent's tail, or when the player touches its own tail. The game ends when any of the following conditions occur: 1) the game time expires; 2 any player is bumped 3) any player is destroyed. [Bulos, et al., 2007; Bulos, et al., 2009]

3. PREVIOUS AND RELATED WORK

In [Bulos, et al., 2007], using the same two-player snake game as our testbed, we investigated the learning performance as well as examined the competitive co-evolutionary characteristics of two Machine Learning (ML) algorithms, namely, Best Response (BR) and Reinforcement Learning (RL). We created four types of agents/snakes (players), two of which were hard coded (Basic and Complex) and the other two were constructed using BR and RL algorithms. As the name suggests, the Basic agent possesses simple reasoning capabilities and is not goal-oriented. It just randomly moves around the game arena avoiding obstacles. Its sole purpose is to

avoid losing. On the other hand, the Complex agent is designed and programed to be goal-oriented and its main purpose is to win the game. We considered it as our benchmark agent.

The BR agent was created using the Best Response learning strategy. The Best Response concept is based on John Nash's "Nash equilibrium", the point where every player has selected the best response based on the opponents' own strategies [Nash, J., 1950]. By definition, Best Response strategies produce the most favorable consequences for players, given the strategies of others [Fudenberg, D., & Tirole, J., 1991; Gibbons, R., 1992]. In BR learning, agents have perfect knowledge of the consequences of their decisions and actions. They gather information, compute for a decision, and then act based on that decision. They calculate the best strategy to a particular game scenario by using the information they collected on an opponent's past behavior. They use multiple and even infinitely many observations to model their opponent. [Bulos, et al., 2007; Namatame, et al., 2004]

The RL agent was created using Reinforcement Learning technique. Reinforcement Learning (RL) is a learning method that uses the concept of a reward system. In RL, the agent can learn even in an unknown environment. RL is unsupervised and it is achieved based on the agent's own experiences and not through the use of examples or through instructions. It has two important characteristics that differentiate it from other learning methods. First, it employs trial-and-error. Through self-discovery, an agent chooses the most appropriate action(s) to be taken for specific situation(s). It maps situations to actions--so as to maximize a numerical reward. Second, it also applies the concept of delayed reward, in which a single action affects not only the immediate reward, but also all subsequent rewards on all subsequent actions and situations. Simply put, an agent avoids a smaller but more immediate reward in favor of a larger but more long-term reward that will be collected later. [Bulos, et al., 2007; Namatame, et al., 2004; Sutton, R. & Barto, A., 1998; Weinberg, M., & Rosenschein, J., 2004]

For this research, we recreated our previous study [Bulos, et al., 2007], and improved on it by conducting more in-depth and larger experiments. Round-robin matches (2,000 games per match) were played among the four agent/snakes players. In our findings, we have observed that both BR Agent and RL agents/snakes move in zigzag and circular motions, making them inferior to other agents/snakes if not trained correctly. Since they are learning agents, their performance progresses overtime. Simply put, they slightly improve their behavior by playing more games through training.

The results of the newly conducted round-robin matches are shown in table 1. Judging by results, we obtained similar findings with that of our previous work [Bulos, et al., 2007]. In both studies, we found out that the behavior of RL as a learning algorithm is comparatively better than BR. However, when compared against our benchmark agent (Complex Agent), the two machine learning algorithms (RL and BR) are inferior.

Because of the disappointing performance of the BR and RL algorithms against the Complex Agent, we felt the need to explore other solutions to adaptive learning agents in competitive environments. In [Bulos, et al., 2009], we attempted to explore a Data Mining (DM) approach/solution, particularly the use of Classification Mining. Our first task was to engineer the input data (observations) into a form suitable for classification mining. Primarily, this involved attribute selection which was intended to eliminate irrelevant attributes and consequently improve the performance of the learning scheme. To aid us in this endeavor, we created five types of DM agents, each exhibiting some specific kind of behavior.

Table 1. Performance Ranking of ML ALGOS

Snake	Win	Draw	Loss
Complex	5186	86	728
RL	2236	629	3135
BR	2088	401	3511
Basic	1628	608	3764

These are the Dumb agent, Playing Safe agent, Hungry agent, Picky agent and Complex-Like agent. Each type of DM agent is characterized (and distinguished from the other agents) by the types of attributes that were selected. These agents were created using J48 classification algorithm of WEKA.

