Early Prediction of Outcome of a Starcraft 2 Game Replay

David Leblanc, Sushil Louis,

Outline Paper

Some interesting things to say here.

***Abstract*—The goal of this paper is to predict the outcome of a Starcraft 2 game based on information in a replay. We explore the problem of how early in game can the winner be determined and what are the reasons why a player loses a game. 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 they 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 ammount of units and the player's actions. Results show that it is very challenging to predict a winner early on, but it is possible to find tipping points in a game and determine which player is ahead in particular segments of a game.**

***Index Terms*—Starcraft, Game, Prediction, AWESOMENESS**

I. Introduction

STARCRAFT 2 is a highly dynamic and non-linear game. It has over the years attracted large crowds of people at professional tournaments. Now, some of the best players in the world have become professional players and make a living playing it and attending tournaments. The reasons for the success of this game are mainly because it is so well designed and balanced, and has a very high skill limit for players. Players are continually trying to find flaws in their play to improve their overall play. *MORE IN DEPTH EXPLANATION OF WHAT STARCRAFT IS HERE!!!*

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 dependant on a player's skill. If the winner of a game could easily and accurately be determined for any given game, this could be an indication of imbalance in the system.

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 determined *what* at that time influenced the outcome. From that, a player can then adjust their play to learn from mistakes, or qualify good play.

A. The problem

The main goal of this paper is ultimately to predict the winner of a game as early as possible, and understand what causes a player to win or lose a 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 by an average player. They often happen earlier in a game and may 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 comparing a snapshot of the game to snapshots of previous games played, and determining the chance of winning given a certain situation.

B. Previous work

I need to do more research here for now. I don't think anyone has done this specific application before. There may be some similar work out there, but for the moment I couldn't find much. INSERT STUFF HERE

C. Approach

Replays of previously played professional tournaments are used to train a system to learn what produces a given outcome, and how to measure which player is ahead or behind. In section II, the overall methodology for answering these questions is presented. Section II-A describes the data and sources further. Features can be extracted from the replays, such as unit counts, buildings, player actions, etc. 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 II-B 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 II-D. Based on the information learned in the classification process, many feature representations modifications were made to improve accuracy, which is explained in section II-E.

Section III presents the results based on the methodology and the experiments conducted which are described in III-A.

David Leblanc is with the Department of Computer Science and Engineering at the University of Nevada, Reno. More interesting stuff here. Reno, Nevada, United States, e-mail: lblancdavid@gmail.com.

Sushil Lousi is with the Department of Computer Science and Engineering at the University of Nevada, Reno. More interesting stuff here. Reno, Nevada, United States, e-mail: xxx@xxx.xxx.

Examples of game outcomes are shown in III-B and a discussion of the error is proposed in III-C.

Finally, section IV summerizes the paper and offers solutions for further work.

II. Problem Data and Methodology

Replays are gathered from professional tournaments which have been played in the last two years. Replays were taken from replay packs from tournaments such as MLG (Major League Gaming), IEM (Intel Extreme Master's), Dreamhack, and other sources *I NEED SOME CITATIONS HERE PROBABLY*.

A. Data Set

The data set includes over 9000 replays, which are of all possible matchups. For this paper, we focused out method mainly on terran versus terran (TvT) matchups. There are two main reasons why we focused on TvT:

1) Terran build mechanics are the simplest of all races. When a unit begins construction, it is completed a fixed ammount of time later. Buildings can only produce units one at a time.
2) Features are more easily extracted from replays for terrans.
3) Since it is a mirror matchup, features 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 1 shows the game time distribution of TvT games. Most games fall in the 10 to 20 minute range but games can be as short as 4 minutes, and as long as 70 minutes.

MORE INTERESTING STUFF TO ADD HERE!!!

B. Feature Extraction

Replay files contain the list of events performed by each player. The replay file can be exported to a text file using SC2Gears software (REFERENCE HERE!!). The output text file format for an event is as follows:

<FrameNumber> <PlayerName> <EventType> <EventDetails>

FrameNumber: The timestamp associated with the event. Can be converted to seconds.

PlayerName: 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 *FrameNumber* is converted to seconds to have a more meaningful representation of time. The *PlayerName* determines which player performed the action, allowing the events to be seperated 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 building (ex: Barracks, Factory, etc...)
2) Train Event: Player trains a new unit (ex: Marine, SCV, etc...)
3) Research Event: Player researches an upgrade (ex: Stim, Combat Shield, etc...)
4) Ability Event: Player uses an ability (ex: 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 we ignore for the most part. The sequences of events parsed and kept seperated for both players.

Onces all events have been parsed from the replay files, the events are sorted and organized in a table, called a build order table (BOT). BOTs are built for each player by inserting sequencially 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 wheter enough time has passed from the time unit was started. If the amount of time is equal or greater than the required build time for that unit, then the unit count can be incremented, meaning a unit has completed. 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. *A FIGURE HERE MIGHT BE HELPFUL???*

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*. 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 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.\

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.

