Strategy detection in real-time strategy games using machine learning

Martin ČERTICKÝ, Martin SARNOVSKÝ, Tomáš VARGA

Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovak Republic

martin.certicky@tuke.sk, martin.sarnovsky@tuke.sk, tomas.varga@student.tuke.sk

Abstract — **Despite excessive amount of research in the field of autonomous RTS gameplay, changes in strategies are often ignored leading to non-optimal results. Agents playing RTS games often use particular strategical decisions in order to cover as many gameplay scenarios as possible. In this paper, we focus on creation a dataset of atypical strategies and using machine learning methods for detection of these strategies during the gameplay. Such information could be used to predict the strategies before they occur, or to correspond witch adaptive behavior able to answer them. We approached the strategy detection in StarCraft game using a set of classifiers trained on data obtained from the various replays. Due to lack of proper datasets available to solve such tasks, the dataset was created and annotated to cover four selected strategies for our experiments. Binary classification models were trained to detect each particular strategy and evaluated in a set of replay data using cross-validation technique. Then, the overall platform architecture to train the models, export them and use in run-time during the gameplay was designed. Models which performed the best were then applied in the games to detect the covered strategies in replays or matches by bots or human players**

Keywords — **Data analysis, Machine learning, Classification, Starcraft**

I. Introduction

The field of video-games is ever expanding with many sub-genres available. Although there are a wide variety of existing types, a lot of them share the same basic principles. With the evolution of technology, the games methodology has to evolve as well. This is not different with RTS (real-time strategy) games which first came in the early eighties, and are still managing to keep their popularity, sales and player base. RTS games as a genre of video games in which players manage economic and strategic tasks by gathering resources and building bases, increase their military power by researching new technologies and training units, and lead them into battle against their opponent(s), serve as an interesting domain for Artificial Intelligence (AI) research. Because of the nature of genre, the games are dynamic, fast, ever-changing and are one of the most played games throughout the history of video-games.

StarCraft has more than 180+ variations of "typical" strategies, that are hard to identify prior to countering it during gameplay. Professional players spend years practicing their skills and studying many strategies while the casual players rarely choose optimal answer during real gameplay. Despite excessive amount of research done in the field of automated RTS gameplay [1, 2], subtle changes in strategies are often ignored leading to non-optimal results. Researchers often consider obvious strategical decisions in order to cover as many gameplay scenarios as possible. On the other hand, numerous papers deal with different aspect such as building of strategy model [4], prediction of build orders [5, 6, 7], or units micromanagement [3, 9]. In this paper, we focus on creating a dataset of atypical strategies which are often overlooked and using machine learning methods for creating adaptive behavior able to answer them. If the RTS games, especially StarCraft have that many strategies, is it possible to classify or predict them based on real-time data, with simple system to add another strategy later?

To correctly predict the player's strategy during the StarCraft match, models for detection of these strategies must be trained, evaluated and then used in the runtime. Several prediction methods were already applied for build order or strategy prediction [13, 14, 15]. We decided to approach the strategy detection as a predictive modeling task. Our approach will be based on training of a set of binary classification models on top of the data obtained from the replays. We decided to choose the binary classification approach from several reasons. When expanding the model to the new strategies, such approach presents more flexible solution, as it is not needed to re-train and re-deploy the model (only a new model for particular strategy will be added). Also, the performance of binary models should be better as it is possible to tune the features and model parameters more specific to particular strategy. Following sub-sections describe the data understanding, pre-processing and modeling phases of the model building process [16].

II. DATA PREPARATION AND MODELS TRAINING

For the problem discussed in this paper, there wasn't a relevant annotated dataset among many datasets available [10, 11, 12]. Therefore, we needed to construct the dataset from available replays and select the relevant attributes for training of the models. The source of the data were replays of various matches during the SSCAI Tournament. The replays consisted of matches performed by various bots. Over 2961 replays were collected, each of them contained two custom bots made by various people in 1vs1 match. To get the data needed for training of the models, we used ScExtractor tool. The tool extracted all relevant information out of the selected replay files. Then, we developed a script, which plays every match again in real-time, frame by frame, in order to gain complete picture - record every possible state of any unit and any event which occurred during the game. Such database comprised complete game data of hundreds of replays and resulting dataset size was more than 7.6 GB.

