Improving StarCraft II Player League Prediction with Macro-Level Features

Yinheng Chen, Matthew Aitchison, and Penny Sweetser

The Australian National University, Canberra ACT 2601, Australia

Abstract. Accurate player skill modelling is an important but challenging task in Real-Time Strategy Games. Previous efforts have relied strongly on micromanagement features, such as Actions Per Minute, producing limited results. In this paper, we present an improved player skill classifier for StarCraft II that predicts, from a replay, a player's exact league at 61.7% accuracy, or within one league at 94.5%, outperforming the previous state of the art of 47.3%. Unlike previous classifiers, our classifier makes use of a macro-level measure of economic performance, called Spending Quotient, which we demonstrate to be an important part of accurately predicting player skill levels.

Keywords: Player Modeling · Real-Time Strategy Games · Machine Learning.

1 Introduction

Matching challenges in a game to a player's skill level is an essential factor in the player's enjoyment of the game [20]. In online games, a ranking system is used to attempt to match players with other players of appropriate skill. However, some players attempt to subvert this system. Players do this either via 'smurfing', whereby high-ranked players create secondary accounts in order to play against low-ranked players [6]. Or, conversely, by 'boosting' which involves low-ranked players asking high-ranked players to play their accounts to increase their rank artificially [8].

To address this problem, a good skill detection model is required. One which can identify these players, as well as any players who might be unintentionally mismatched to their current league (e.g., a player who is returning to the game after a long period or someone playing on a friend's account). Because the person playing the account can not be verified, player skill modelling is more complex than simply analysing the account's win/loss record. There are also important use cases for skill detection in offline games, where a game may need to adjust dynamically to the player's skill level [7]. Adapting challenges to the player's increasing skill level is another important aspect of player enjoyment [20].

Research in Real-Time Strategy (RTS) games has been an important area of interest for AI researchers since Buro's call for AI research in RTS games in 2004 [4]. Despite the increased interest in RTS games, one research area that has been relatively undeveloped in RTS games is player skill modelling, especially when

applied to AI players. RTS games provide some unique challenges, such as partial information, real-time decision making, and trading immediate versus future payoffs. For this reason, assessing the skill level of a player can be extremely challenging.

When constructing a player model, it is critical to identify the features that most strongly predict the player's level [22]. We examine how a player's skill level can be identified from their replay data in RTS game StarCraft II and aim to determine which are the most relevant features in predicting player skill. StarCraft II, developed by Blizzard Entertainment and released in 2010, has become an increasingly important testbed for AI algorithms due to its challenging environment and well-supported API [21]. In this paper, we report on research in which we constructed a model to predict a player's skill level. Our model extracts useful features from 1-vs-1 replay data and uses them to predict the league placement of each player.

2 Real-Time Strategy Games

A considerable amount of recent AI work in games has focused on complete information, limited action-space games, such as Chess [17], Go [18] and many of the Atari games [11]. However, real-world problems are often only partially observed and have large, sometimes infinite, action spaces. RTS games can help to bridge the gap between games and the real world by providing more challenging environments in which to demonstrate the abilities of AI algorithms. RTS games require the player to control multiple units and make difficult decisions about strategic trade-offs under incomplete information. Due to these challenges, advancements in AI in RTS games has progressed relatively slowly compared to other types of games. Standard RTS games involve three main components:

- Units - the most important component for a player to achieve the winning goal in most RTS games. Players are required to recruit and control their units to defeat their opponent and win the game.
- Collecting Resources and Resource Management - in most RTS games, resources can be used to construct buildings, recruit units, and develop technology. Players use their workers (i.e., a type of unit) to collect resources.
- Research - players research new technologies to improve the quality of their units and increase their resource collection rate. Technology often involves trading a short term cost for a long term reward. Some units may have technological prerequisites making advancement critically important to building a powerful army.

