Machine Learning Engineer Nanodegree

Capstone Proposal

Parth Mishra
October 26, 2016

Proposal

Predicting Winners in League of Legends

Domain Background

https://www.youtube.com/watch?v=BGtROJeMPeE

League of Legends in an online multiple player battle arena (MOBA) game that is currently the world's most popular online game with 100 million monthly players worldwide.[1] Like the genre name suggests, League is very much a team game with RPG and RTS influences. There are 10 players total with 5 on each team whoa are all using a unique in-game avatar known as a "Champion" with various special abilities. Using these Champions, players fight against each other and attempt to take control of each other's respective base and thereby winning the game. Like most games in the MOBA genre, League is characterized by a very big learning curve due the vast complexity the game offers. Player skill often manifests itself in what is known as "mechanical ability". This refers to a player's command of their physical keyboard and mouse and how well they are able to use them to perform complex combinations of abilities with correct accuracy and timing.

However, while the traditional perspective in this domain is to view a player's skill as an extension of their mechanical skill, this is not exactly correct. League of Legends' complexity is generally understood to be more so in the macro level decision making process than in pure mechanical ability alone. The macro level decisions can refer to such things as resource management in the game, team utility provided, playing around several static objectives on the map, etc. The amount of variables that can influence macro decision making is extremely high and thus higher ranked players typically only get there after years of playing. Given this game's popularity, many players constantly seek to find ways to improve their skill and better identify weak points in their game. Knowing the optimal method of play comes mostly through either experience or through coaching via various mediums. However, using statistics available from every game of League of Legends can provide some data backed insights that could potentially influence player's ability to become better in less time compared to the aforementioned traditional methods of improvement. By looking at the individual statistics of players of various skill levels (7 ranks to be exact),

[1] http://www.forbes.com/sites/insertcoin/2016/09/13/riot-games-reveals-league-of-legends-has-100-million-monthly-players/#7581d79a10b1

it is possible to identity the differences in player skill by looking at the various metrics the game provides via its API. Not only can it be used to identify players of different skill, further analysis on the game data can reveal what particular metric or set of metrics are the best indicators of "skill".

Part of why I'm submitting a proposal for this topic is because this game is one of my favorites and I've devoted a lot of time and effort into getting better at it myself (I'm in the top 2% of player rankings) as well as teaching others how to get better at it. Noting the inefficiencies in the learning and teaching process, I've always wanted to formalize a data centric analytical approach to improvement. Riot Games, the maker of the game, does research into this area as well, as evidenced by posts like this one: http://na.leagueoflegends.com/en/page/turbo-mode-data-behind-your-spell-spamming where research was done based on quite literally how often a player presses a certain key per game across all skill levels. From this, insights into the mindset of players based on how often they use or didn't use a certain ability were extracted and backed by real statistical evidence.

Problem Statement

The main objective here is to use the various, quantifiable in-game statistics to accurately identify the rank of a player based solely on their average in game statistics. In doing so, we could then identify the biggest statistical influences of player skill. Given the multitude of various statistics available to track, the preprocessing to reduce the dimensionality of the data will be integral part of interpreting the results of the skill classifier. A good classifier that can accurately predict the rank of player would mean that the input vector would represent in-game stats that could provide quantifiable thresholds of differences in player skill.

Datasets and Inputs

Due to the niche nature of this problem, there are not very many publicly available data sets to use to tackle this problem. However, through use of the publicly useable Riot Games API[2], I plan to sample ranked game data for players of various skill rankings. The queries return JSON formatted data that can be then transformed into a standard CSV format. Queries on player game data returns numerous categorical data that needs to preprocessed in order to reduce the amount of time needed to process these data. The categorical data may not be well understood outside of people familiar with the game some documentation must also be made in order to concisely show the meaning of the various categories. The resultant dataset will feature a mixture of categorical and continuous data so some preprocessing of dummy variables may be appropriate prior to beginning any analysis. Additionally, one weakness of these data is due to the fact that it's only available for the current iteration of the game known as a "patch". The game gets modified, or "patched", every 2 weeks so it's not guaranteed that the results from the dataset used in the experiment would be necessarily indicative of results a few patches from now. My belief is that while the game does undergo minor changes every 2 weeks or so, they generally are not game mechanics changing to the point where any insights derived from this dataset would not generally hold some months later. A more thorough approach would be to sample from every patch over the course of what is known as a competitive "Season" (they last ~10 months usually and go through ~15-20 patches)

2 https://developer.riotgames.com/api/methods

to have a more generalized model that is not affected by temporary (read: patch long) oddities. Due to foreseen time and resource constraints, this doesn't appear to be feasible and I would most likely just use samples for a single patch.

Solution Statement

The solution to this problem has been mentioned briefly before but essentially, the idea here is to create an accurate modeling of player rank based on in-game statistics which logically will be made up of the most relevant statistics that are indicative of player skill. After clearly defining the feature vector, for each individual statistic, threshold values for the various player tiers could be computed and presented in such a way that one could establish quantifiable "goals" of what a player at each tier statistically looks like. As mentioned in the previous section, the solution might not be exactly replaceable were one to sample at a different time due to the ever changing nature of the game. Again, due to time limitations, it will not be possible to establish a comprehensive sampling method over time that would mitigate this issue, but I do acknowledge that it would be the optimal approach.

