Early Prediction of a Game Outcome in StarCraft 2

David Leblanc and Sushil Louis
Department of Computer Science and Engineering
University of Nevada Reno
Reno, NV 89557

Abstract

The goal of this paper is to predict the outcome of a StarCraft 2 game based on information contained in a replay. We explore the problem of early game outcome prediction and the reasons for the determination. Understanding these factors may not only improves player performance, but also help game designers balance the game. Features are extracted from replays of previously played games, and are used to train a system that determines a winner. Each feature represents a snapshot of game at a specific time and encodes information about the state of the game at a particular time. Results show that it is very challenging to predict a winner early on, but it is possible to find tipping points in a game. The tipping points determine which player is ahead in particular segments of a game and give additional insight on the actions that produced the outcome.

1 Introduction

Starcraft 2 is a real-strategy game or RTS, which allows players to choose from three types of races: Terran, Protoss, and Zerg. Each race has different characteristics, strengths, and weaknesses that make the game highly dynamic and non-linear. The objective of the game is to gather resources, build structures, train armies, and attempt to destroy the opponent. Each race have different structures and unit types which have varying purposes. For example, the Protoss favor stronger, slower, and more expensive units while Zergs produces faster, weaker, and large quantities of armies. Terrans are a balance between Zergs and Protoss.

Generally, in tournaments, players face one another in a "one versus one" (1v1) format. Over the years, professional tournaments have attracted large crowds of players and spectators. What started the excitement for Starcraft 2 was the successful franchise Starcraft Brood War. Now, some of the best players in the world have become professional and attend tournaments regularly. The reasons for the success of this game are mainly because it is well designed and balanced, and provides a very high skill complexity limit for players. Players are continually trying to find flaws in their play to improve their overall skill.

Figure 1: Starcraft 2. Two players in a battle in a 1v1, Terran versus Protoss match

Because of the complexity of the game, the reasons why a player wins a game is often misunderstood by spectators and casual players. In many cases, the outcome of a game can be predicted earlier than what most players realize. If it is possible to determine the winner of a game early on, then there are many potential applications that may benefit from this knowledge. Game balance designers are responsible for assuring that the game remains challenging and highly dependent on a player's skill. Understanding the flow of a game, and the reasons for the outcome of a game could be valuable information for game balance designers. For a game to be interesting and challenging, there should be many possible ways to reach an outcome. Early prediction of a winner also helps players improve their overall play. By knowing *when* the outcome of a game was predicted, the player could then look at that specific time and determine *what* at that time influenced the outcome. From that, a player can then adjust their play to learn from mistakes, and qualify good play.

1.1 The problem

The main goal of this paper is ultimately to predict the winner of a game as early and accurately as possible, and understand what causes a player to win or lose a game. Identifying tipping points in games would help grow the overall understanding of the game. Games can vary widely from one to another, but they generally have tipping points where one player either takes to lead or falls behind. These tipping points can give an indication of what a specific player did right, or did wrong. Tipping points can range from subtle to obvious. Obvious tipping points are easily identified and explained by most players. On the other hand, subtle tipping points can be very difficult to identify by an average casual player. They often happen earlier in a game and usually play a role in the final outcome of the game.

A secondary goal is to identify these minor and major tipping points, and characterize them to understand the factors that contributed to the outcome of a game. To characterize a tipping point, a probability measure of outcome must be calculated for any point in a game. The probability can be measured by training a system to predict the outcome of a game based on a snapshot of the game at some time in the game. The snapshots are a set of features that have been extracted from the game (such as structure count, unit count, actions, etc) and represent the overall state of the game at that point.

1.2 Previous work

There has been much previous work done on Starcraft related application. Most of the focus in RTS game research is on the study of artificial intelligence to design opponents. An example of this is [8] where the focus of the work tried to design an agent able to respond to unforeseen events and developed a reactive goal-driver autonomous agent.

Another StarCraft related work, [6] explores the concept of map creation and design. They use evolutionary search to generate maps which maximize the entertainment factor based on player feedback. The application of this work is mainly to assist map designers in the map creation process.

In an interesting approach, [5] used a Bayesian model to predict the opening strategy for a player. The predictions are done specifically for the opening of the game and are learned using replays labeled with a specific opening strategy type.

For specifically predicting the outcome of games, there has been some work in the area of professional sports. [7] present a method to predict the outcome of football games based on fuzzy rules, neural networks and genetic programming. They use features from the football teams themselves such as ranks and scores. In a similar work, [4] used time-series history with Multi-Layer Perceptron learning to predict the winner of a tennis match.

