Predicting arbitrary events in competitive computer team games

Artem Vasenin

21/10/2016

Supervisor Yingzhen Li (yl494)

Director of Studies Lawrence Paulson (lp15)

1 Introduction

1.1 Background

Multi-player computer games are becoming very popular and accurate estimation of player skill and role in the game is required for multiple purposes, such as matching players with their equals for enjoyable game-play or predicting the outcome of a match between professional teams for tournament admissions and betting purposes. Current techniques only consider player's overall skill in calculation, but frequently games requires a certain number of roles to be fulfilled and a team which fills all the required roles will be more likely to beat another team in which players are all good at the same role even if the players in the former team are less skilled on average.

1.2 Summary of work

This project will aim to implement an algorithm which will identify how likely a certain event is to occur in a game based on the team composition and previous games of each player. A collection of those events can later be used to define what roles a game have and calculate player's skill in each.

2 Starting point

I am not aware of any similar work that has been done, therefore my starting point consists of current literature such as "TrueSkill"[1]. I do not have much experience with any machine learning libraries, therefore I will learn need how to use one. I am quite experienced in Python, in which the project is will be coded, therefore I will not need to learn any new languages.

3 Description

3.1 Algorithm Description

The algorithm should be able to provide a probability distribution for continuous variables and class probabilities for discrete variables, e.g. A player can buy a certain number of items, each item can be considered a class and the algorithm should give probabilities for each. On the other hand, amount gold earned by the player in a game is better represented as a continuous variable.

A lot of variables in the game are closely dependent on each other, I would like to use neural networks to calculate the amount of dependency. This should allow the algorithm to make more accurate predictions. The algorithm should consider relationships between all available variables in the game and using that give a better estimation of outcome than just looking at one variable alone.

3.2 Possible structure of the algorithm

Estimate potential distribution of each variable by considering them independently using a sample from the data set. Construct a neural network where inputs are calculated values of each player in the game and the output is the probability distribution for particular variables. Train the network, then use back-propagation to adjust distributions of each variable.

[1] https://www.microsoft.com/en-us/research/wp-content/uploads/2007/01/NIPS2006_0688.pdf

3.3 Test Platform

The game called "Dota 2"[2] will be used for testing and evaluation of the algorithm. In this game two teams of five players each compete. Each player controls a few units (only one most of the time), the main one called "Hero". Each team has a number of buildings at the start of the game, which are under the control of the game AI. The objective of the game is to destroy a particular building belonging to the enemy team, while at the same time protecting your own. There are other units in the game which players can kill, which grants gold and experience to the player. With gold the player can purchase items for their hero and by gaining experience player's hero increases in strength.

Matches in this game are independent of each other, i.e. actions taken in one match do not change the starting point of another match. This is quite important as we would like to evaluate each match individually. I believe this game is appropriate because it is mostly skill based, there is almost no luck involved. It also requires at least two (usually five) different roles to be filled in by each team's players, which should be identified by the algorithm. It also currently heavily values player cooperation rather than solitary play, therefore good player composition is important and should be revealed by the algorithm. It is also one of the most popular and competitive games today which should provide plenty of data.

The game provides an API through which it is possible to request the summary of each game, which is returned as a set of key-value pairs (examples in section 8.1), therefore minimal preparation of data is required. Professional teams usually play more than a hundred games a year, which should be sufficient amount for calibration. Professional games are publicly available, which removes the need for permission and makes data collection easy. I will manually collect match-IDs using official tournament websites and then download the data through the API. During collection of data, I will separate outliers, which I will consider to be values further than two standard deviations from the mean. I will only start discarding outliers after 20 data points. After the data set is collected algorithm should not need to interact with online data and will run offline.

[2] http://dota2.com and https://en.wikipedia.org/wiki/Dota_2

3.4 Possible problems

Sometimes games are not zero-sum and therefore teams are likely to not play normally, which is bound to produce outliers, those games need to be identified and potentially removed from the data set.

3.5 Possible extensions

For the core project, I will only be predicting a few variables in one game if the core project succeeds I will try predicting other/all variables in the same game or applying the algorithms to other games.

4 Criterion of success

For discrete variables I will be use logarithmic loss for evaluation. For continuous variables I will use mean squared error metric for evaluation. I will consider the project a success if distributions produced by the algorithm will have MSE of less than one half of the standard deviation (standard deviation of all player's results in the data set). I only aim for one half because while player skill areas can give a good estimate of variation in performance there are other variables such as personal relationships and players' psychological state which cannot usually be inferred from the data alone.

This value will be calculated by separating the data into three parts: training, validation and evaluation set. Potential algorithms will be trained on the training set, they will then be compared using validation set and the best one will be used for prediction in evaluation set. During training and validation, maximum likelihood estimation will be used for comparison.