A. Data preprocessing

In order to reduce the data size, we selected the tables and attributes relevant to discussed problem. Also, we processed the data using a set of functions to enhance them with specific attributes related to the player's race or types of units built. After that, the data were aggregated based on 5 second interval (cca 240 frames). Resulting dataset was then annotated using OpenBW replay-viewer module. When the particular strategy was occuring during the match, the corresponding records (target attributes) were flagged to value 1, until bot's strategy changed to something else. Attributes of the final dataset are listed in Table 1.

PlayerReplayID	Replay ID
AC_FRAME	Frame number
Race:	Player's race
NumberOfBuildings:	Buildings constructed
NumberOfWorkers:	Workers trained
NumberOfAttackUnits	Attack units trained
NumberOfAttacks:	Attacks completed
RatioAttackToNon:	Ratio attack actions to non-attack actions
exp_feature1:	Protoss_Forge is built?
exp_feature2:	Protoss_Pylon is built?
exp_feature3:	Protoss_PhotonCannon is built?
exp_feature4:	Is scout (unit first tagged as a scout) in the group?
exp_feature5:	Is scout near the enemy base?
exp_feature6:	Zerg_SpawningPool is built?
exp_feature7:	Zerg_Extractor is built?
exp_feature8:	Zerg_Zeglings are trained?
exp_feature9:	How many Protoss_Gateway are built?
exp_feature10:	Protoss_Assimilator is built?
exp_feature11:	Protoss_CybernaticsCore is built?
exp_feature12:	Protoss_Zealots are trained?
exp_feature13:	AttackUnits near the enemy base?
Results1:	Strategy 1 - Cannon rush active?
Results2:	Strategy 2 - Zergling rush active?
Results3:	Strategy 3 - 2-gate active?
Results4:	Strategy 4 - 3-gate active?

Table 1 - Attributes used in created dataset

B. *Models training*

To solve the problem defined within this paper, we picked three models different classification models to compare; Random Forests, Naive Bayes and Gradient Boosting Tree, in order to get different results. Each model was trained to solve the binary classification task to predict one of the target attributes (*Results1-4*). We used the implementation of the models from the sci-kit learn python library. Feature selection was also performed for each of the trained models. As we used Random Forests, we extracted features importance from the initial model results. We used the data to select the attributes, which were not relevant to train the model for particular strategy. In case for Cannon Rush, we decided to remove the attributes *Race* and *NumberOfWorkers* from the training data since they either lack importance or are self-explanatory (Cannon Rush is possible only when the opponent's race is Protoss). Similar approach was used to reduce the feature space for other strategies. For training of the models for particular strategies, we used data from 108 replays which corresponded to 17778 records. During the model training, we used *GridSearchCV* method to examine the different combination of model hyper-parameters in order to find the best model with respect to the specified metrics.

III. Experiments and evaluation

For evaluation purposes, we used standard metrics used to specify the model quality such as precision and recall. Those were specified as the main factors during the hyper-parameters tuning and were used to evaluate the final models on the testing set. For training and evaluation of the models, we used 10-fold cross-validation approach.

Strategies	Models			
	RFT		GBM	
	P	R	P	R
Strategy 1	0.690	0.756	0.734	0.742
Strategy 2	0.930	0.832	0.992	0.832
Strategy 3	0.892	0.673	0.980	0.730
Strategy 4	0.928	0.860	0.957	0.886

Table 2 - Precision and recall of trained models

Table 2 summarizes the performance of the models Naive Bayes model is not included as its results were much worse, when comparing to Random Forests and GBM models. As it is can be seen from the results, GBM models performed well in each task. By incorporating the metrics, we were able to compare the different models. After closer consideration, we decided to use GBM models for several reasons. GBM gave almost constantly the best results in each considered metric on both datasets. he best GBM models were trained using those parameters: ∗ Strategy 1 model: Maximum features: 10, learning rate: 0.01, max. depth: 5, min. samples in the leaf: 10 ∗ Strategy 2 model: Maximum features: 9, learning rate: 0.01, max. depth: 7, min. samples in the leaf: 50 ∗ Strategy 3 model: Maximum features: 7, learning rate: 0.01, max. depth: 4, min. samples in the leaf: 1 ∗ Strategy 4 model: Maximum features: 9, learning rate: 0.01, max. depth: 9, min. samples in the leaf: 20 We also considered time constraints - time to build the model and time to predict the newly arriving instances (in run-time). From that perspective, both Random Forests and GBM models achieved similar performance results, therefore running time was not considered as important feature when choosing the model. Trained models then were deployed in the proposed system (see section IV. Proposed architecture). System obtains live data from the actual replay or game (played by both, bot or human player) sends it to the server which computes prediction for each of trained strategies. Results are retrieved to the client and players (or users watching a replay) are being notified about detected strategy. Integration of proposed solution was tested on various replays and game matches involved players and multiple bots.