Much of the work presented has been done on the first StarCraft franchise, but little work has been done on the latest edition of StarCraft. The main reason is because StarCraft 2 is a relatively recent game (since 2010). The work presented in this paper is an initial exploration of the game and its strategies.

1.3 Approach

Replays of previously played professional tournaments are collected and features are extracted which are then used to trained a system that predicts the outcomes of games. The system learn what produces a given outcome, and provides a qualitative measure of which player is ahead or behind. In section 2, the overall methodology for answering these questions is presented. Section 2.1 describes the data and feature extraction further. Features used are unit counts, buildings, player actions, and other. This information is then represented as a feature vector of histograms for any time t in a game. Each feature vector then represents a snapshot of the game at a specific time. Section 2.2 explains in detail how the features are extracted and represented.

Based on the features, the system is trained to determine the outcome and to evaluate the probability of outcome. This is discussed in depth in section 2.4.

Section 3 presents the results based on the methodology and the experiments conducted which are described in 3.1. Examples of game outcomes are shown in this section and a discussion of the results is proposed in 3.2.

Finally, section 4 summarizes the paper and offers solutions for further work.

2 Problem Data and Methodology

Starcraft 2 replays are gathered from professional tournaments which have been played in the last two years. Replays were taken from replay packs of tournaments such as MLG (Major League Gaming), IEM (Intel Extreme Master's), Dreamhack, and other sources. The replays provided by those sources are a good source of data because the level of play is at the highest level of competition. Professional players have a deep understanding of the game and are much more consistent in their level of play then casual players. Therefore, the data collected can be assumed to be

slightly less noisy and optimized for outcome prediction.

2.1 Data Set

The data set collected from the tournaments includes over 9000 replays, which contain all possible match-ups. Since there are three races, there are six possible match-ups:

- Zerg versus Terran (ZvT)
- Zerg versus Protoss (ZvP)
- Terran vs Protoss (TvP)
- Zerg versus Zerg (ZvZ)
- Protoss versus Protoss (PvP)
- Terran versus Terran (TvT)

For this paper, we focused out method mainly on TvT match-ups. There are a few main reasons why we focused on TvT. First of all, Terran build mechanics are the simplest of all races. When a unit begins construction, it is completed a fixed amount of time later. Generally speaking, buildings can only produce units one at a time. All these facts ensure that features are more easily and accurately extracted from replays for terrans. Finally, since TvT is a mirror match-up, features extracted are the same for both players and can be directly compared side-by-side.

There is a total of 853 TvT replays in the data set constructed. Figure 2 shows the game time distribution of TvT games in the data set. Most games range 10 to 20 minutes but certain games can be as short as 4 minutes, and as long as 70 minutes. It also includes a large variety of different strategies. In the past two years, the overall game strategies have evolved as players optimized the way to play the game. There have also been many balance changes in the game itself, which has impacted the strategy of the game.

2.2 Feature Extraction

Replay files contain information of the game played and the list of events performed by each player. The information can be exported to a text file using the SC2Gears software [1]. The output text file format for an event is as follows:

Frame Player EventType EventDetails

Frame: The timestamps associated with the event. It can be converted into seconds.

Player: The name of the player performing the action.

EventType: The type of event or action the player performed (ie: Train, or Build, Research, etc...)

EventDetails: The details associated with that type of actions, such as the unit type, target location, assignment, etc...

The *Frame* is converted to seconds to have a more meaningful representation of time. *Player* determines which player performed the action, allowing the events to be separated for each individual player. The *EventType* determines the action taken by the player. For the purpose of this application, the replay files are parsed and the events are split into five event category types:

1. Build Event: Player builds a structure (ie: Barracks, Factory, etc...)
2. Train Event: Player trains a new unit (ie: Marine, SCV, etc...)
3. Research Event: Player researches an upgrade (ie: Stim, Combat Shield, etc...)
4. Ability Event: Player uses an ability (ie: Cloak, Call Down MULE, etc...)
5. General Actions: Contains all mouse-click, hotkey, control groups, camera movements, and other events.

Finally, the *EventDetails* contains extra specific information associated with the event, which is currently ignored for the most part. The sequences of events parsed from the file and kept separate for each players.