5 Further envisaged evaluation metrics

Logarithmic loss, MSE, mean actual error and maximum likelihood can all be used for further evaluation. I plan to also evaluate convergence rate of the algorithm and the variety of events that can be predicted with MSE less than one half of standard deviation. I will also evaluate how efficient the

code is, in respect to the result it gives (I will analyse both complexity and real-world performance). I will also try to compare the performance of this algorithm compared to others which are currently used in the industry, but that may not be possible since a lot of them are not open.

6 Plan of work

The twenty eight weeks work period will be separated into fourteen two-week sections, with the following goals:

1. 21 October - 3 November

   Researching what exactly are currently used methods/techniques in the industry and researching what kind of prediction algorithms are available. Evaluating available libraries (TensorFlow, Theano, scikit-learn, Pylearn2, Pyevolve). Libraries will be evaluated on their ease of use (amount of documentation and other support material), performance and range of available tools.

   Deliverables: A list of prediction techniques and algorithms which should be tried.

2. 4 - 17 November

   Collecting test data, preparing/cleaning it. Trying currently available inference approaches (for example predicting variables independently by fitting distribution to the data set), which were identified during the research. Start learning how to use chosen library.

   Deliverables: Data set which will be used for training and evaluation. A choice of a library which will be used for the project. List of approaches which can be extended/adapted.

3. 18 November - 1 December

   Starting work on the main algorithm. Thinking of possible approaches and working out the theory. Writing code to test each approach.

   Deliverables: Pseudo-code for each possible approach.

4. 2 - 15 December

   Identifying which approaches were successful and Implementing. Comparing and improving them.

Deliverables: First working code for each approach and result statistic for each of them.

- 16 - 29 December

  Christmas break

5. 30 December - 12 January 2017

   Correcting bugs in code. Further improvement in the code.

6. 13 - 26 January

   Writing progress report. Ideally, success criterion achieved by this point. Working on potential improvements.

   Deliverables: First draft of the progress report. First version of working algorithm.

7. 27 January - 9 February

   Progress Report Deadline 3rd of February

   Working on extending the algorithm to work with a wider range of variables within the game or extending it to other games. Alternatively, if the accuracy of the algorithm can be improved, I plan to also work on that.

8. 10 - 23 February

   Further improvements.

9. 24 February - 9 March

   Optimising the code and correcting bugs. Learning how exactly to formally evaluate machine learning algorithms.

   Deliverables: Final version of the algorithm. Script that will be used for the evaluation of the algorithm.

10. 10 - 23 March

    Aggregating completed work, summarising results and preparing an outline for the dissertation. Beginning the writing of the dissertation. Evaluating the final algorithm.

    Deliverables: Evaluation results for the final algorithm

11. 24 March - 6 April

    Writing the dissertation.

12. 7 - 21 April1

    Cleaning up the dissertation, making it presentable. Making sure it's clear and checking for errors. Complete first draft of the dissertation and ask the supervisor to check it.

    Deliverables: First draft of the dissertation.

13. 22 April - 4 May

    Work with supervisor on correcting errors and improving the text.

    Deliverables: Finalised content of the dissertation.

14. 5 - 19 May

    By this time the content should be finalised and cosmetic/presentation improvement should be made. Submit the dissertation before the deadline.

    Deliverables: Submission of the dissertation

7 Resources declaration

I am planning to use machine learning libraries, such as Theano or TensorFlow, in this project. I plan to use Python in my project. I will also need a data set for training and evaluation which I will obtain during the second work section.

I plan to use my own desktop computer and laptop, specifications of the former are:

Processor Intel Core i5-4690K @ 3.50GHz

RAM 16GB

Graphics Card GTX 980ti

OS Dual-boot: Windows 10 / Ubuntu Gnome 16.04

My contingency plans against data loss are to use git for version control on GitHub. The code will be daily copied to Google Drive and also copied weekly to external hard drive.

My contingency plans against hardware/software failure are that I can use my laptop in case my desktop fails or MCS computers.

I warrant that I accept full responsibility for any hardware or software failure.

8 Appendix

8.1 List of possible variables

Win/Lose (Discrete)

Whether the player won the match.

Hero (Discrete)

Five heroes which were picked by the team or particular hero which was played by the player. There are currently 112 heroes available, 104 of which were picked in the last major tournament.

Items (Discrete)

There are about 140 items in the game, but the player can only use six at any one time (in most cases).

Match length (Continuous)

Duration of the match.

Final level & XPM (Continuous)

A hero starts with level one, the level increases with experience up to 25. XPM = experience/minute, since matches vary in duration, this can be a better indicator of player skill.

Total gold & GPM (Continuous)

Gold is awarded whenever a player kills enemy units or destroys enemy buildings. GPM = gold/minute, similar to XPM.

Kills (Continuous)

Kills are awarded whenever a player kills an enemy hero.

Deaths (Continuous)

Deaths are increased whenever player hero dies.

Assists (Continuous)

Assists are awarded if a player helped their ally kill an enemy hero.

KDA (Continuous)

KDA = (Kills+Assists)/Deaths.

Also "Last hits", "Denies", "Hero damage", "Tower Damage", "Hero heal" (all continuous). There are many other variables which might be considered during the possible extension.