Dota 2 Game Outcome Prediction

Group Information:

Group number: 6
Group name: miner_FAQ
Members: Yifei Li (Group Leader), Qi Zhang, Xinyuan Wang

Abstract

This report describes the experiment on predicting the result of a public DOTA 2 game. Each game is a 5 on 5 competition, the players have to play heroes chosen from 112 heroes. Our prediction is based on the heroes they picked. The experiment uses three algorithms, KNN, Logistic Regression and Frequent Pattern Mining. The output is the average accuracy of our predictions with respect to each algorithm.

Introduction of the Background

DOTA(Defense of the Ancients) 2, is a multiplayer online battle arena (MOBA) video game developed and published by Valve Corporation. It is a stand-alone sequel to a Warcraft 3 mod called DOTA. Each match consists of two teams(the radiant and the dire) of five players controlling “heroes”, and their aim is to destroy the opposing team’s base. There are 112 unique heroes now(patch 6.85).

It is a common sense between DOTA 2 players that the heroes chosen by each team significantly influences the outcome of each match. Usually, when players are playing the game, based on their previous experiences, they have an intuition for picking heroes. However, the intuition is too implicit and may vary from player to player. We want to apply data mining technology to figure out the explicit information about the heroes picked by two team and based on the information, we can predict the outcome of a certain match.

Our research can provide the game designers with the heroes selections and win rates. They can release new patches which could balance the game and make the game more competitive. This would definitely improve the user experience. This research could also be useful when it comes to other fields such as the football games or basketball games. Our research is very fundamental, but it does provide a way to analyze the outcomes of competitive sports.

Problem definition and formalization

As we mentioned before, there are two teams named Radiant and Dire in every single match. There are five players on each team. Each team has to pick 5 heroes from the all 112 heroes pool without duplicates.

$$match = \begin{cases} \{r1, r2, r3, r4, r5\} & \text{radiant} \\ \{d1, d2, d3, d4, d5\} & \text{dire} \end{cases}$$

We use r1,r2,...,r5 and d1,d2,...d5 to denote the hero IDs for each player in each team. The hero ID ranges from 1 to 112 which means there are 112 heroes in total. In each match, each ID should be unique. There is a built-in mechanism in DOTA 2 which could balance the players' skills for each public game. A public game means a match made by the system, you don't know who will be your teammates or opponents. However, no one could even know the probability of winning for each team before the match is done. Therefore, if we can find a way to learn the winning rate for each team as soon as the ten heroes are picked. It will be helpful to understand which team's heroes combination is better. However, the heroes combination strategies affect the match result in most of the matches, it is reasonable to predict winner and loser based on the heroes of each team.

To sum up, we filter the heroes selection information of each match from large amounts of match data. Then use our heroes selection information as input to train our model. And for every coming match, we will make outcome prediction based on selected heroes. Finally, we will compare our predictions with the actual outcomes to get our prediction accuracy. Our baseline is random guess, in other words, the basic accuracy is 50%. Therefore, we hope we would reach more than 50%, the higher the better.

Data Description and Preprocessing

We obtained the match data from DOTA 2 server. There is a DOTA 2 match history WebAPI[1]. By using this API, we can retrieve the match history and match details in JSON or XML format. In our project, we are using JSON.

First of all, since we are using hero IDs, we made a web request to get all the hero IDs and their description.

Here is a screenshot of the hero_id.json file, we can see the hero IDs and corresponding names.

```
"heroes": [
    {
        "name": "npc_dota_hero_antimage",
        "id": 1,
        "localized_name": "Anti-Mage"
    },
    {
        "name": "npc_dota_hero_axe",
        "id": 2,
        "localized_name": "Axe"
    },
    {
        "name": "npc_dota_hero_bane",
        "id": 3,
        "localized_name": "Bane"
    },
    {
        "name": "npc_dota_hero_bloodseeker",
        "id": 4,
        "localized_name": "Bloodseeker"
    },
```


Once we get the here IDs, next step is to collect the match data. There are large of data online. And in order to obtain these match data, we implemented a web crawler which can make the HTTP request, and collect the data based on the match id. Each success request would return a json file which contains the details of a certain match. Due to the server limitation, we can only make one call per second.

The match data contains the basic information about every single player in each team. The figure above shows how the data looks like for one player. The “account_id” represents the


```
"result":{
   "players":[
      {
         "account_id":111078806,
         "player_slot":0,
         "hero_id":44,
         "item_0":145,
         "item_1":50,
         "item_2":135,
         "item_3":156,
         "item_4":53,
         "item_5":61,
         "kills":15,
         "deaths":13,
         "assists":16,
         "leaver_status":0,
         "gold":2436,
         "last_hits":187,
         "denies":0,
         "gold_per_min":488,
         "xp_per_min":530,
         "gold_spent":24350,
         "hero_damage":22045,
         "tower_damage":1903,
```

player ID. “player_slot” is the position of a player. “hero_id” is the mostly useful information for our project which shows which hero one player has picked. The “item_0”, “item_1”, ..., “item_5” denotes the item ID. One player can have multiple items in one match. “Kills”, “deaths”, “assists”, …, “tower_damage” represent the contribution for a player in this match. The match data contains a list of players..

