RoadTo10K Final report

ECE 324 Final Project

Team member:

Ke (Coco) Zhang

Haobang (Larry) Wu

Word Count:2000

Introduction

The goal of this project is to predict the outcome of DotA2 matches. Dota2 is a popular esports game where two teams of five human players (teams are known as 'Radiant' and 'Dire') play against each other and one of the team will finally win.

The most critical factor in winning a DotA game is the players' skill and performance. However, this is hard to predict as it varies from match to match even for the same player. We are interested in how "hero" picks will influence the winning chances for each team. Before a game starts, each of ten players pick a unique character, knowns as "hero," to play for that match. There are 117 heroes in total, and once a hero is picked in a game, it cannot be picked again. Each hero has unique abilities, making some heroes synergize with each other, and certain heroes exploit certain heroes' weaknesses. The stage of picking heroes is thus essential for winning a game and fascinating to investigate.

The project is split into two parts. The first part predicts the winning team with hero picks only. With the result, a hero suggestion tool is created to suggest picking the 10th hero that can optimize the winning probability for the team, given the other nine picked heroes.

For the second part, two more early-game features – gold and experience – are added along with the hero picks to make outcome prediction. Heroes gain gold and experience in various ways during the match (e.g., killing enemy heroes) and therefore empower themselves. We take heroes' gold and experience at the 5-minute mark to use as initial measurements of human performance. We are aiming to increase our accuracy by adding the human factor.

We decided that neural network is an appropriate tool since there are many synergies and counters between heroes during the picking phase. "Gold" and "experience" also contribute to the winning chance of teams during mid-game, but the extent depends on the heroes picked on each team. For example, some heroes are extremely strong early-game but weak late-game, so even if a team with such heroes have a reasonable gold advantage at 5 minutes, it is probable for them to lose the game if they do not "close-out" on the game soon. It is interesting to see if neural nets can "detect" these relationships and make accurate predictions.

Background and related work

We have found several existing works on predicting the DotA2 match outcomes. A notable one is Andrei Apostoae's 'Dota2-predictor' [1], which is a prediction tool constructed for the same purpose with our first part. Besides the hero picks, he also uses a particular data preprocessing, which defines a new feature representing the relation (synergy and counter) between the heroes. Using logistic regression, he achieved 60% accuracy. He also reported that NNs achieved similar accuracy, although details/source code was not given. He was using data from an older version of the game while we were motivated to see if we can produce similar or better results using data from the newest version.

Project Overview

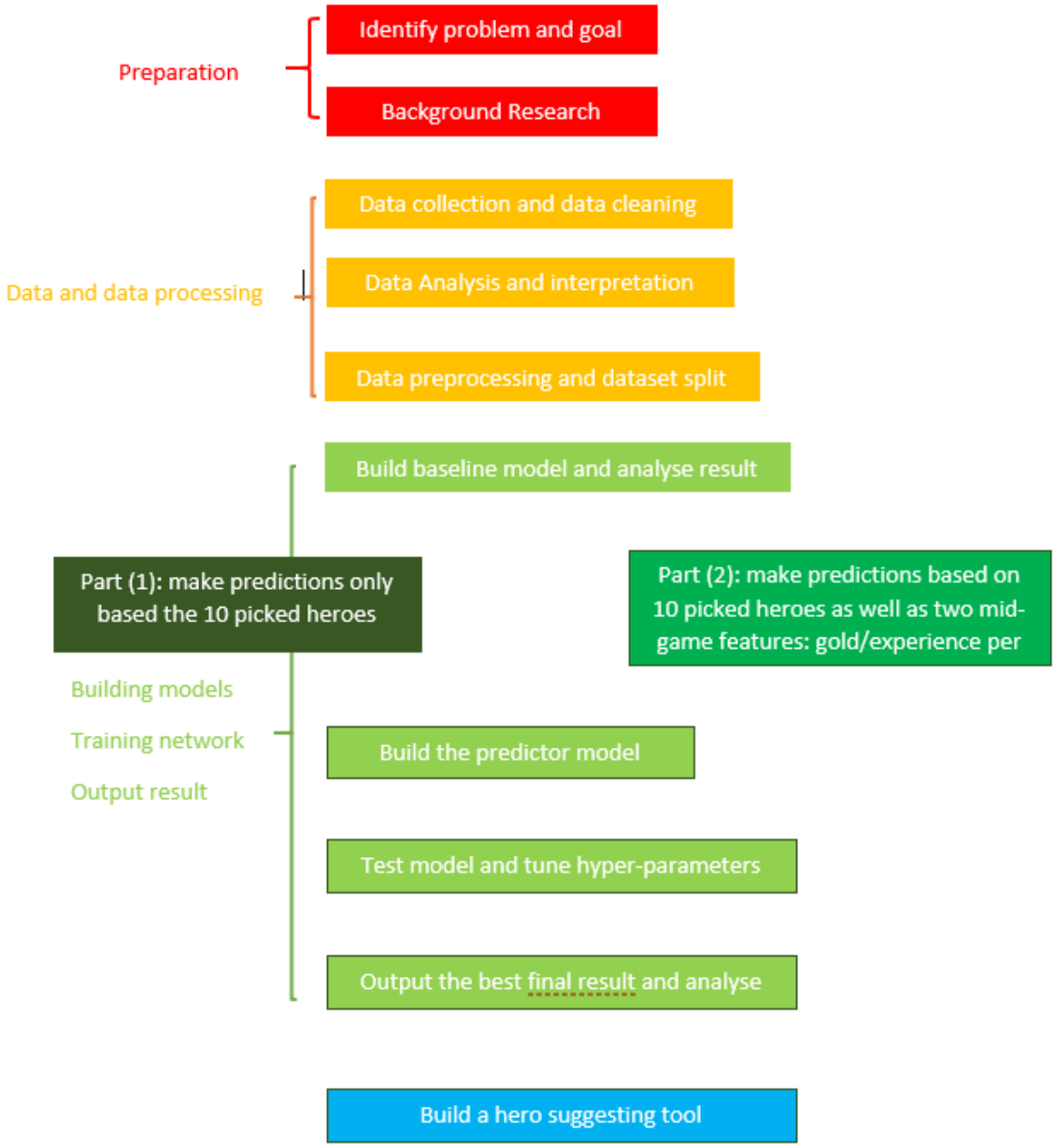

Figure 1 Project Illustration

Data and data processing

Part 1

Collection

We collected and cleaned data on 100k matches using OpenDota's API [2].

