Thomas Huang, David Kim, Gregory Leung
EECS 349 Machine Learning
Spring 2015, Northwestern University
Professor Doug Downey

Introduction

League of Legends is an online competitive multiplayer strategy game that pits a team of five players against another team of five players with each player selecting a distinct character, known as a champion, from a pool of currently 124 champions. It is currently the most played online game in the world, with over 7.5 million players concurrently during peak hours.

Our task is to use data given before a match begins, and predict which team will win. This is valuable, because it gives players a tool to analyze whether they should consider leaving (known as dodging) a match before it starts to avoid losing.

Motivation

Before a match starts, 10 players each take turns picking from the pool of available champions. This phase is known as “champ select”. During champ select, a player can see the names of their team members, and the champion picks for that game.

After champ select is finished, there is a short 30 second pause before the game begins. At any time during this, any player can choose to cancel the match should they feel that the game is a losing game.

Our goal is to try and predict using the given data whether the game will be lost or not. From the above definition of champ select, the available data for a given player are the champion picks for all 10 players, and the names of the 5 people on the player’s team, including the given player.

Data

Using the League of Legends Match API, we generated a training set and testing set, each of 300 matches. Information for each match includes the 10 players in the match, the champions which those players are using for the match, and the overall winrate for each player for the champion that they are using.

A player can only see the match histories of his teammates, rather than all 10 players, before the match begins. So we also investigated how accurately we can predict the winner of the match using only one team’s data.

We used 5 methods to analyze the datasets, looking at

1. Champion Picks (no separate training/testing data): 10 attributes for each player from a category of 124)
2. Individual Winrates on the Current Champion: 10 attributes for each player giving the winrate [0, 1.0] that a player has on the specific champion they are playing
3. Summed Winrates: Same as above, but 2 attributes with the winrate of team 1 versus team 2
4. Summed Winrates with logarithmic weighting based on number games played on champion
5. Summed Winrates for One Team Only.

J48: Decision Tree
IBk: Nearest Neighbor
MLP: Multilayer Perceptron
BN: Bayes Net
NB: Naive Bayes

Champion Picks

	ZeroR	IBk	IBK (k = 3)	BN	NB
10-fold CV	52.5%	61.8%	56.8%	63.3%	64%

Individual Winrates

	J48	IBk	MLP	BN	NB
10-fold CV	77.0%	83.6%	86.9%	81.8%	87.6%
Testing Set	85.7%	87.0%	92.0%	87.7%	91.0%

Summed Winrates

	J48	IBk	MLP	BN	NB
10-fold CV	87.3%	89.7%	89.3%	89.3%	89.7%
Testing Set	92.8%	84.1%	92.8%	91.3%	92.8%

Summed Winrates with logarithmic weighting based on # games played on champion

	J48	IBk	MLP	BN	NB
10-fold CV	75.7%	81.7%	81.7%	77.0%	80.7%

Testing Set	79.0%	79.7%	84.7%	76.3%	84.3%

Summed Winrates for one team only

	J48	IBk	MLP	BN	NB
10-fold CV	79.3%	81.0%	79.0%	79.3%	79.7%
Testing Set	89.0%	80.3%	86.3%	86.7%	87.3%

When we initially examined the champion picks, our data indicated that examining those attributes was not that useful with Naive Bayes being the highest at 64% over the 52.5% that ZeroR did. Our conclusion was that champion picks was not a very useful stat to look at.

Surprisingly, when we looked at individual win rates for each player on the champion they are playing, we found a very significant increase up to 91% on the testing set with Naive Bayes. Intuitively, winrate is a very good predictor for the outcome of a game, because people who tend to win games with a specific champion may have more positive impact on the current game.

However, that left the problem of having 10 distinct attributes for each player, when in reality it does not matter with combination or ordering the players come in. In other words there are 5 players on team 1 and 5 players on team 2, so there should ideally be 2 attributes. In order to solve this, we summed the winrates for team 1 and for team 2. There was not a very large change, resulting in 92.8% on Naive Bayes.

Using the same data above, we also tried weighting the data for an instance logarithmically based on the number of games the player has played on their selected champion. Our reasoning was that players who have more experience with their champion have more impact on the game, and thus should be weighted more in the algorithm. However, it performed much worse with about a 10% reduction in accuracy across the board from the previous dataset.

The problem with the previous models is that determining an instance for those models requires you to know the 10 champion picks for a game, and the match history of all 10 people on those champions. In “champ select” a player only sees the names of the players on their own team, so we tested summing the winrate for the players own team. The best result on the testing set was J48 decision tree at 89%. Upon investigating the tree in Weka, the decision tree was a single branch deciding whether the attribute was greater than or equal to 260.43 (summed winrates of team in percentages was greater than 250.43%).

Conclusion

It is very easy to predict who is going to win by just summing the champion-specific win rates of players on a champion in game. Looking at champion picks alone will not provide a good predictor, but may be good as additional attributes in classifiers where the extra dimensionality does not hinder it, such as decision trees. This suggests that for most games played, individual skill on a champion is the most important aspect in determining a game's outcome over aspects like team compositions and experience.

Further examinations could be looking at other data such as other game statistics of players on a champion, like gold earned in previous games or the kill/death/assist averages. However, the limitation is that this takes more API calls and significantly longer to process. This also potentially requires classifiers that can handle missing attributes, since the API is more likely to return error response instead of data.

References

Riot Games API: https://developer.riotgames.com/