After collecting the json files, with the help of Gson[2], we can convert the json files to POJOs(Plain Old Java Object). Each json file is converted to a corresponding Java object. Then based on some filtering rules, we deleted some invalid match data. For instance, since we predict based on the public games, the ones which are not public games are what we need. And most of the matches are 5v5 matches, we delete the data without enough number of players, and so on.

Methods Description (detailed steps)

In order to solve prediction problem, there are many algorithms exist in real world. Such as Kmeans, KNN, etc. In particular, we choose Logistic regression, KNN and frequent pattern mining as our top of implementations.

- **Logistic regression**

  Logistic regression is a basic algorithm for matrix data classification problems. In this project, we are going to implement two versions of Logistic regressions.

  1. Logistic regression version 1

  Since logistic regression is used for matrix data, we need to create a matrix before any implementation. According to the Web API, there are 112 different hero IDs. Therefore, we designed a 226-element vector,

$$x_i = \begin{cases} 1 & \text{if a radiant player played as hero with id i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{112+i} = \begin{cases} 1 & \text{if a dire player played as hero with id i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{224} = \begin{cases} 1 & \text{constant column} \end{cases}$$

$$x_{225} = \begin{cases} 1 & \text{if the radiant team won} \\ 0 & \text{otherwise} \end{cases}$$

The “i+1” is due to ID starts from 1 while index starts from 0.

For instance, assume we have a dummy match data like this,

players	hero ID
radiant1	44
radiant2	36
radiant3	103
radiant4	64
radiant5	2
dire1	96
dire2	53
dire3	21
dire4	66
dire5	107

radiantWin = true

Let’s use a 1-D array M with 225 columns to denote this match(initial values are all 0).

Then for the radiant team, M[43], M[35], M[102], M[63] and M[1] should be 1, for the dire team, M[208], M[165], M[133], M[178], M[219] should be 1. And M[224](constant), M[225](outcome) should also be 1. Therefore, The vector contains 12 columns with value 1 and the other columns are 0.

Each game is converted to a vector, then we will perform logistic regression on these vectors.

2. Logistic regression version 2

In the second version, we separate each match into two parts, the radiant team formation and the dire one. So each match object is converted to two 114-element vectors.

$$x_i = \begin{cases} 1 & \text{if a player played as hero with id i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{112} = \begin{cases} 1 & \text{constant column} \end{cases}$$

$$x_{113} = \begin{cases} 1 & \text{if the team won} \\ 0 & \text{otherwise} \end{cases}$$

For instance, let us use the aforementioned dummy match data. This time, we need to separate it into two 1-D vectors, R and D which stand for radiant and dire.

For vector R, R[43], R[35], R[102], R[63], R[1], R[112](constant) and R[113](label) should be 1, meanwhile, for vector D, D[95], D[52], D[20], D[65], D[106] and D[113](constant) should be 1, and D[113](label) should be 0.

In this case, assume we have n matches, then we will have a training matrix with 2n rows since we separate each match into two part.

Then we can predict the probability of winning the game for each team in the same match. Afterwards, we declare the team with the higher winning probability as the output of the match prediction.

There are two main reasons that we developed two versions of Logistic regression. The first one is that by separating one match to two vectors, we can obtain more vectors. Although the dimensions keep the same, we want to know if this would affect the final result. Another reason is that for each match, we have a winner vector and a loser vector, we believe our model can learn more from this from this kind of separation.

We are going to combine the second version of Logistic regression with frequent pattern mining. We will talk about it later.

KNN

We are using the vectors similar to the vectors mentioned in the first version of Logistic regression. It is a 224-element vector, hero IDs from two teams.

First of all, we set the number of neighbours as the square root of the number of all the matches[3].The key problem in KNN is the definition of distance. In our case, we are using a terminology similarity to describe the distance between two matches. The higher the similarity is, the closer the distance is. At first, we were just using the following formula.
i and j denotes two matches, if at a certain index k both i_k and j_k are 1, then similarity is increased by 1.

$$d_{ij} = \Sigma_{k=1}^{224} AND(i_{k-1}, j_{k-1})$$

For instance, let's see the following table,

	0	1	2	3	4	...
m_1	1	0	1	1	0	...
m_2	0	0	1	0	0	...

This table only displays the first five indexes. Only index 2 could increase the similarity of the two matches by 1 while other indexes could not count.

Frequent Pattern Mining

Frequent patterns can reflect the synergistic and antagonistic relationships between heroes. We can use it to find good combinations of heroes. We have implemented Apriori algorithm to find out the frequent patterns in each winner of a game. We are only considering winners since we want to predict the winning probability of a team.

First of all, we have to decide the threshold, min support. Suppose we have n matches, since we are only considering the winner, there are 5n heroes in total. So the average appearances of each hero is 5n/112. However, the distribution is not even. The baseline is 5n/112, we tried several values and decided to use 5n/336, this number would get better amounts of frequent patterns.

