Predicting events in computer games using machine learning

Artem Vasenin

April 17, 2017

Proforma

Name Artem Vasenin

College Clare College

Title Predicting arbitrary events in competitive computer team games

Examination

Year 2017

Word Count

Project Originator Artem Vasenin (av429)

Project Supervisor Yingzhen Li (yl494)

Original Project Aim

The aim of this project was to develop an algorithm which could predict whether certain events would happen in multiplayer computer games. In most modern multiplayer games, players are provided with a statistic after each match describing their performance in that match and summarising key choices they have made. The project should be able to predict the values of such statistics. The mean squared error of such predictions should be less than half of variance of the variable.

Summary of Work Completed

A graphical model was created which produces a probability distribution for continuous statistics.

Special Difficulties

Declaration of Originality

I, Artem Vasenin of Clare College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed

Date: April 17, 2017

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Multiplayer computer games are becoming very popular, more than a 100 million people play League of Legends every month [1]. A game's success often rests on how enjoyable it is. Current method of improving player experience is to make sure that all players have equal chance of winning in a game. For that purpose their skill has to be tracked and teams have to be arranged such that they are of equal strength.

In many popular games several roles have to be filled on each team for optimal performance. Current algorithms, such as TrueSkill [2], only track player's overall skill and do not consider what roles the player prefers and how good they are at each one. This often leads teams being composed of players all wanting to play in the same role, which either leads to team underperforming or some players not enjoying the game as much as they could. Moreover, I believe that the events that happen in game are more important to many player's experience than the actual outcome.

To be able to match players better a system has to be built that will take into account how the game is played and what strategies exist in it. Creating such a system using classical techniques would require a deep knowledge of workings of a game. Unfortunately, I do not have such knowledge of most of the games and obtaining such knowledge although might be fun, will take too long. Furthmore, most such games change their rules every few months to keep players interested, therefore performance of hard coded system would decrease as the time goes on. As a result I decided to use machine learning techniques which would help me create a system which can adapt to different games by itself and also update its judgement as game changes.

1.2 Related Work

Before starting the project, I have looked into papers and articles about predicting events in sports and computer games. I have found out that nothing similar has been tried before. Most closely related algorithms (such as [2] and [3]) were made to only predict the outcome of the game, not any related statistic. Outside of academia, other algorithms were made which used machine learning to improve their performance, but again only focusing on the outcome of the game.

Chapter 2

Preparation

2.1 Requirement Analysis

2.2 Starting Point

At the beginning of the project I had:

- Basic knowledge of artificial intelligence from *Artificial Intelligence I* Part IB course.
- Programming experience in Python (acquired by working on personal projects) and Java (acquired by completing courses in first two years of my study).

During the course of the project I had to gain following qualities:

- Understanding of various machine learning techniques.
- Familiarity with a chosen ML library.
- Understanding of theory behind ranking algorithms.

2.3 Theoretical Background

Since I decided to use machine learning in my project I had to get some background in the area. Unfortunately, computer science course is in Lent term, therefore I attended a similar course in Engineering department in Michaelmas term, before beginning work on the theory.

2.4 Software Engineering

2.4.1 Workflow

I have decided to use an agile approach of working on this project, since I was not sure exactly how the algorithm should work at the start of the project. The project would be done in two phases:

1. Primary research would be done during the second half of Michaelmas term. A prototype would be developed during Christmas break.
2. At the beginning of Lent term the prototype would be evaluated. In the first two weeks of February the theory would be improved using insights gained during the development of the prototype. In the second two weeks of February the prototype would be refactored to

comply with changes in the theory. At the beginning of March the revised implementation would be evaluated.

This approach allowed me to incorporate the knowledge I gained during the evaluation of a prototype into the final version of the algorithm.

2.4.2 Version Control

Git was used for version control. This allowed me to roll-back to a previous version of the project in case a mistake was made. It also allowed me to compare different versions of a specific component. The repository was hosted on GitHub which allowed me to work on the project from multiple machines.

2.4.3 Backup

The code was continuously synchronised to Dropbox[1]. Weekly checkpoints were saved to an external hard drive.

2.5 Tools used

2.5.1 Programming Language and ML Libraries

Machine learning is one of the key elements of this project, therefore I needed a library that would implement key techniques used in ML for me, so that I don't have to write them myself. I have compared a number of libraries (such as Tensorflow[2] and Torch[3]), I was primarily interested in how much funcitonality they provide, their performance and how easy it is to learn them.

I have compared their functionality by completing standard machine learning task, such as creating a neaural network to classify images in the MNIST[4] dataset. Performance was compared by timing how long it would take to achieve 99% accuracy, in most cases it came down whether GPU acceleration was supported. Since I would have to learn how to use the chosen library in detail I also paid attention to the amount of support material available. I took note of quality of documentation and also compared number of questions and answers on StackOverflow.

In the end I have arrived at the conclusion that they all provided required functionality and were sufficiently easy to use. API for most of them were written in different languages. I did not want to learn a new language in addition to learning a library, therefore I decided to use TensorFlow, API for which was written in Python, a language I am most comfortable writing code in. To a lesser extent Python was also chosen since it has a package manager[5], which makes it much easier to install additional packages.

2.5.2 IDE

Pycharm was chosen the integrated development environment (IDE) for the project. Pycharm has all of the core features of an IDE, such as syntax checking, autocompletion, running of code, etc. The use of an IDE streamlined the process of development and allowed me to focus

[1] dropbox.com/
[2] tensorflow.org/
[3] torch.ch/
[4] yann.lecun.com/exdb/mnist/
[5] pypi.python.org/pypi/pip/

on improving the algorithm rather than worrying about language syntax or library functions signatures.

Chapter 3

Implementation

Chapter 4

Evaluation

Chapter 5

Conclusions

Bibliography

[1] Forbes. *Riot Games Reveals 'League of Legends' Has 100 Million Monthly Players*. 2016. URL: forbes.com/sites/insertcoin/2016/09/13/riot-games-reveals-league-of-legends-has-100-million-monthly-players/#274b74155aa8 (visited on 04/16/2017).

[2] Ralf Herbrich, Tom Minka, and Thore Graepel. "TrueSkill(TM): A Bayesian Skill Rating System". In: MIT Press, Jan. 2007, pp. 569–576. URL: https://www.microsoft.com/en-us/research/publication/trueskilltm-a-bayesian-skill-rating-system/.

[3] Ruby C. Weng and Chih-Jen Lin. "A Bayesian Approximation Method for Online Ranking". In: *J. Mach. Learn. Res.* 12 (Feb. 2011), pp. 267–300. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.1953057.