To evaluate and compare their performance, each DM agent played 5,000 games each against three opponents namely, RL Agent (which in initial the study [Bulos, et al., 2007] was found to be better than Best Response), Basic Agent and Complex Agent. In our findings [Bulos, et al., 2009], we observed that effective classification rule mining does not solely rely on collection and processing of voluminous data to produce a good decision tree.

This study has shown that effective classification rule mining is heavily dependent on the type of attributes selected. Excess and irrelevant attributes may create unnecessary noise which may spoil the results of the analysis while too few attributes may yield incomplete information. We have observed that improper attribute selection causes the performance of a learning scheme such as J48 to degrade.

Based from the results of the experiments conducted in [Bulos, et al., 2009], the Complex-Like agent performed better than any of the other four DM agents. It won 88.18% of its 5,000 games against the RL agent, won 85.40% of its 5,000 games against the Basic agent, but won only 44.22% of its 5,000 games against the Complex agent.

Once again, because of the inferior performance of our data mining agent (Complex-Like agent using J48) against the benchmark agent (Complex Agent), we are motivated to seek other ways of improving its learning behavior. In the next section we present our improved data mining framework. We investigate a series of Data Mining classification algorithms (besides J48) that are applicable to our domain and that may enhance the learning performance of our agent.

4. GENRE PREDICTION USING SQL

In our two-player competitive game environment, the agents/snakes are capable of observing their opponents' local states and history of actions in terms of inputs and outputs. Through the use of data mining algorithms, our learning agent determines the function or the underlying decision model that maps those inputs to the outputs. The observations captured and stored during matches are used to discover useful patterns. Simply put, data models are automatically extracted from the data collected. The data mining framework that we used in this study is depicted in Fig. 1. It consists of three main modules, namely Data Observation, Model Extraction and Model Interpretation. The framework is designed to build a learning agent that behaves similarly and whose game performance is at par with the Complex agent.

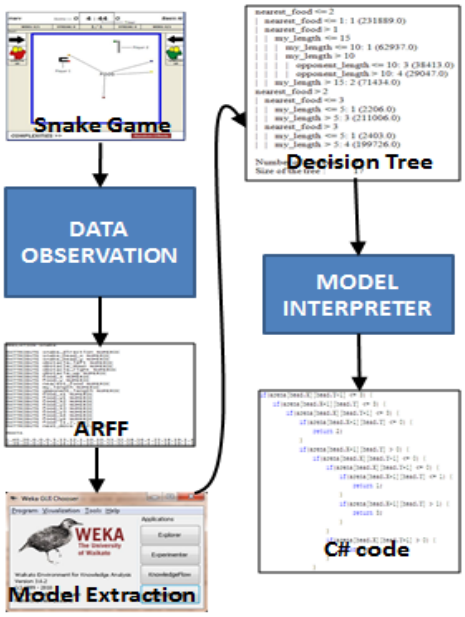

Fig. 1. Data Mining Framework

4.1 Data Observation

In our data mining framework, we assume that agents/snakes have perfect knowledge of their environment as well as the consequences of their decisions and actions. Such knowledge is obtained through observation. In the Data Observation module, the set of relevant data to be used in training phase (or learning step) is captured and stored. In building the training data, we observed a total of 24,997 matches between two Complex Agents. The Complex Agent is our benchmark agent and our research objective is to model an agent learner by observing the actions of a victorious (winner) Complex Agent. Unlike in our previous study [Bulos, et al., 2009], this time we restricted our observations (data collection) to those games won and considered only those positive actions that contributed to the victory. Data on games that were drawn or lost, as well as trivial actions, were considered irrelevant and thus eliminated from the training set.

According to [Witten, I. et. al., 2011], preparation of the training set for a data mining undertaking usually consumes the bulk of the effort invested in the entire data mining process. However, our application domain (two-player snake game) automatically generates vast quantities of raw behavioral data which is easy to acquire and entirely devoid of noise. In other words, our choice of application domain allowed us to do minimal data

pre-processing. Preparation of the training datasets was confined to attribute selection and creation of the ARFF file, which is a standard way of representing datasets that consist of independent, unordered instances and do not involve relationships among instances. [Witten, I. et. al., 2011]

4.2 Model Extraction

Essentially, building a decision model for our learning agent involves the application of data mining techniques using the dataset (ARFF file) constructed in the Data Observation module as training set. We used WEKA as our data mining workbench. It is open source software issued under the GNU General Public License. It has an unparalleled range of machine learning algorithms and related techniques. It now includes many new filters, attributes selection algorithms, and components such as converters for different file formats and parameter optimization algorithms. [Hall, M. et al., 2009; Witten, I. et. al., 2011]

Among the many data mining algorithms embodied in WEKA, we exclusively focused our investigation on classification techniques that produce decision trees and rule models. Rules and decision trees generated via WEKA can be easily parsed and converted to C# program codes. They constitute the decision model of the agent learner and the corresponding C# codes are used to construct/program the agent/snake player.