Since each team consists of 5 players, the length of frequent patterns range from 1 to 5. We have found there are always many 2-tuples but seldom 3-tuples and almost no 4-tuples and 5-tuples. This is very reasonable because firstly, when you are picking heroes, your opponents are picking heroes too. Therefore, even there are some heroes with higher winning rates, two teams both could get some of them and break the frequent patterns. Secondly, in DOTA 2, there is also another terminology named counter-pick, it represents that when you pick a hero, your opponents would pick a specific counter hero or pick a hero who is the best match with your pick and break your

plan. And in tournaments, there is a step before each game called ban & pick. Two teams would take turns to ban some heroes and pick some heroes. Therefore, no matter in public games or in tournaments, it is quite hard to pick an ideally perfect team formation. This also proves that 3-tuples, 4-tuples and 5-tuples can not exist. In conclusion, we only take frequent patterns with one or two heroes into consideration.

As we said, we will combine frequent pattern mining with our second logistic regression. We extended the dimension of our vector by 2. For each winning team, we search through our frequent patterns list and record the number of frequent patterns we have found.

The modification of the vector is shown by the following figure,

$$x_{113} = \{ n \quad \text{number of length-1 frequent patterns}$$

$$x_{114} = \{ n \quad \text{number of length-2 frequent patterns}$$

$$x_{115} = \begin{cases} 1 & \text{if the team won} \\ 0 & \text{otherwise} \end{cases}$$

Previously, index 113 represents the win or loss of the team, but now we have extended the vector by 2 dimension, so the label is moved to index 115, index 113 and 114 now represents the number of frequent patterns in this team.

For instance, let us the dummy match again. The winning team is formed by [44, 36, 103, 64, 2] these heroes. Firstly, we search for 44, 36, 103, 64 and 2 one by one through our length-1 frequent patterns and write down the number to our extended column, then we search for all the possible combinations, such as [44, 36], [44, 103], [36, 64], [103, 2] and so forth, record the number to the other extended column. Finally, we will perform logistic regression on these extended vectors.

Experiments Design and Evaluation

Since all the public games are made by the default match making system, the players' skills are close. Therefore, it is straightforward that the win or loss is fifty-fifty. So the coarse baseline of our accuracy is 50%. We are going to perform cross-validation based on our whole dataset. Each time, use 80% of total matches to train our model and use the left 20% to test our model.The evaluation criteria is the average accuracy of our predictions. The higher, the better.

We implemented these three algorithm by ourselves, so we have to make sure our algorithms would work properly first. Therefore, we created a small simulated data set and used it to test the correctness of our algorithm right after we finished coding. After proving the correctness, we began using real data.

Logistic regression 1

Matches	5000	15000	30000	46554
Accuracy	0.518	0.546	0.567	0.557

Logistic regression 2

Matches	5000	15000	30000	46554
Accuracy(without FP)	0.598	0.597	0.585	0.578
Accuracy(with FP)	0.597	0.598	0.591	0.590

KNN

Matches	5000	15000	30000	46554
Accuracy	0.562	0.562	0.565	0.569

Conclusion

Since our baseline is 50%, all the three methods achieve accuracy higher than it. The Logistic Regression 2 with frequent patterns reaches the highest prediction accuracy, while KNN is a little lower and Logistic Regression 1 is the lowest. And if we could crawl more data, the results will be higher and more stable.

Based on the results, we can claim that Logistic Regression 2 is better than Logistic Regression 1. This proves our guess to be true. The separation does improve the learning phase. However, the frequent patterns mining does not help much. But there seems to be a tendency that with more matches, more frequent patterns, the accuracy could probably be higher.

Future Enhancement

By intuition, players' skills and the synergistic or antagonistic relationships between different heroes would influence the outcome of the match. Right now, we are only considering the relationships between heroes. In the future, we want to improve our model to integrate with players' historic performances. We may even build a hero recommendation system for players which could help them decide their picks.

Besides combining with players' skills, with regard to our current model, firstly, we want to extract more features such as heros' gold income, KDA(kills, deaths and assists) and so forth to enhance our model. Secondly, we want to design a better distance formula for KNN method. Last but not least, we can do some time series mining. We can divide each game into early time, late time or even for every 10 minutes. Then based on different time periods, we can find out more useful information which could influence the outcome.

References

1. DOTA 2 WebAPI http://dev.dota2.com/showthread.php?t=47115
2. Gson User Guide https://sites.google.com/site/gson/gson-user-guide
3. Maier, Markus, Matthias Hein, and Ulrike von Luxburg. "Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters." *Theoretical Computer Science* 410.19 (2009): 1749-1764.

Task Distribution Form

Task	People
1.Collecting and preprocessing data	Qi Zhang
2. Implementing Logistic Regression and KNN	Yifei Li
3. Implementing Frequent Pattern Mining	Xinyuan Wang
4. Evaluating and comparing algorithms	All members
5. Writing report	Yifei Li

Additional Points

We implemented all the three algorithms, Logistic Regression, KNN and Frequent Pattern Mining by ourselves, we also integrated frequent patterns with Logistic Regression. At the same time, we collected DOTA 2 match data using a crawler on our own.