Server-side Early Detection Method for Detecting Abnormal Players of StarCraft

Kyung-Joong Kim[1] and Sung-Bae Cho[2]
[1] Dept. of Computer Engineering, Sejong University, Seoul, South Korea
[e-mail: kimkj@sejong.ac.kr]
[2] Department of Computer Science, Yonsei University
[e-mail: sbcho@cs.yonsei.ac.kr]
*Corresponding author: Sung-Bae Cho

Abstract

StarCraft is one of the most popular online real-time strategy games supporting multiple players but there are some undesirable players causing social problems to the community. For example, players can remove uncertainty of the game (fog-of-war) using map hacking tools, lose the game purposely to increase the rank of friends and disclose their team's information to others through external communication channels. It is necessary to detect automatically such players from their non-standard strategic behavior. In this work, we tried to build models to discriminate whether the current player's build orders and timings are from one of standard strategies or not in the early stage of the game. Experimental results on 1139 Protoss vs. Terran games showed that it is possible to predict unknown strategy with 10-fold cross-validation accuracy 99% (sensitivity 96%) using 15 minutes of game playing by K-Nearest Neighbor classifier. Especially, it achieved 97% accuracy (sensitivity 80%) only using information available for the first 6 minutes of the game issuing early warnings to administrator.

Keywords: StarCraft, Game Replay, Non-Standard Behavior, Machine Learning

1. Introduction

StarCraft is one of the most popular games played by multiple users through online. Because they play online without identity, some users try to use hacking tools, and cheating to get wins for high ratings in battlenet. It significantly decreases the willingness of users to play the games because they cannot win against someone with the hidden weapons. For online game companies, it is important to detect such players and give penalty to them [1]. However, the number of games played online is huge and it is not possible to monitor all of the games.

Recently, StarCraft has gained interest from computational intelligence researchers as a testbed for AI algorithms. For example, since 2009, StarCraft has been used as a competition tool for AI researchers [2]. The goal of the competition is to promote the devlopment of artificial intelligence tools for real-time strategic games. Although this is a quite challenging task, recent studies show the possiblity of playing the game autonomously and understanding other player's behavior [3].

It is not difficult to categorize StarCraft games played by users into one of predefined strategies by expert players. Furthermore, they can recognize the player's strategy in the early stage of the game with small number of units built.

This research was supported by Basic Science Research Program and the Original Technology Research Program for Brain Science through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0012876) (2010-0018948)

Based on their expertise, they can easily identify players with the hacking tools and cheating s. In case of the abnormal plays, the behaviors of the players should be different which is enough to be distinguishable by experts. For example, they play the game as if they have no obstracle to access the other player's information.

In this paper, we applied machine learning algorithm to detect non-standard strategic behavior in the early stage of the game for the StarCraft (Figure 1). From the expert players' game replays, it learns to classify the current status of the games into standard or not in terms of strategic behavior. If the player's behavior is recognized as non-standard one, it should be reported to administrator and the game is permanently recorded into game database.

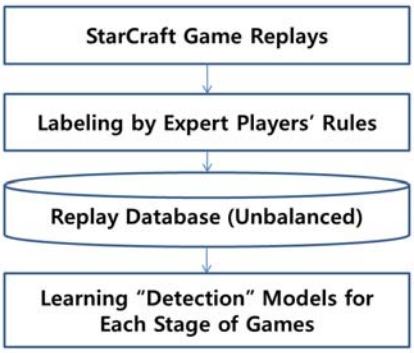

Figure 1. An overview of the proposed method

2. Related Works

Online games are popular nowadays but suffer from security issues [4]. Summeren classified several cheatings in online games into exploiting misplaced trust, collusion, abusing game procedures, exploiting machine intelligence, modifying client-side system infrastructure, exploiting a bug or loophole and internal misue. However, the detection of cheating is challenging because the chaters have complete control over their client computers running the cheating programs. Kaiser *et al*. used emulation-based anomaly detection approach for the cheating detection [5].