Classification techniques in data mining involve the analysis of a set of training data and construction of a model for each class based on the attributes of the given data. Usually, a decision tree or a set of classification rules are generated by the process. The objective of classification is to organize and categorize data into distinct classes. [Han, J. et al., 2012]

The WEKA data mining or machine learning algorithms that we used in our study are J48, SimpleCart, BFTree, REPTree, RandomTree, JRip and PART. The first five algorithms produce decision trees while the last two generate rules. The J48 algorithm is the implementation of C4.5 in WEKA. C4.5 is a classification algorithm that was developed by Ross Quinlan. It is an extension of the ID3. It builds decision trees from a training dataset by applying information entropy. [Witten, I. et. al., 2011]

SimpleCart employs the minimal cost-complexity pruning strategy. It is named after the CART (classification and regression tree) learner that initiated this strategy [Breiman, et al., 1984]. However it provides none of the other features of CART. In SimpleCart, the minimum number of instances per leaf, the percentage of training data used to construct the tree, and the number of cross-validation folds used in the pruning procedure can be set. [Witten, I. et. al., 2011]

BFTree builds a decision tree using a breadth-first expansion of nodes rather than the depth-first expansion used by standard decision tree learners (such as C4.5). It contains pre- and postpruning options which are based on finding the best number of expansions to use via cross-validation on the training data. While fully grown trees are the same for breadth-first and depth-first algorithms, the pruning mechanism used by BFTree will yield a different pruned tree structure than that produced by depth-first methods. [Witten, I. et. al., 2011]

REPTree generates a decision or regression tree utilizing information gain/variance reduction. It prunes the decision tree using reduced-error pruning. Optimized for speed, it only sorts values for numeric attributes once. Like C4.5, it handles missing values by splitting instances into pieces. In REPTree, the minimum number of instances per leaf, maximum tree depth (useful when boosting trees), minimum proportion of training set variance for a split (numeric classes only), and number of folds for pruning can be set. RandomTree constructs a tree that considers a given number of random features at each node without performing pruning. [Witten, I. et. al., 2011]

JRip is an algorithm for fast, effective rule induction. It implements RIPPER [Cohen, W. W., 1995], an acronym for repeated incremental pruning to produce error reduction. It also performs heuristic global optimization of the rule set. PART obtains rules from partial decision. It constructs the tree using C4.5's heuristics with the same user-defined parameters as J4.8. [Witten, I. et. al., 2011]

4.3 Model Interpreter

The Model Interpreter module takes a decision tree or set of rules (generated via WEKA) as input, parses it and then automatically converts it to C# program codes. This program code is embedded when constructing or programming the agent/snake player.

5. TESTS AND RESULTS

To evaluate and measure the performance of the classification algorithms, we conducted round-robin matches among them including the Complex agent, which served as our benchmark agent. We grouped the agent learners into two: (1) decision tree and (2) rules. Each match consisted of 2,000 games.

We ranked the performance of each algorithm (by number of points) and the rankings are shown in tables 2 (for decision tree algorithms) and 3 (for rules algorithms). We gave 3 points for a win, 1 point for a draw and no points for a loss. Among the decision tree learners, BFTree has the best performance followed by the J48. Both techniques registered higher number of wins than the Complex agent. The other decision tree learners were inferior compared to the Complex Agent. Among the rules learners, both JRip and PART ranked below the Complex Agent. JRip performed better than PART.

Table 2. Ranking of Performance of DECISION TREES Algorithms

Snake Type	Win	Draw	Loss	Points
BFTree	6739	513	2748	20730
J48	6702	482	2816	20588
Complex	6645	514	2841	20449
Simple Cart	6560	562	2878	20242
REP	986	826	8188	3784
Random	545	706	8749	2341

We then conducted round-robin matches among Complex agent, BFTree (winner in group 1), PART (which performed better than JRip in group 2), and Reinforcement Learning (RL) agent (which performed better than Best-Response). The Complex agent was ranked highest (see table 4) followed closely by BFTree.