2.1 StarCraft II

StarCraft II (SC2) is an RTS game developed by Blizzard Entertainment and a sequel to StarCraft and its expansion, Brood War. SC2 was released in 2010 and is still one of the more popular RTS games played today, with an active

player base and professional eSports tournaments. While similar in many ways to other RTS games, SC2 is fairly unique in its asymmetric races. Players must choose between three races, Terran (humans), Zerg, and Protoss. Each race has different units, buildings, and game mechanics. For example, Terran (see Figure 1) can fly their buildings around the map, while Zerg is limited to building only on 'creep' that spreads slowly from their base. The economy in SC2 is based on two types of resources, Minerals and Vespene. These resources must be collected by workers and are in limited supply. Often players will need to expand to higher risk areas on the map to secure these resources. Blizzard Entertainment, along with DeepMind, provide support for AI research in SC2 via the StarCraft II Learning Environment (SC2LE) [21].

The built-in AI in the SC2 game has ten hard-coded difficulty levels. Much of the existing work on SC2 AI uses the built-in AI as a tool to test and develop their algorithms [9, 16, 19, 13]. This is due to the built-in AI being consistent, fast to compute, and providing a significant level of challenge. Early reinforcement learning based AI could only tie against the easiest built-in AI [21]. However, later work learned to exploit the built-in AI's weaknesses and can now consistently defeat the hardest level AI.

Fig. 1. A Terran player playing StarCraft II.

2.2 Ranking Players in StarCraft II

In SC2 ranked games, players are divided into seven leagues according to their ability: Bronze, Silver, Gold, Platinum, Diamond, Master, and GrandMaster. Initial placement into a league is by unranked placement games, after which a

player can ascend or descend the leagues based on their performance. There are 300,000 active players in SC2 ranked games, with most players' leagues being between Silver and Diamond [2].

A player who is ranked in Gold league is roughly equivalent to that of Level-10 built-in AI. This means the performance of most reinforcement learning based AI is below that of the average ranked human players' performance. The Grand-Master rank is extremely rare. Blizzard Entertainment limits the number of GrandMasters per region[1], ensuring the total number of GrandMaster players in the world does not exceed 800. As a result, only 0.27% of players can be GrandMaster in ranked matches.

SC2 uses a Match Making Ranking (MMR) system to track players' ability in ranked matches. Players earn MMR when they win a match and lose MMR when they lose a match. Once a player's MMR reaches a threshold, they will be upgraded to the next league.

3 Related Work

Most of the research in SC2 AI has been about developing AI algorithms to play the game well. Meanwhile, there has been only a small amount of work done on predicting players' skills. Early work on the problem achieved a weighted average accuracy of 47.3% [3], outperforming the most common class prediction baseline by using a variant of a Support Vector Machine on a training set of 1,297 replays. Player performance in the original StarCraft has also been investigated, with Gradient Boosting Regression Trees and Random Forests being used to create six classifiers for each race pairing [14]. However, this research predicted the winning player rather than each player's league. In parallel with the work on player modeling in RTS games there has also been work on assessing skill levels in the related Multiplayer Online Battle Arena (MOBA) games. Which skill factors are strong predictors for match outcome vary across different MOBA games, reinforcing that features should be game specific [5].

The challenge of dynamically adapting game difficulty to a player's ability is closely related to player modelling. Previous research has inferred difficulty curves from player-vs-level match data using win/loss data [15]. While this work primarily relates to adjusting player-vs-level difficulty, it could be adapted to player-vs-player matchmaking. In some situations, it may be preferable to design experiments to analyse player skills (e.g., by manipulating the game environments). The PsyRTS platform, a web-based toolkit for RTS games, allows for this, and is able to leverage existing crowd-sourced platforms [12]. However, experiments must be created ahead of time and cannot take advantage of the large set of exiting replay data.

[1] Blizzard Entertainment divides their servers into European, American, Korean, Oceanic, and Chinese regions.

4 Method

In order to investigate the strengths of various machine learning algorithms on this problem we trained models using five different algorithms. Each model was trained using features (see Section 4.3) extracted from a set of 4,917 replays, with the algorithm's hyperparameters tuned according to a 5-fold cross validation score. The best model from each algorithm was then evaluated on a holdout test set according to both its average weighted accuracy and F_1-score.