IV. Conclusion

This paper introduced a system to gather the data from StarCraft games and to detect the player's strategy using machine learning models. For evaluating the system, predictive models were trained using collected game data and used to identify the chosen strategy in real-time. Our approach is based on training of binary classifiers for each covered strategy. This enables extension of the system with to ability to use different models to identify other strategies (not covered in our use cases), or to update the existing models with more precise ones. Designed system also enables to deploy such models and use them on live data during the real-time gameplay. We also investigated enhancement of the particular strategy identification with finding sequential patterns characteristic for each particular strategy or to combine the detection system with case-based reasoning approaches in order to create recommendation system which would advise the player or agent a best counter-strategy for actual situation.

Acknowledgements

This work was supported by Slovak Research and Development Agency under the contracts No. APVV16-0213 and No. APVV-015-0731.

References

[1] D. Churchill, A. Saffidine, and M. Buro, “Fast heuristic search for rts game combat scenarios.” in AIIDE, 2012, pp. 112–117.

[2] Z. Wang, K. Q. Nguyen, R. Thawonmas, and F. Rinaldo, “Monte-carlo planning for unit control in starcraft,” in Consumer Electronics (GCCE), 2012 IEEE 1st Global Conference on. IEEE, 2012, pp. 263–264.

[3] M. Certick`y and M. Certick`y, “Evolving reactive micromanagement controller for real-time strategy games,” in Proceedings of Scientific Conference of Young Researchers, 2015, pp. 225–228.

[4] J.-L. Hsieh and C.-T. Sun, “Building a player strategy model by analyzing replays of real-time strategy games,” in Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on. IEEE, 2008, pp. 3106–3111.

[5] B. G. Weber and M. Mateas, “Case-based reasoning for build order in real-time strategy games.” in AIIDE, 2009.

[6] M. Certick`y and M. Certick`y, “Case-based reasoning for army compositions in real-time strategy games,” in Proceedings of Scientific Conference of Young Researchers, 2013, pp. 70–73.

[7] N. Justesen and S. Risi, “Continual online evolutionary planning for in-game build order adaptation in starcraft,” in Proceedings of the Genetic and Evolutionary Computation Conference. ACM, 2017, pp. 187–194.

[8] M. Certicky, “Implementing a wall-in building placement in starcraft with declarative programming,” arXiv preprint arXiv:1306.4460, 2013.M. Matzopoulos, “Dynamic Process Modeling: Combining Models and Experimental Data to Solve Industrial Problems,” in *Process Systems Engineering*, Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA, 2011, pp. 1–33.

[9] A. Shantia, E. Begue, and M. Wiering, “Connectionist reinforcement learning for intelligent unit micro management in starcraft,” in Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011, pp. 1794–1801.

[10] G. Synnaeve, P. Bessiere et al., “A dataset for starcraft ai & an example of armies clustering,” in AIIDE Workshop on AI in Adversarial Real-time games, vol. 2012, 2012.

[11] G. Robertson and I. D. Watson, “An improved dataset and extraction process for starcraft ai.” in FLAIRS Conference, 2014.

[12] Z. Lin, J. Gehring, V. Khalidov, and G. Synnaeve, “Stardata: A starcraft ai research dataset,” arXiv preprint arXiv:1708.02139, 2017.

[13] B. G. Weber and M. Mateas, “A data mining approach to strategy prediction,” in Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on. IEEE, 2009, pp. 140– 147.

[14] H. Park, H.-C. Cho, K. Lee, and K.-J. Kim, “Prediction of early stage opponents strategy for starcraft ai using scouting and machine learning,” in Proceedings of the Workshop at SIGGRAPH Asia. ACM, 2012, pp. 7–12.

[15] H.-C. Cho, K.-J. Kim, and S.-B. Cho, “Replay-based strategy prediction and build order adaptation for starcraft ai bots,” in Computational Intelligence in Games (CIG), 2013 IEEE Conference on. IEEE, 2013, pp. 1–7.

[16] R. Wirth, “Crisp-dm: Towards a standard process model for data mining,” in Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining, 2000, pp. 29–39.