StarCraft is suffering from several cheating and hackings. For example, it is possible to lose their games continously to give the other a victory point. This is called as "win-trading" and a kind of "cheating by collusion. [6]" In case of "escaping" cheat, players purposely disconnect their communication to make their game as non-scorable one because losing a game will affect his rank [7]. "Map hack" lifts fog-of-war from the game's map and allows to play without uncertainty on the other players' activities [8].

There are several approaches to prevent the cheatings or hackings from StarCraft online playing. Kaminsky *et al*. proposed to use mouse biometrics to detect cheating players [8]. It exploited mouse moves including a button press, mouse drags to predict the identity of users for StarCraft. It can be applied to identify players with different account names and detect bots. Belicza developed a system to recognize StarCraft hackings from StarCraft replays [9][10]. It is based on predefined expert knowledge on the hackings and could be weak on new types of attacks.

3. Machine Learning for Server-Side Detection

3.1 Game Replays

Although they're stored in a binary format, it is possible to convert them to game logs using LordMartin Replay Browser[1]. An example log is shown in Table 1. In the logs, game states such as the amount of resources are not available for analysis because replays are viewed by performing a deterministic simulation based on the actions logged. However, it provides with sufficient information to analyze players' build orders.

Each replay was encoded as a feature vector, containing temporal features that record when the player expands different branches of the tech tree. If a unit type or building type is produced multiple times in a game, each feature describes when it is first produced. It is possible to construct two feature vectors for each game log where each vector represents a single player's actions for an entire game. If the game is played between *A* and *B*, there are two feature vectors f_A and f_B.

$$f(x)_P = \begin{cases} t & \text{time when } x \text{ is first produced by } P \\ 0 & x \text{ was not (yet) produced by } P \end{cases}$$

, where *x* is a unit type, build type or unit upgrade.

[1] http://lmrb.net

A subset of an example feature vector for a Terran player is shown in Table 2. In the game, second gas was not produced by the player.

Weber *et al.* [11] labeled the game logs with a strategy using rules based on analysis of expert play. Each race has different rule sets[2].

Table 1. An example of game logs from a StarCraft replay

Minute	Player	Action	Argument
1:02	Player1	Build	Barracks (Build)
1:07	Player2	Build	Spawning Pool (Morph)
1:15	Player1	Build	SupplyDepot (Build)
1:21	Player2	Build	Extractor (Morph)
1:40	Player1	Build	Barracks (Build)
2:14	Player2	Upgrade	Zergling Speed (Metabolic Boost)
2:20	Player1	Build	SupplyDepot (Build)
3:00	Player2	Morph	Lair

Table 2. A subset of an example feature vector (from a Terran player's feature vector for a Protoss vs. Terran match)

Attribute	Game Time
Depot	1:20
Barracks	2:05
Gas	2:40
Expansion	11:00
Second Expansion	15:11
Third Expansion	18:45
Fourth Expansion	0:00
Second Gas	0:00

[2] When you apply the rule set, you should be careful to deal with the attribute with 0:00. Although 0:00 is intended to represent that the building or unit is not (yet) constructed in the game, the rule set can recognize it is constructed earlier than others because the value is smaller than others'. In this work, the 0:00 is replaced with infinite to avoid such misinterpretation.