4.1 Data Collection

Since there is no established SC2 replay dataset, we constructed one from recent replays. A total of 4,917 replays were collected from three websites[2]. These replays were uploaded by global players, who posted their games online. Average features over time were extracted from 1-vs-1 replays. With each replay providing information about two players.

Replays that lasted less than 2-minutes or data from unranked players were excluded, giving 4,114 player data-points distributed as shown in Table 1. The relatively few instances of Bronze and Silver players is likely due to new players being less likely to upload replays than more experienced players. Due to the low number of data-points for these categories, we excluded the Bronze and Silver league from the model. During prepossessing, all features were normalised to between 0 and 1. After filtering, the dataset contained 3,981 player data-points. These replays were then split randomly between a training set of 2,985 replays and a holdout test set of 996 replays.

Table 1. Distribution of players by league the dataset.

League	Number of Players
Bronze	12
Silver	128
Gold	500
Platinum	1,355
Diamond	1,685
Master	357
GrandMaster	244

4.2 Evaluation

We selected and evaluated models based on their F_1-score, which we calculated as

$$F_1 := 2 \times \frac{P \times R}{P + R}$$

[2] https://sc2replaystats.com, https://gggreplays.com/matches and https://lotv.spawningtool.com/replays

where P is the precision, and R the recall of the model for a given league. An F_1 score of 0 indicates a very poorly performing model, whereas a score of 1 indicates perfect prediction. We performed an average of the F_1-scores for each league, weighted by the number of examples in that league. We also included the weighted average accuracy for reference,[3] in order to compare against previous work in SC2 player modeling by Avontuur [3]. However, due to the imbalance in the league distribution, this measure will not be as meaningful a measure of performance as the F_1-score. The final evaluation was performed on the holdout test set. We report both the weighted F_1-score, individual class F_1-scores, and weighted accuracy for comparison to Avontuur's work.

4.3 Feature Selection

A total of 11 features were extracted from the replay files and used as input to the model. The features are as follows:

Actions Per Minute In RTS games, players give instructions to units by clicking the mouse or typing on the keyboard. The computation of (average) Actions Per Minute (APM) is

$$\text{APM} := \frac{a}{t}$$

where a is the total number of actions performed over t minutes. A higher APM of a player means that the player has the ability to process more events in each unit of time in the game. Having a high APM does not necessarily mean that a player is an advanced player, but high-level players usually have a higher APM. The average APM for GrandMaster is approximately 300 (5 actions per second), Bronze league players can only achieve 60 APM (1 action per second).

Spending Quotient While APM is more an indication of micro management and fine-grained control over units, macro-management, such as economic management and strategy, is also important. Players collect resources and spend resources on a number of different things. To maximise the use of resources and to build a more efficient economy, players minimise unspent resources and increase their resource collection rate by expanding and training more workers. Spending Quotient (SQ) [1] is a feature to quantitatively measure players' economic management in a game and is calculated as

$$\text{SQ}(i, u) = 35 \times (0.00137i - \log u) + 240$$

where i represents resource collection rate and u represents unspent resources. Resources can be calculated as a weighted arithmetic mean of Mineral and Vespene, where the weight of Mineral and Vespene ranges from 1:2 to 1:3. Like

[3] League accuracies are averaged together weighted by the number of examples of that league in the dataset.

the measurement of APM, players who have higher SQ should be considered as more advanced players. Figure 2 shows the average SQ of global players in each league of ranked matches in SC2. Although the difference between average SQ among leagues is not as large as that of APM, players in higher leagues have higher SQ.