Once parsed, the events are sorted by time and organized in a table, called a build order table (BOT). BOTs are built for each player by sequentially inserting build, train, and research events, and keeping track of time to complete the events. When an event is processed, it is first added to the table, the production count for that event is incremented, then the BOT is updated. The update process goes down the table, looking at production and time, and calculates the amount of time passed since production started. If the amount of time passed is greater or equal than the time required to build that specific unit, then the unit count is incremented for that type, meaning a unit has completed production. The update process is done for each unit type at each event. After all events have been registered and processed, the BOT for each player contain the unit, building count, and research progress for each unit type, at any point in the game.

Features can then be extracted from the BOTs at any time t in a game. These features will essentially be a snapshot of the unit count at that time t.

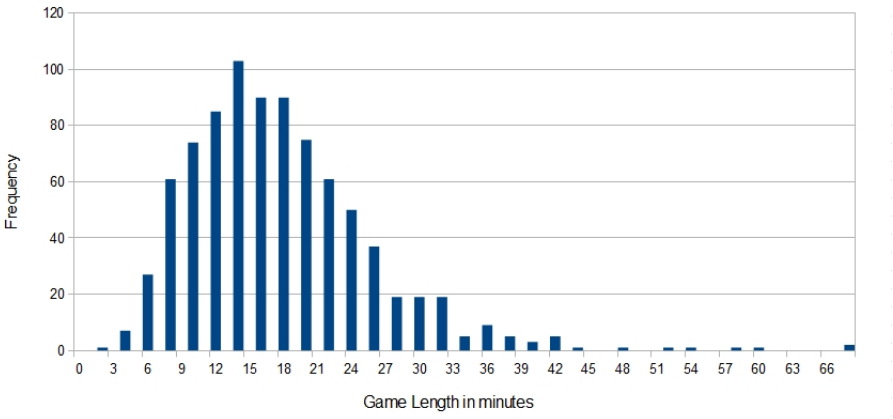


Figure 2: Time distribution of Terran versus Terran games in the constructed data set.

The features also contain the count of ability events, and general player actions per minute (APM) measure. APM is the number of actions performed by the player over time. The APM measure is split into two main types: micro and macro. Micro APM counts the abilities and mouse-click events, while macro events consist of building, training, and researching events.

Finally, the winner of a game can be identified by determining which player leaves the game first. A win is encoded with a 1, and a loss is encoded with a 0.

2.3 Feature Representation

Once all the features have been extracted from the replay, they are represented as a vector of attributes. Each vector is a snapshot of the game at time t for a player. For a single Terran player, 75 different attributes can be extracted from the events. For Protoss and Zerg, the unit, building, and research count are different, therefore would result in a vector of different length. The distribution of the attributes is displayed in table 1.

The vectors encode all the available information available for a single player at time t. The are three main options for representing features extracted:

1. Single Player
2. Both Players
3. Difference of Players.

The first option accounts only for features from a single player and results in a vector of 75 attributes (for Terran). The second option concatenates the vectors of the first option for both players, which results in a vector of length 150. Because we are dealing with a mirror match-up (TvT), attributes are the same for both players, and can be compared side-by-side. Therefore, both players can be represented as a single vector of 75 attributes using either the difference between each attribute.

For any feature extracted at time t, two vectors can be produced. For option 1, two single single player vectors are recovered at each t. For option 2, the two vectors can either be {player1, player2} or the inverse {player2, player1}. Similarly, for the Difference between the players, it can either be {player1 - player2} or {player2 - player1}. Beyond the representation, it can be further split into two types of snapshots used for classification: Basic and Temporal.

Basic Features

Basic features are simply the vectors described above taken at specific snapshots at time t of the game. The length of the vectors for a single player is 75, as mentioned above.

Temporal Features

Temporal features contain the basic features, the mean and variance of the previous snapshots. A window of time is taken from time t and k snapshots are extracted from that window. The mean and variance of each attributes over time are calculated and added to the feature vector. This essentially adds two features per attribute, resulting in a feature vector with a length of 225 (75+75+75) for Terrans. This representation increases the amount of information encoded about change over time in the attributes.

2.4 Feature Evaluation

The outcome of the game is determined by which ever player is the first to concede. An important assumption is made which is that the outcome of the

Unit Histogram	Building Histogram	Research Histogram	APM	Total	**Outcome**
13 unit types	17 building types	37 upgrade types	8 types	75	**Win (1), Lose (0)**

Table 1: Feature vector representation and distribution of attributes. APM 8 attributes: Abilities, Actions, Build, Train, Research, Micro, Macro, and Overall APM.

game determines the outcome for all features extracted during the game. This is a strong assumption because it could be incorrect. Because of the nature of Starcraft 2, a player could be at a disadvantage at some time t, but overcome his opponent later in the game. Players can make brilliant decisions that lead to victory later on, or make disastrous mistakes which lead to defeat. There is no guarantee that the outcome of a given feature vector accurately represents that feature. Despite this fact, our assumption is that the trained classifier can learn from these scenarios, and learn whether a given snapshot is likely to result in a win or a loss.