3.2 Performance Measurement of Detectors

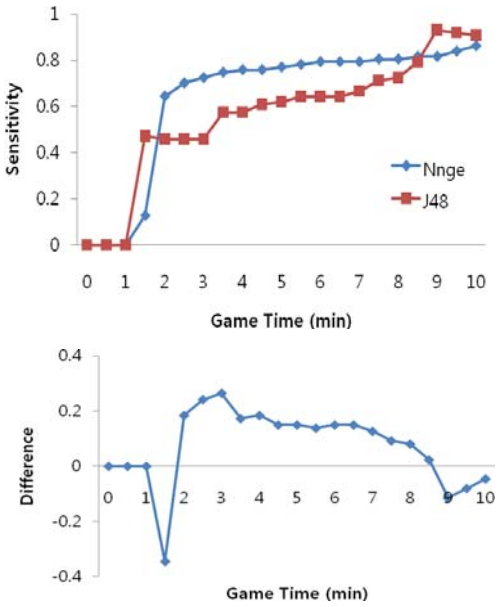

Figure 2. An example of sensitivity and their difference comparison between Nnge (Non-nested Generalized Examplars) and J48 (Decision Tree) classifiers on Protoss Vs. Terran games

The goal of machine learning algorithm is to classify the current player's strategy into standard or non-standard ("unknown") for each game time. It is natural that the number of samples of the "unknown" cases is much smaller than the one from standard one. Due to the imbalance of the data, the algorithm is likely to classify the "standard" cases very well but poor in the "unknown" samples.

In machine learning, sensitivity measures the percentage of actual positives (in this paper, "unknown" strategy) which are correctly classified. In Figure 2, NNge classifier shows about 80% sensitivity around four minutes. It means that the classifier fails to identify 20% of all players with unknown strategy at the time of four minutes. Because the game is usually played more than ten minutes, the time is not enough to see all the buildings and units that the player plan to construct.

In this context, we call the machine learning algorithm's ability as "foresight" on the opponent's strategy. Given two algorithms, one

can outperforms another in the early stage of the game but opposite in the last minute. It is necessary to measure the superity of algorithm for the problem. In this paper, we propose to use the integration of "sensitivity" over time as a measure for the comparison ("foresight"). Because the integral is computationally expensive, we approximate it using the sum over M sampling points. If the "foresight" of algorithm A is larger than one of B, it means that A is better than B.

As mentioned before, it is desirable to predict correctly in the early stage of the game, it is necessary to give different weights on the integration. In this paper, we define "Weighted Foresight" which gives high weight on the "sensitivity" of the prediction in the early stage of the game. In the weighted foresight, the parameter C controls the importance of early stage prediction. If C=0, it is the same with the normal foresight value. If C=1, the weight decreases linearly from 1 to 0. If C>1, the weight for the early stage is much higher than the one in the late one.

$$Foresight_A = \int_0^{GameTime} Sensitivity_A dt \approx \sum_{i=0}^{i=M} Sensitivity(i)_A$$

$$Weighted\_Foresight_A = \int_0^{GameTime} Sensitivity_A \times \left(\frac{(GameTime - t)}{GameTime} \right)^C dt$$

3.3 Machine Learning Algorithms

Rodriguez *et al*. proposed a method for generating classifier ensembles based on feature extraction [12]. The idea is to apply PCA (Principal component analysis) to create the training data for a base classifier. For each base classifier, the feature set is randomly split into K subsets and PCA is applied to each subset. As a result, K axis rotations take place to form the new features for a base classifier. Decision tree is used as a base classifier.

L. Breiman *et al*. proposed random forest combining "bagging" and the random selection of features in order to construct a collection of decision trees [13]. It is similar to bagging "decision trees" [14] but in the learning of tree, for each node of the tree, randomly chooses m variables and calculates the best split based on the variables.

Random committee builds an ensemble of base classifiers trained using a different random seed but based on the same data [15]. In this paper, base classifier is a random tree that constructs a tree that considers K randomly chosen attributes at each node.

4. Experimental Results

WEKA toolkit is used to implement the machine learning algorithms [15]. The paremeters of algorithm are set with the default values provided by the developers. The results are obtained from 10-fold cross-validation. The StarCraft game replays are modifed from the public database [11]. It is reformulated as binary classification problem by relabeling each game as one of "standard" and "unknown." In this paper, we use 1139 Protoss Vs. Terran game replays. The Protoss's strategy are labeled using expert rules ("Rule Set"). 56 attributes of each replay indicate the timing of construction of building and production of units. The number of replays labeled as "standard" is 1052 (92%) and "unknown" is 87 (8%).