C. 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. The vector has 75 attributes, and an output value which represents whether or not the player ends up winning the game. The distribution of the attributes is displayed in table I.

The vectors encode all the available information for a single player at time *t*. Since the goal of this application is to determine the winner of a game, both players must be representing in the feature vector. The are three methods used to represent both players in one vector:

1) Ratio between attributes.
2) Difference between attributes.
3) Concatenation of two feature vectors, resulting in a 150 attribute vector.

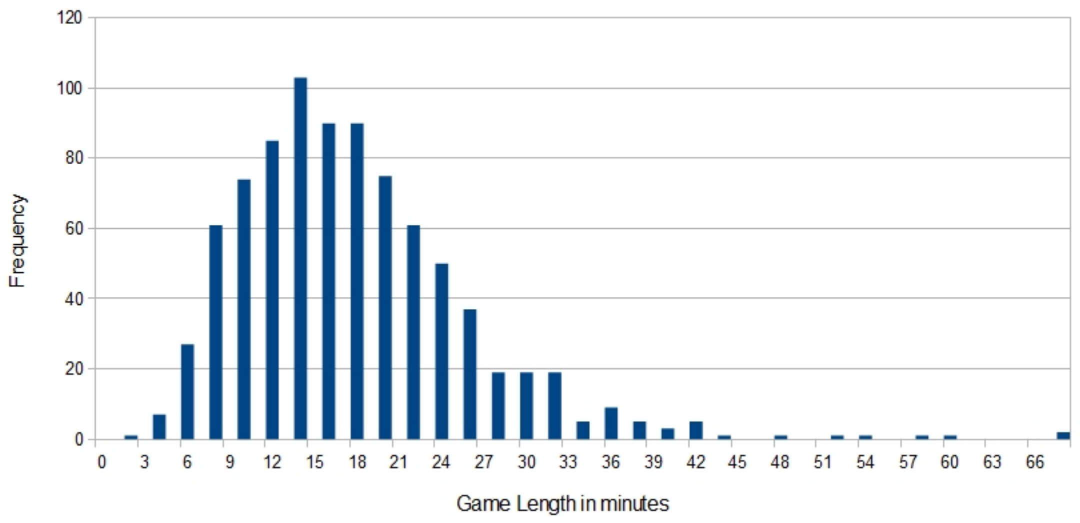

Fig. 1. Time distribution of TvT games in the data set.

TABLE I
FEATURE VECTOR REPRESENTATION AND DISTRIBUTION OF ATTRIBUTES. APM 8 ATTRIBUTES: ABILITIES, ACTIONS, BUILD, TRAIN, RESEARCH, MICRO, MACRO, AND OVERALL APM.

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

Because we are dealing with a mirror match-up (TvT), attributes can be compared side-by-side for both players, so representation options 1, and 2 are acceptable. This would not be the case for all non-mirror match-ups. All three methods are tried and experimented with for this application.

The ratio method is best representated as a percentage count of units. For example, if the first player has 2 marines, and the second has 1 marine, the first player would then have 66.6% of all marines, and the second 33.3%. For the difference representation, the resulting value for the first player would be 1, and -1 for the second player.

Once both players have been represented in the feature vector, there are two main ways to further represent the data. *I SHOULD INCLUDE SOME MORE FIGURES HERE THAT WOULD BETTER EXPLAIN THE FEATURE REPRESENTATION.*

1) Spatial Features: Spatial features are simply the basic feature vectors described above taken at specific snapshots at time t of the game.

2) Temporal Features: Temporal Features add information from past snapshots of the game. 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). This representation increases the amount of information encoded about change over time in the attributes.

D. Feature Evaluation

The features should be evaluated to determine which attributes contribute the most to the outcome of the game. The outcome associated with each feature vector represents whether a player wins or loses at the end of the game. There is no guarantee that the outcome of a given feature vector accurately represents that feature. A player could be ahead at one point in the game, but end up losing the game, or vice versa. Despite this fact, can train a classifier to learn whether a given snapshot is likely to result in a win or a loss.

1) Classifiers: A multitude of classifiers have been used to classify feature vectors. The system is trained using WEKA (*REFENCE HERE PLEASE!!!*) and the results are shown in section III.

2) Clustering: Because the outcome associated with a feature may not be reliable for earlier times in a game, clustering the data set can give some understanding on the distribution of features in the space. The outcome can be calculated for each feature based on frequency of the outcomes within the clusters. By calculating the frequency, the outcome can then be measured as a likelyhood of winning or losing a game. This could be a solution to the problem of innaccurate outcome representation.

E. Modification and Improvement

COOL STUFF I CAN ADD HERE FOR THE FINAL REPORT.

III. RESULTS

My results are awesome *ADD SOME MORE COLORFUL COMMENTS AND INFORMATION HERE!!!*

A. Experimental Setup

PARAMETERS FOR THE CLASSIFIERS AND CLUSTERING METHODS SHOULD BE DEFINED HERE. FOR THE MOMENT, I USED DEFAULT PARAMETERS IN WEKA, AND AM STILL WORKING ON EXPERIMENTING WITH CLUSTERING METHODS.