2.5 Outcome Prediction

The features and their associated outcomes are used to trained a system which predicts the outcome of a give snapshot. Because the outcomes are either a win, or a loss, the prediction can be done using a classifier. A multitude of classifiers have been used to predict the outcomes. We used the WEKA toolkit [3] to evaluate the prediction accuracy of multiple different systems. The results are shown in section 3.

As mentioned in 2.3, since there are two players, two sets of features can be extracted at any time t. Because one player must win, and the other lose, this should hold true for the prediction as well. This constraint can be used to further improve the overall accuracy of the system. By comparing the outcome predicted by one player versus the outcome of the other, we can determine a confidence measure for the prediction.

For determining the confidence measure, we used Gradient Boosted Trees [2]. This classifier can do both classification and regression which allows the predicted outcome to have a confidence associated with it.

3 Results

3.1 Experimental Setup

Feature snapshots are extracted from all the replays at 30 second increments. The first three minutes of all games is ignored since generally very little happens in the first three minutes of a game. Of the 853 TvT replays, 70% of features extracted are used for training, and the remaining 30% is reserved for testing. The overall accuracy of predictions of various classifiers (using WEKA) are presented in table 2.

Since the goal of the project is to determine when the outcome of the game can be accurately determined. Table 3 presents the results of the outcome prediction. The columns represent the percentage of correct prediction of snapshot beyond a time t. The numbers inside the table represent at which percentage of the game the prediction accuracy of the snapshots was reached. The last row is the number of games that were successfully predicted with the given certainty.

3.2 Discussion

From the results presented in table 2 and table 3, We can clearly see that StarCraft 2 outcome prediction is a challenging problem. In table 3, we can see that for 86.0% of the games in the test set, we could on average predict over 50% of the snapshots after the first 8.9% of a game. Another interesting result from the table is that the earliest we were able to predict 100% of snapshots was after 66.7% of a particular game. In order words, the winning player dominated the last third of the game. On average, we are only able to predict with 100% accuracy the final 4.9% of a game. This is a clear indication that in many cases it is difficult to accurately predict the winner of a game early on due to the variations in strategies.

Although the accuracy appears low for the classification, the outcome predicted in many cases is correct. It appears that in many cases, there are moments where winning player falls behind. Example prediction graphs are show in Figure 3. This Figure shows how dynamic games can be and there are multiple stages of transition in most games.

4 Conclusions and Future Work

Prediction of outcome of a Starcraft 2 game is very challenging, as it should be. The replay data set used was taken from professional level tournaments, therefore the quality of players and strategies are optimized, resulting in closer games and less flawed overall performance. At that level of play, one would expect accurate prediction of outcome a difficult task.

The information extracted from the replay data

Classifiers	Basic Features			Temporal Features		
	Single	Both	Diff	Single	Both	Diff
Random Forest	51.9%	58.1%	57.6%	54.0%	58.7%	61.5%
AdaBoost (Stumps)	52.1%	54.2%	59.0%	53.9%	56.4%	62.2%
AdaBoost (Random Trees)	50.6%	54.0%	56.4%	52.7%	56.1%	55.5%
Naive Bayes	51.4%	53.1%	59.9%	52.6%	55.7%	**63.6%**

Table 2: Results of overall outcome prediction of different classifiers in WEKA, and different feature representation method

Time Percentage	50%	60%	70%	80%	90%	100%
Mean	8.9%	13.7%	20.0%	26.4%	34.5%	95.1%
Median	0%	0%	0%	16.0%	33.6%	96.0%
Stdev	20.3%	24.1%	26.8%	29.7%	30.4%	3.6%
Earliest	0%	0%	0%	0%	0%	66.7%
Latest	97.1%	97.1%	97.1%	97.1%	97.1%	99.0%
Games Predicted	**86.0%**	**85.3%**	**82.2%**	**79.8%**	**79.1%**	**78.3%**

Table 3: Prediction Time table.

set is incomplete, lacking units lost and income information. It is also impossible to flawlessly extract the accurate count of units and buildings due to many factors which are handled by the game engine itself. Despite this lack, valuable results have been extracted from this data, proving that the game is very well balanced and designed.

