CS486 Project

Jason Sun

August 10, 2015

Abstract

In this paper, I look at building a character recommendation engine for the computer game Dota 2. I explore the a machine learning algorithms called sum-product networks to solve the problem.

I discuss the modelling details for the machine learning algorithms I use. I also provide insight into how future data collection and feature selection can be done.

There are two main benefits to do this analysis. First, it is a survey of machine learning algorithms applied to Dota 2 – in the process gaining insights into the algorithms. Second, it can be a useful tool for the Dota 2 community.

1 Introduction

Dota 2 is a popular online real time strategy computer game made by Valve. It is based off Dota, which was a custom map for the Warcraft 3 game by Blizzard.

There are many similar games (that can be called competitors), such as League of Legends by Riot, and Heroes of the Storm by Blizzard, The recent Dota 2 international competition *The International 5* had a prize pool totalling $18.4 million US dollars[1]. Dota 2 has about 600 000 unique players per day[2] as of July 2015.

This seems fits as a machine learning recommendation problem. But it is challenging due to a large volume of the solution space. With 110 heroes to choose and five heroes per team, this results in over eight quadrillion possible team combinations.

Recommending heroes using machine learning is challenging because it tries to capture via raw data what professional players have developed a gut instinct for through hundreds of thousands of hours of game time.

1.1 Game

A game match consists of 10 players, each selecting a different **hero** (character to play). At the beginning of every match, this selection must be made within

[1] `https://www.dota2.com/international/compendium/`
[2] `http://steamcharts.com/app/570`

the first 60 seconds[3].

1.2 Characters in the game

Heroes come from a pool of different heroes, currently 110 out of 112 can be played. Once a player chooses a hero, no other player can select that hero for the same match. Heroes have a wide-range of characteristics and abilities that, combined with the massive hero pool, make each match unique. Heroes each have their strengths and vulnerabilities. Generalizing to the real-world, an archer is good from attacking afar, but not good in close range. So a counter for an archer is to pick a character that can quickly close the distance.

1.3 Character selection

In choosing a hero, players must keep in mind the individual strengths and weaknesses of each hero, but also the strengths and weaknesses of heroes already chosen by other players.

An effective hero pick is one that synergies with the heroes chosen by teammates, and both exploits the weaknesses and minimizes the strengths of the heroes chosen by the opposing team.

Well devised hero choices can give a team a large advantage before the match even begins. The goal of the project is to provide this recommendation by applying machine learning to learn patterns from previous matches.

1.4 Recommendation goal

The recommendation engine must answer the following question:

Recommend a few heroes that are often seen to be picked along side the current selection of heroes.

The current selection of heroes can be chosen by the enemy or by your team. The recommendation engine provides a suggestion in a fashion similar to completing a word of length 10, by suggesting some alphabet characters commonly seen together with the provided input.

1.5 Sum-product network

I intend to use a sum-product network (SPN) model for this recommendation engine. SPNs outperform other state-of-the-art deep architectures, according to the SPN paper[4].

Deep architecture to solve this problem seemed appropriate because recent pattern recognition technologies such as voice recognition and image object recognition have made significant improvements one the deep architectures were integrated.

[3] Before one gets penalized and loses 2 **gold** (in game currency) per second

[4] `http://turing.cs.washington.edu/papers/uai11-poon.pdf`

2 Related Work

2.1 Dotabuff

Dotabuff[5] is a popular website for analyzing game matches. Its analysis methods are proprietary, but it seems to uses Valve's web API for recent games data mining, and possibly even download the game replays (which are large files about 20-40 Mb), because some of the match data is not available from the API alone. This website allows the user to visualize statistics in an organized manner. The website currently makes no effort to provide hero recommendations, but can be helpful in verifying collected data.

2.2 Other recommender papers

There are some papers I found that explored generally the same recommendation problem, using different models.

1. Paper that modelled this problem using the K-nearest neighbour model[6]. They utilize a K-nearest neighbours approach with custom weight and distance functions.

2. Another paper used logistic regression approach[7]. They studied how the win rate depends on hero selection by performing logistic regression with models that incorporate interactions between heroes. Their models did not match the naive model without interactions which had a 62% win prediction rate, suggesting cleaner data or better models are needed.

3. Finally a paper that used the random forest approach [8]. An attempt was made to create a model using machine learning that can predict the winning team of a Dota 2 game given partial data collected as the game progressed. A couple of different classifiers were tested, out of these Random Forest was chosen to be studied more in depth.

2.3 Sum-product network papers

The sum-product networks paper[9] and their image completion problems[10] served as references. In particular, the image completion problems contained the complete set of results and the SPN code.

[5] https://www.dotabuff.com

[6] http://cs229.stanford.edu/proj2013/PerryConley-HowDoesHeSawMeARecommendationEngineForPickingHeroesInDota2.pdf

[7] http://cs229.stanford.edu/proj2014/Atish%20Agarwala,%20Michael%20Pearce,%20Learning%20Dota%202%20Team%20Com

[8] http://www.bth.se/fou/cuppsats.Nsf/all/0e9be67e75c84276c1257e04004d222d/$file/BTH2015Johansson.pdf

[9] http://turing.cs.washington.edu/papers/uai11-poon.pdf

[10] http://alchemy.cs.washington.edu/spn

3 Dataset

I acquired through the Valve's API call[11] for 5000 most recent high level matches. To use these you need a Steam API key[12].

The dataset satisfies the following requirements:

- The game mode is *ranked captain's mode*. This game mode is where players play for rank, and generally take the game more seriously. Consequently, hero choices would be factored in, more so than other match modes.
- The skill level of the players is *very-high*. Utilizing only very-high skill level matches allows learning from best players.
- No *abandoned* games. Players that leave the match before the game is completed affect the outcome of the match.

The data for each match is structured as JSON and includes which heroes were chosen for each team. The training set is 90% of the matches, and 10% for a test set.

As there are 110 heroes, each game is one instance of 48 trillion possible games $\binom{110}{10}$. With my limited number of most recent matches, it is a rough approximation of how the hero mechanics work.

4 Modelling

In this section I describe how I formulate the problem as a **sum-product network**[13] problem. The goal is to learn what heroes are commonly picked along side with heroes currently seen.

The **learning process** is as follows:

1. Create a feature vector representing a match.
2. Try to complete the match

The **testing process** is to provide a set of heroes that are commonly seen along with what heroes are currently selected.

Note this section is incomplete.

5 Experiments

My intent was to use the SPN code that was used for image completion, but I have not figured out how to formulate the recommender as a SPN problem.

[11] http://dev.dota2.com/showthread.php?t=47115
[12] `API Key: BAD3D130F4D06DB91EB99C1009E4A8BD`
[13] `http://turing.cs.washington.edu/papers/uai11-poon.pdf`

6 Conclusion

In this paper I try to build a hero recommendation engine by providing a set of heroes that are commonly seen along with heroes currently selected. I have not achieved my goal.

7 Future Work

- Perhaps different modelling methods could be used, such as bag of words.
- The recommendation engine can be hosted on a website, so that the user can get suggestions by entering what heroes are currently seen, choosing the desired hero from a list of auto-complete suggestions.