Feature snapshots are extracted from all the replays at 30 second increments. A few experiments are done on the data to determine whether or not it is possible to determine the winner of a game or not. In the first experiment, features are extracted and represented as explained in section II, and trained and tested on various parts of a game. Figure 2 shows various results of prediction accuracy at different times in replays. *I'M NOT GONNA EXPLAIN THAT FIGURE BECAUSE IT IS BIAS AND WILL BE REPLACED IN THE FINAL REPORT.*

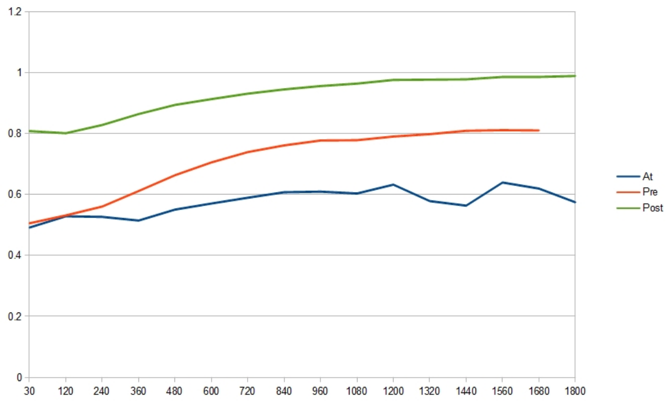

Fig. 2. This graph is absolutely horrible and will not make it to the final report. But it does show some interesting information about accuracy within the features themselves. I will most likely generate a different graph which will be better labeled and actually show true results. Since I already showed this in class, and it is really bias, and will not be in the final report, I won't bother explaining what it means.

In a second experiment, the features extracted from the TvT data are split into three main groups:

1) Early Game: All snapshots which are in the first third of the games
2) Mid Game: All snapshots which are in the second third of the games
3) Late Game: All snapshots which are in the last third of the games

In table II, the results shown are the accuracy of outcome prediction by training on certain parts of the game, and testing on other parts of the game.

TABLE II
RESULTS OF PREDICTION ACCURACY FOR CROSS TESTING AND TRAINING USING DIFFERENT PERIODS OF A GAME

Training / Test	**Early Game**	**Mid Game**	**Late Game**
Early Game	1	0.587	0.541
Mid Game	0.543	1	0.683
Late Game	0.519	0.621	1

The third experiment split the data into by using 70% of the games for training, and reserving 30% of the games for testing. The overall accuracy of predictions of various classifiers (using WEKA) are presented in table III.

TABLE III
RESULTS OF OVERALL OUTCOME PREDICTION OF DIFFERENT CLASSIFIERS

Classifier	**Prediction Accuracy**
Random Forest	63.4%
Boosted Random Tree	64.5%
Classifier A	80.0%
ClassifierB	90.0%
Classifier C	100.0%
Classifier D	174.3%

The fourth experiment was done using clusters of features. The clusters describes states of the game and the probabilities of each outcome are encoded in the clusters. *THIS IS WHAT I AM CURRENTLY WORKING ON. I WOULD LIKE TO STUDY HOW GAMES PROGRESS OVER TIME AND FIND TIPPING POINTS. I STILL HAVE SOME WORK TO DO ON THAT, BUT HERE IS WHERE THE RESULTS OF THOSE EXPERIMENTS WILL BE REPORTED.*

B. Examples

I currently do not explicitly have examples of results I got. I this section, I intend to have a discussion on what features got correctly or incorrectly classified.

I would also like to show results of my clustering method. The clustering method provides a way to get a probability of winning or losing a game. That probability changes over time. I would like to have some sort of graph here that shows that change over time, and shows some of the so-called tipping points in a game. A cool idea would also be to have a video of a game being played with a probability displayed over the player.

C. Error Discussion

Based on the results obtained, it is clear that outcome prediction of a Starcraft 2 game is a challenging problem. Because it is unclear how to measure whether or not a player is ahead at any given snapshot, many of the misclassified snapshot could be in fact correctly classified. This problem is somewhat addressed when using clustering method for predicting probability of outcomes.

MORE DISCUSSION HERE DEPENDING ON MORE FORMAL RESULTS OF CLASSIFICATION AND CLUSTERING METHODS.

IV. 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 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 engin 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 overfitting. Also, this paper focused mainly on TvT, other match-ups could be considered, and similar methods applied to them with success. The main short-comming of the method was the data extraction from replays. This short-comming 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 pertinant statistical values.

GO MORE IN DEPTH FOR THE FINAL REPORT. ALSO, ADD IN REFERENCES!!!!!!!!!

[1]

References

[1] Y. Okada, K. Dejima, and T. Ohishi, "Analysis and comparison of PM synchronous motor and induction motor type magnetic bearings," *IEEE Trans. Ind. Appl.*, vol. 31, pp. 1047–1053, Sep./Oct. 1995.

David Leblanc All about me and the what my interests are.

MY FACE HERE!

Sushil Louis Same again for the co-author, but without photo