Benchmark Model

Like many benchmark models for classifiers, the ones that could be used would be a random classifier or an statistically-based classifier. Having a classifier that is better than just guessing at random would be a good place to start given that there do not seem to be any well established methods of classification of player ranking. A slightly less naïve and probably better approach could be to just simply use some descriptive statistics of player sample distributions for each particular rank in order to gain some ballpark estimations of classifications which would perform slightly better than random guessing. This would give a more realistic measure of improvement.

Evaluation Metrics

Using a simple accuracy evaluation metric (# of correct classifications / # of total classifications) is a basic but useful metric for determining the usefulness of a classifier. The more accurate a classifier is, the better the model is at representing the underlying statistical relationship. A robust model that could accurately determines player rank would almost certainly be comprised of only the most useful/relevant statistics for determining player skill.

Project Design

1. **Preprocess data**

There is no dataset ready for this analysis so I will begin by generating one by randomly sampling ranked game data for 200 random players that belong to one of 7 in-game ranks: Bronze, Silver, Gold, Platinum, Diamond, Master, and Challenger where Bronze is the lowest skill players and Challenger is comprised of the top players on a server. The rankings will be used as labels for the dataset. If the multivariate classification problem that would result from this is too granular, I could also create smaller labels that bring together several tiers that group somewhat similar tiers together. For example:

Bronze-Gold = Class 1
Platinum-Diamond = Class 2
Master-Challenger = Class 3

I'm using 200 player samples since the Challenger ranking is static and only composed of the top 200 players in the game at any given moment in time whereas the other rank tiers have no such restriction. Therefore the total dataset will be 1400 samples. Samples in this case refer to a unique game ID. There are several considerations about the nature of the data retrieved that must be considered prior to including it as a sample in the dataset.

A ranked game can consist of players of differing ranks e.g. a single game may have 6/10 players ranked "Bronze" and 4/10 players ranked "Silver". This tends to happen when the players involved are in the top or bottom of their respective rank's distribution. Players in these games are on the verge of being upgraded into the next tier or downgraded into the previous tier. One possible way around this would be to only include a game in the sample set for a particular rank if either all the players are of the same rank or some high threshold such as 8/10 players in the game are of the same rank. The drawback here is that actual player distributions tend to be positively skewed with the majority of players near the bottom of their rank tier so "middle-sampling" would work, but might not be optimal in the case where skill discrepancies within a tier are high. This is in part caused by the Ranking system in place which is a modified, obfuscated MMR (match making rating) system that is generally represented in a continuous fashion through such systems as the Elo system that is popular for games like Chess.

Rank Tiers have subs tiers that are numerical representations of player ranking with respect to tier e.g. Bronze 5 is lower skill than Bronze 1 but both are Bronze, Bronze 1 is one step lower than Silver 5, etc. As mentioned previously, since it is usually case that the majority of players within a division are in the <Tier> 5 division, random sampling will likely represent a skewed representation of skill in each tier. Given there are 5 divisions, it seem like it would make sense to only sample games of players in <Tier> 3 division since that would mean the players in this division are the logical mean of that particular Rank Tier. Another approach could be to actually use the player distributions to select a more realistic average division so as to better represent the majority of players in a single tier. My intuition would be that this would be Tier Division 4 or 5 since that's where the majority of players are. This might be optimal since the players in Divisions 1 and 2 are often outliers that can end up causing a positively skewed distribution.

2. Data Exploration

Once a satisfactorily sampled dataset is produced, it is imperative that we reduce the extremely dimensional data set. The exact tradeoff of features to get rid of could be inferred through several feature transformation methods and dimensionally analysis. For example, since we are interested most in the statistics that separate each tier, using something like Principal Component Analysis (PCA) could help in identifying those particular features without too much loss of information. PCA would maximize the variance which accomplishes one the aforementioned goal.

3. Training and Evaluation

In order to see how well the chosen dimensions and features selected for the samples model the data, several implementations of classifiers can be used. Since the data is labeled, this problem calls for a supervised learning algorithm in order to perform classification. No particular supervised classifier stands out as an immediately obvious solution so most likely, several classifiers would be used such as SVMs, Naive Bayes, and Logistic Regression. The classifier with the highest success could then have its parameters tuned using some various methods like Grid Search to find optimal parameters for the given classifier algorithm.

Depending on the results from some simple classifiers, it may also be worth exploring some ensemble approaches like Random Forests. Although, the multi class classification required might narrow down algorithm selection since some algorithms are clearly better suited towards binary classifications e.g. Decision Trees.

4. Results Analysis

Creating an optimal classifier is not necessarily the point of this project. The point of generating the accurate classifier is to develop a model from which we can derive the most important statistics. Ideally this would mean that once these features that comprise the model are understood, descriptive statistics of each feature for each Tier distribution can be utilized to make thresholds values for various categories and answer the question: “Statistically speaking, what does a player in each rank tier look like?”