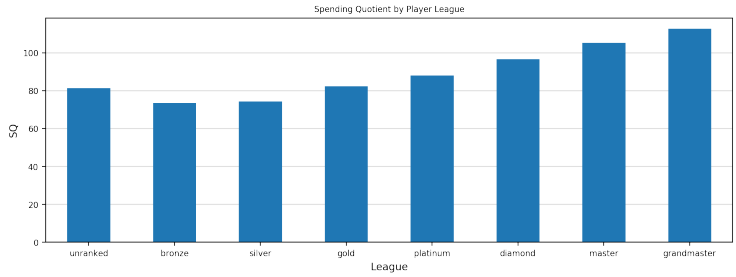

Fig. 2. Average spending quotient varies by league, with higher leagues having improved spending quotient scores.

Other features

- *Game Length* The length of the game in seconds.
- *Avg Unspent Minerals/Vespene* How many resources the player left unspent. Higher values tends to indicate weaker players, as strong players are more able to spend resources quickly.
- *Avg Mineral/Vespene Collection Rate* The rate at which the player collects resources.
- *Avg Workers* The average number of workers over the game. Higher level players tend to have more workers.
- *Avg Supply Used/Made* This indicates the number and strength of the units recruited. A higher number might indicate that the player had to replace many lost units, while a lower number could indicate that the player was not able to establish a good resource collection rate.
- *Supply Blocked Time* Units require supply to be built. If no supply is available, unit production is halted. Longer supply blocked times usually indicates a weaker player.
- *Produced Units* The number of units recruited by the player.
- *Killed Units* The number of units killed by the player.
- *Killed Workers* The number of opponent worker units killed by the player.

4.4 Model Selection

We considered five models for classifying the replays: K-nearest Nearest Neighbour (k-NN) using Euclidean distance, Linear Support Vector Machine (Linear

SVM), Non-linear Support Vector Machine (Non-linear SVM), Random Forest, and a Gradient Boosting Classifier. Each algorithm was tuned via a grid search over hyperparameters as detailed in Table 2. The best model was selected by performing 5-fold cross validation on the training set, according to the weighted average F_1-score. Feature importance for the Gradient Boosting Classifier was assessed using Gini importance [10].

Table 2. Hyperparameters for each algorithm where selected via a grid search using 5-fold cross validation, with the model with the best F_1-Score being selected.

Algorithm	Hyperparam.	Values tested
KNN	k	[1..30]
SVM-Linear	c	2^i for $i \in [-5..8]$
SVM-Nonlinear	c	2^i for $i \in [-5..8]$
SVM-Nonlinear	gamma	['auto', 'scale']
Random Forest	estimators	[10, 100, 1000]
Random Forest	max features	[Auto, 1, 2, 3]
Random Forest	max depth	[0..9]
Gradient Boosting	estimators	[10, 100, 1000]
Gradient Boosting	max depth	[3, 5, 10]
Gradient Boosting	learning rate	[0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1]

5 Results

In this section we evaluate the performance of the classifiers, as well as the importance of each feature used as input to the models.

5.1 Classifier Performance

The kNN and Linear-SVM classifiers struggled to classify the player's league, with F_1-scores of 0.516 and 0.495 respectively. The Non-linear SVM model performed better with an F_1-score of 0.560. However, the best performing models were the Random Forest Classifier and Gradient Boosting Classifier. Both models had accuracy above 60%, with the Gradient Boosting Classifier performing slightly better with an F_1-score of 0.593 compared to 0.579. The full results are shown in Table 3. The confusion matrix in Figure 3 shows that Gold players are often predicted as Platinum, whereas Master players are misclassified as Diamond. As some players sit on the boundary between leagues, in terms of their skill, it is not surprising that these leagues are confused and that under uncertainty the classifier opts for the more populated Diamond league. If we allow classification predictions to be correct if neighbouring the proper league, then the classifier's weighted accuracy rises to 94.5%, as shown in Table 4. This means that, although the classifier is sometimes wrong about the league, it is very unusual for it to predict a completely inappropriate league.

Fig. 3. The confusion matrix for the Gradient Boosting Classifier. Results are normalised. A large portion of the errors are in confusing diamonds ranked players with the league above or below it.

Table 3. Holdout test scores for each classifier. The Gradient Boosting Classifier model outperformed all other models, with Random Forrest being slightly behind.