"Rule Set" is a set of rules based on human expert knowledge and perfect to classify the strategy of game replay when there are enough information. However, it is quite weak to identify its strategy with limited information in the early stage of the game (Figure 3). On the other hands, NNge is consistently performs well ranging from early stage and the late time.

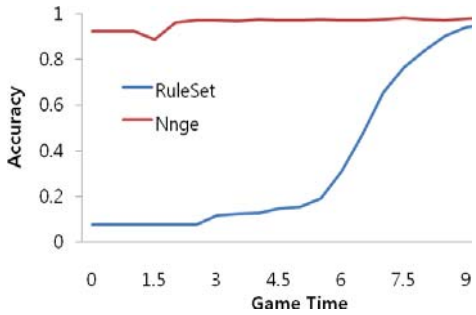

Figure 3. The comparison of accuracy between NNge and RuleSet

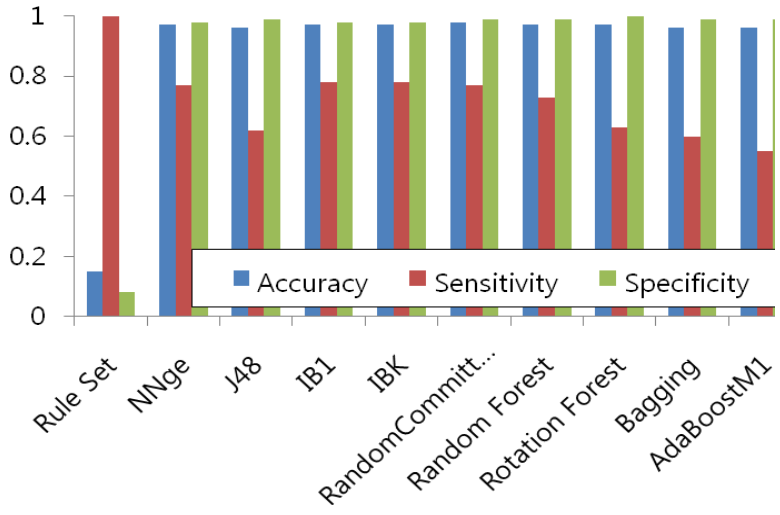

Figure 4. Accuracy, sensitivity, and specificity comparison of classifiers (Game Time = 5 minutes)

In Figure 4, the accuracy and specificity for the machine learning algorithms are near 1.0. However, the sensitivity is different for the classifiers. Although Rule Set is good for sensitivity but very poor for specificity. Based on the observation, the sensitivity is important to identify good classification algorithm for the problem.

Figure 5 shows the foresight and weighted foresight (C=5) for the algorithms. In the foresight, the classifiers (IBk, IB1, and NNge) inspired by k-nearest neighborhood ideas perfom well. If we give high weights on the prediction accuracy in the early stage of the game, the results are different. The most powerful classifiers are "Random Forest," "Random Committee" and "IBk." Two of them are from ensemble approach.

5. Conclusion and Future Works

In online multiplayer games, it is important to gurantee the fairness of the game. However, some players use hacking tools and external communication chanels to get benefit from the online games. The detection of the cheating is difficult and computationally expensive. In this paper, we try to propose machine learning approach to detect games with "unknown" strategies as early as possible to give alarms to administrator.

"Foresight" and "Weighted Foresight" measures are proposed to give insight on the performance of each classification algorithm for the problem. Experimental results on StarCraft show that the IBk and ensemble approach achieve the best performance in terms of the "foresight."

References

[1] K.-M. Woo, H.-M. Kwon, H.-C. Kim, C.-K. Kim and H.-K. Kim, "What can free money tell us on the virtual black market?" *ACM SIGCOMM*, 2011.