However, looking at the head to head matches held between the Complex agent and BFTree, BFTree performed better. It won more games against the Complex agent (935 wins vs 926 wins).

Table 3. Ranking of Performance of RULES Algorithms

Snake	Win	Draw	Loss	Points
Complex	2425	360	1215	7635
PART	1516	417	2067	4965
JRip	1467	407	2126	4808

Table 4. Ranking of Performance of Group Winners

Snake	Win	Loss	Draw	Points
Complex	4032	269	1699	12365
BFTree	3902	318	1780	12024
PART	2522	367	2952	7933
RL	803	210	4987	2619

6. CONCLUSSION

In this study, we investigated the problem of learning about other agents in a competitive environment. We modeled the agent learner by observing the actions of a benchmark agent (Complex Agent); and then we used a data mining framework to determine the underlying decision model that the benchmark agent uses when mapping its inputs (observations) into outputs (actions). That is, we used our observations on the benchmark agent to train our agent learners.

We constructed several models of agent learners by applying various classification techniques. Particularly, we focused our investigation on classification techniques that produce decision trees (J48, SimpleCart, BFTree, REPTree and RandomTree) and rules (JRip and PART) because the decision trees and rules produced via WEKA can be easily parsed and automatically translated to C# program codes.

Our experiments show that we have modeled an agent that learns from the actions of a benchmark agent through observation and using data mining techniques. Although overall the results indicate that the benchmark agent performed better than any of the algorithms investigated, the BFTree agent learner very closely follows its performance. On head to head matches, BFTree agent learner won more games than the benchmark agent.

For our future research, we would explore the use of association mining to model our agent learners.

7. REFERENCES

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J., (1984), Classification and Regression Trees. 1984. Monterey, CA: Wadsworth.

Bulos, R. Amurao, R., de Jesus, J., Gallardo, A. and Moreno, M., (2007), Adaptive Co-evolution using Reinforcement Learning, Proceedings HNICEM, 2007.

Bulos, R., Cheng, T., Cua, J. and Opaco, C., (2009), Classification Rule Mining as a Learning Algorithm in Competitive Environments, Proceedings of the 11th Science and Technology Congress, 2009.

Cohen, W. W., (1995), Fast effective rule induction. In A. Prieditis, & S. Russell (Eds.), Proceedings of the Twelfth International Conference on Machine Learning, (pp. 115–123), 1995

Fudenberg, D., Tirole, J., (1991), Game Theory, 1991, Cambridge MA: MIT Press

Gibbons, R.,(1992), A Primer in Game Theory, 1992, Harvester-Wheatsheaf.

Gorman, B., (2009), Imitation Learning through Games: Theory, Implementation and Evaluation, Dublin: PhD Dissertation, 2009

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H., (2009), The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1., 2009, Official site: http://www.cs.waikato.ac.nz/ml/weka/

Han, J., Kamber, M. and Pei, J., (2012), Data Mining: Concepts and Techniques 3rd Edition, Morgan Kaufmann, 2012.

Hu, J. and Weliman, M. P., (2001), Learning about other agents in a dynamic multiagent system. Journal of Cognitive Systems Research, 2(1), pp. 67-69, 2001.

Namatame, A. Murakami, Y., and Sato, N., (2004), Co-evolutionary Learning in Strategic Environments. Co-evolutionary Learning in Strategic Environments, World Scientific, pp. 1-19, 2004 [online]

Nash, J., (1950), Equilibrium points in n-person games, Proceedings of the National Academy of Sciences 36(1):48-49, 1950.

Princeton University. (n.d.), Observational learning. [Online] Accessed 11 July 2014. http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Observational_learning.html

Sutton, R. & Barto, A., (1998), Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press. 1998

Weinberg, M., and Rosenschein, J., (2004), Best-Response Multiagent Learning in Non-Stationary Environments. Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on, pp. 506 – 513, New York, NY, USA

Wikipedia, Observational learning. (2012). [Online] Accessed 11 July 2014. http://en.wikipedia.org/wiki/Observational_learning

Witten, I. H., Frank, E. and Wall, A. H., (2011), Data Mining Practical Learning Tools and Techniques, Third Edition, 2011, MA, USA: Morgan Kaufmann