Algorithm	Accuracy	F_1-Score	Hyperparameters
k-NN	0.538	0.516	k=10
Linear SVM	0.562	0.495	c=64
Non-linear SVM	0.597	0.560	c=64
Random Forest	0.615	0.579	max_depth=auto
Gradient Boosting	**0.617**	**0.593**	learning_rate=0.1, max_depth=5, n_estimators=100

Table 4. Classification accuracy for the Gradient Boosting Classifier for leagues within given distance. Accuracy improves dramatically when including the neighbouring league. This indicates that when the classifier miss-classifies a player, it does so by a small margin.

Distance	Accuracy
Exact	61.7%
1 league	94.5%
2 leagues	99.6%
3 leagues	99.9%

5.2 Feature Importance

It is not surprising that the most useful feature for predicting a player's skill is their APM. This is commonly recognised as an important distinction between lower tier and upper tier players. All features selected contribute meaningfully to the classifier's prediction. For a breakdown of each feature's importance, see Figure 4. The addition of the average SQ feature proved to be more important than any of the other features, except for APM. This is likely the reason for the classifier's improved performance over the previous state-of-the-art, which did not include the SQ feature [3].

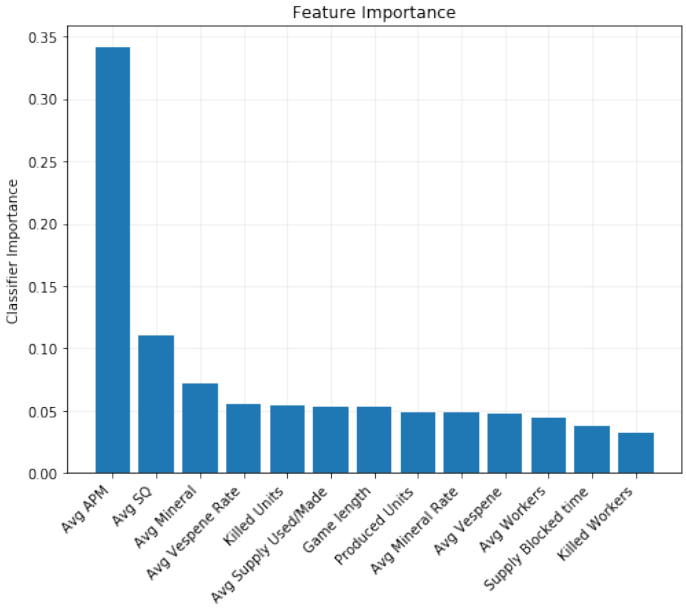

Fig. 4. Importance of each feature in the Gradient Boosting classifier. APM, a measure of micromanagement, is by far the most important feature. However, SQ, a macromanagement measure of economic performance is the strongest of the remaining features.

6 Discussion

Our results show the importance of including macro-level features, such as Spending Quotient, when accurately predicting player skill level in StarCraft II. The Gradient Boosting Model shows that while predicting the exact league of a human player is difficult, predicting to within one league is possible to very high accuracy (94.5%). This would be useful to game developers as it would

be possible to identify players that are well out of their correct league placement (e.g., a GrandMaster on a smurfing account). We found that, as expected, APM is a powerful indicator of a player's ability, far exceeding all other features. We also discovered that Spending Quotient, a measure of the player's economic management, provides a lot of additional information about the player's ability and should be included, along with the other features, when predicting a player's league. Identifying Master league players proved to be very difficult for the model, which is likely due to the small number of master players and relatively little difference in ability between Master and Diamond level players. GrandMaster players, on the other hand, are identified very accurately by the model.

7 Conclusion

Our Gradient Boosting model demonstrates that identifying player skill level in StarCraft II from replay data is possible and that the addition of the Spending Quotient feature provides important information in predicting the player's skill. We were able to predict the players' correct league to an accuracy of 61.7%, improving greatly on the previous state-of-the-art of 47.3% [3]. When identifying a player's skill to within one league, an accuracy of 94.5% is achievable. Identifying player skill allows developers to more easily identify smurfing and boosting behaviour and improve customisation of the game to the player's skill level.