Future work for this project would be to improve feature representation to maximize classification accuracy while minimizing over-fitting. Also, this paper focused mainly on TvT, other match-ups could be considered, and similar methods applied to them with success. The main short-coming of the method was the data extraction from replays. This short-coming could be avoided by extracting features from a live game, in real-time, and possibly extract the missing information such as units lost, income, and other pertinent statistical values.

References

[1] A. Belicza, SC2Gears (https://sites.google.com/site/sc2gears/), 2010-2012

[2] J. H. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, *Technical Discussion: Foundation of TreeNet(tm)*, 1999

[3] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

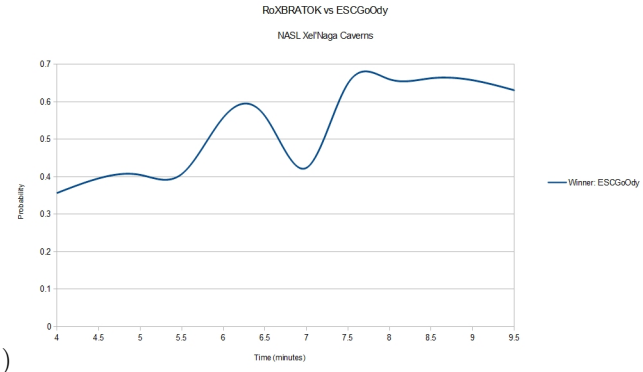
[4] A. Somboonphokkaphan, S. Phimoltares, and C. Lursinsap, Tennis Winner Prediction based on Time-Series History with Neural Modeling.

[5] G. Synnaeve and P. Bessiere, A Bayesian Model for Opening Prediction in RTS Games, with Application to StarCraft, *Computational Intelligence and Games*, 2011

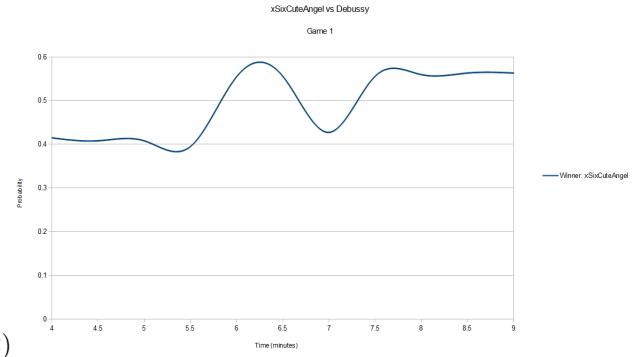
[6] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelback, G. N. Yannakakis, Multiobjective Exploration of the StarCraft Map Space, *IEEE Conference on Computational Intelligence and Games*, 2010

[7] A. Tsakonas, G. Dounias, S. Shtovba, and V. Vivdyuk, Soft Computing-Based Result Prediction of Football Games.

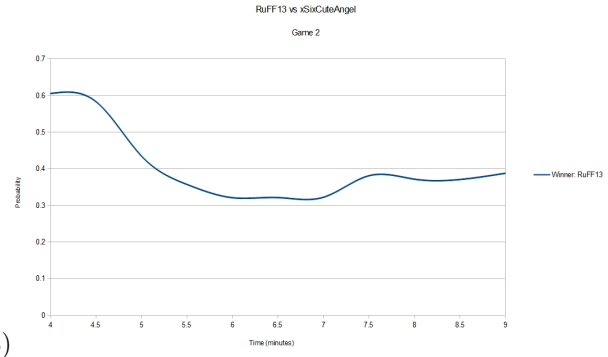
[8] B. G. Weber, M. Mateas and A. Jhala, Applying Goal-Driven Autonomy to StarCraft, *Association for the Advancement of Artificial Intelligence*, 2010



(1)



(2)



(3)

Figure 3: Example Games of prediction probability as time passes in a game. In (1), the winning player opted to go for a more defensive and technology focused play, while his opponent favored an aggressive and economic opening. The aggression was stopped and the winning player counter attacked with higher technology and wins the game. In (2), the winning player opens with a very risky and aggressive type of play. Since it is risky, his probability of winning starts out lower. His aggressive play is successful this time and does enough damage to allow him to go on and win the game. Finally, (3) presents a failure in prediction. The game is completely unorthodox; both players play aggressive and destroy each other. The prediction probability shown in (3) is incorrect for this particular game