[2] http://eis.ucsc.edu/StarCraftAICompetition

[3] G. Synnaeve, and P. Bessiere, "A Bayesian model for opening predictions in RTS games with applications to StarCraft," *IEEE Conference on Computational Intelligence in Games*, 2011.

[4] R. van Summeren, *Security in Online Gaming*, Bachelor Thesis, Radboud University Nijmegen, 2011.

[5] E. Kaiser, W. Feng, and T. Schluessler, "Fides: Remote anomaly-based cheat detection using client emulation," *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 269-279, 2009.

[6] J. Yan, and B. Randell, "A systematic classification of cheating in online games," *Proceedings of ACM SIGCOMM workshop on Network and System Support for Games (NetGames)*, pp. 1-9, 2005.

[7] J. Yan, and H.-J. Choi, "Security issues in

online games," *The Electronic Library*, vol. 20, no. 2, pp. 125-133, 2002.
[8] R. Kaminsky, M. Enev, and E. Andersen, "Identifying game players with mouse biometrics,"http://abstract.cs.washington.edu/~miro/docs/mouse_ID.pdf.
[9] A. Belicza, StarCraft Hacking Recognition, http://bwhf.googlecode.com/files/Starcraft Hacking Recognition.pdf
[10] A. Belicza, Starcraft Broodwar hacker finder, anti-hack, replay analyzer-organizer and utility tool, http://code.google.com/p/bwhf/
[11] B. Weber, and M. Mateas, "A data mining approach to strategy prediction," *IEEE Symposium on Computational Intelligence and Games*, pp.140-147, 2009.
[12] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619-1630, 2006.
[13] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.
[14] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.
[15] I. H. Witten, E. Frank and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2011.

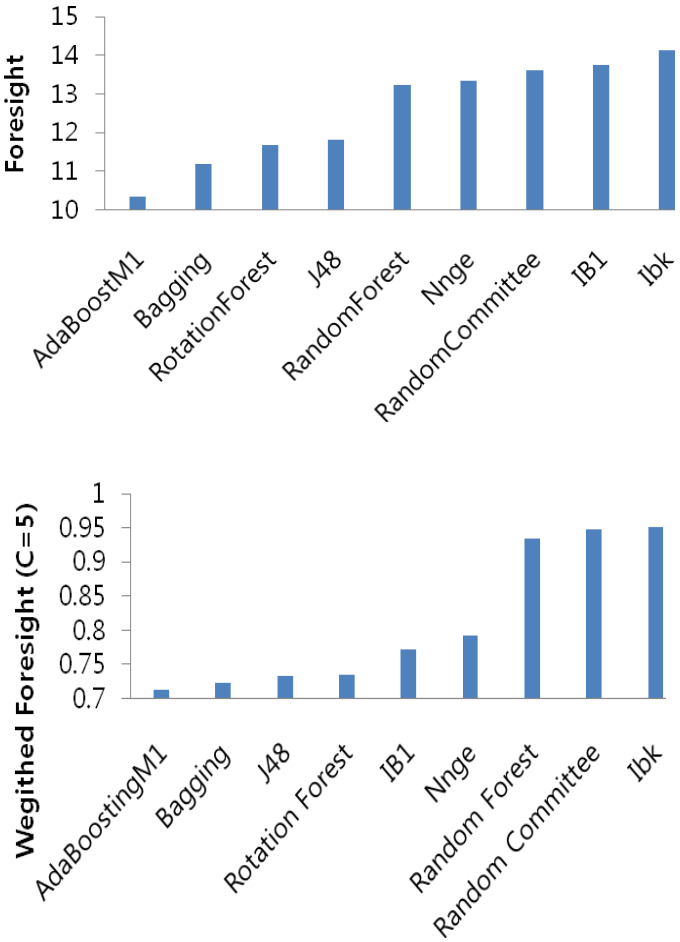

Figure 5. Comparison of machine learning algorithms