7.1 Future Work

Due to the large number of different ways players can play StarCraft II, our classifier would benefit from training additional high-quality matches. Producing a model specifically for each of Blizzard's regional servers would also be beneficial as there are notable differences in play styles between these regions. It may be, for example, that a Diamond league player on the Korean server would be ranked Master on another server. Therefore, dividing the dataset into regions could improve the performance.

References

1. Liquipedia. https://liquipedia.net/starcraft2/Spending_quotient, last accessed: 4th August 2020
2. Ranked ftw. https://www.rankedftw.com/stats/population/1v1, last accessed: 4th August 2020
3. Avontuur, T., Spronck, P., Van Zaanen, M.: Player skill modeling in starcraft ii. In: Ninth Artificial Intelligence and Interactive Digital Entertainment Conference (2013)
4. Buro, M.: Call for ai research in rts games. In: Proceedings of the AAAI-04 Workshop on Challenges in Game AI. pp. 139–142. AAAI press (2004)
5. Chen, Z., Sun, Y., El-Nasr, M.S., Nguyen, T.H.D.: Player skill decomposition in multiplayer online battle arenas. arXiv preprint arXiv:1702.06253 (2017)

6. Hippe, R., Dornheim, J., Zeitvogel, S., Laubenheimer, A.: Evaluation of machine learning algorithms for smurf detection. CERC2017 p. 65 (2017)
7. Hunicke, R.: The case for dynamic difficulty adjustment in games. In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology. pp. 429–433 (2005)
8. Kou, Y., Li, Y., Gui, X., Suzuki-Gill, E.: Playing with streakiness in online games: How players perceive and react to winning and losing streaks in league of legends. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. pp. 1–14 (2018)
9. Lee, D., Tang, H., Zhang, J.O., Xu, H., Darrell, T., Abbeel, P.: Modular architecture for starcraft ii with deep reinforcement learning. In: Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference (2018)
10. Menze, B.H., Kelm, B.M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W., Hamprecht, F.A.: A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. BMC bioinformatics **10**(1), 213 (2009)
11. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
12. Palencia, D.O.V., Osman, M.: Psyrts: a web platform for experiments in human decision-making in rts environments. In: 2019 IEEE Conference on Games (CoG). pp. 1–4. IEEE (2019)
13. Pang, Z.J., Liu, R.Z., Meng, Z.Y., Zhang, Y., Yu, Y., Lu, T.: On reinforcement learning for full-length game of starcraft. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4691–4698 (2019)
14. Ravari, Y.N., Bakkes, S., Spronck, P.: Starcraft winner prediction. In: Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference (2016)
15. Sarkar, A., Cooper, S.: Inferring and comparing game difficulty curves using player-vs-level match data. In: 2019 IEEE Conference on Games (CoG). pp. 1–4. IEEE (2019)
16. Shao, K., Zhu, Y., Zhao, D.: Starcraft micromanagement with reinforcement learning and curriculum transfer learning. IEEE Transactions on Emerging Topics in Computational Intelligence **3**(1), 73–84 (2018)
17. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815 (2017)
18. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017)
19. Sun, P., Sun, X., Han, L., Xiong, J., Wang, Q., Li, B., Zheng, Y., Liu, J., Liu, Y., Liu, H., et al.: Tstarbots: Defeating the cheating level builtin ai in starcraft ii in the full game. arXiv preprint arXiv:1809.07193 (2018)
20. Sweetser, P., Wyeth, P.: Gameflow: a model for evaluating player enjoyment in games. Computers in Entertainment (CIE) **3**(3), 3–3 (2005)
21. Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A.S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al.: Starcraft ii: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782 (2017)
22. Yannakakis, G., Spronck, P., Loiacono, D., Andre, E.: Player modeling. In: Artificial and computational intelligence in games, pp. 45–59. Dagstuhl Publishing (2013)