We used the "publicMatches" call method from the opendota API. Each call provides us with 100 matches' data in the form shown in Figure 1. From each call, we found the matches that meet our requirements (shown in Figure 2) and add the match data to our dataset.

We restricted our match data to "Ranked," "All Pick" matches where the average "MMR" of players is more than 4500. To explain: "MMR," or match-making rating, is a numerical rating of the player's skills. 4500 is ~5% of the players' skills. "MMR" is gained and lost during "ranked" game modes, where the players with similar "MMR" (skill) are put in the same match, and the winning team of each game gain MMR. "All Pick" is a game lobby type where all heroes are available for picking, reducing random factors in other game modes where players can only choose from limited heroes.

By setting the above constraints, we ensure that the two teams are relatively balanced and have relatively high skills. During data collection, only the data satisfying the constraints were recorded.

```
curr_id = curr_matches.loc[99].match_id
curr_matches = curr_matches[curr_matches.avg_mmr > 4500]
curr_matches = curr_matches[curr_matches.lobby_type == 7]
curr_matches = curr_matches[curr_matches.game_mode == 22]

matches = pd.concat([matches, curr_matches], sort=False)
matches.to_csv('data.csv')
```

Figure 2 Choosing data

Processing

After initial data analysis, we found some outliers in class ''Game Duration'' which represents the game playing time. We removed games with duration less than 20 minutes because these games were likely extremely one-sided due to player(s) using an account with less "MMR" than their actual skills. We also balanced the number of games based on class "Radiant_win" to have equal numbers of data for the two different game results. There are around 90k samples after processing.

	avg_mmr	avg_rank_tier	cluster	dire_team	duration	game_mode
0	4646.0	75	137	71,104,14,93,39	1276	22
1	4915.0	78	152	74,25,104,90,92	1850	22
2	6320.0	78	187	41,65,9,31,13	1975	22
3	4693.0	64	251	112,19,102,9,41	2157	22
4	4673.0	66	111	21,71,56,102,6	1935	22

	lobby_type	match_id	match_seq_num	num_mmr	num_rank_tier	\
0	7	5108240400	4286117452	6.0	6	
1	7	5108235218	4286116524	4.0	6	
2	7	5108233612	4286116186	5.0	7	
3	7	5108232113	4286116809	1.0	4	
4	7	5108231913	4286114385	6.0	10	

	radiant_team	radiant_win	start_time
0	76,12,9,84,98	True	1573431681
1	73,26,51,108,6	False	1573431079
2	93,100,74,19,102	False	1573430880
3	5,86,21,1,23	True	1573430704
4	25,90,76,28,72	False	1573430684

Figure 3 Sample Data

To use the data in our NN, we converted each team formation feature to a 1x129 one-hot vector, where value 1 represents the team picks the hero and 0 otherwise. Then we snitched the two vectors for each team together and resulted in a 1x258 vector with ten "1" s for our input. The "radiant_win" column was converted to the labels 1 (for True) and 0 (for False).We used 80% data for training, 14% for validation and 6% for test data.

Part 2

Collection

We used the dataset from Kaggle [2], and the raw data consists of 120 features. We initially planned to collect data by ourselves, but when running on our own collected matches, only ~10% of matches had in-game data specific to time. Testing results showed that the data we were able to collect are insufficient to show results.

Cleaning

For our model input, we used the gold and experience difference between the teams at the 5-minute mark in addition to the heroes picked. We calculated the value by subtracting the total gold/experience on team dire from the total gold/experience from team radiant. The hero indices are treated the same from part 1.

	match_id	r1_hero	r1_level	r1_xp	\
count	181024.000000	181024.000000	181024.000000	181024.000000	
mean	90512.500000	51.183716	1.291657	1272.037862	
std	52257.271902	33.047336	0.699610	491.334159	
min	1.000000	1.000000	0.000000	0.000000	
25%	45256.750000	21.000000	1.000000	917.000000	
50%	90512.500000	48.000000	1.000000	1171.000000	
75%	135768.250000	77.000000	2.000000	1631.000000	
max	181024.000000	112.000000	5.000000	3280.000000	

	r1_gold	r1_lh	r1_kills	r1_deaths	\
count	181024.000000	181024.000000	181024.000000	181024.000000	
mean	1123.112604	10.510446	0.371365	0.314986	
std	404.795499	7.518889	0.678274	0.590839	
min	0.000000	0.000000	0.000000	0.000000	
25%	816.000000	4.000000	0.000000	0.000000	
50%	1078.000000	10.000000	0.000000	0.000000	
75%	1372.000000	16.000000	1.000000	1.000000	
max	4616.000000	46.000000	8.000000	7.000000	

Figure 4 Data Summary for the sample data

Data analysis

Below is some analysis for input data to provide an overview about how these data is related to game result.

Part 1

We first calculated the average win rate for each hero. From the plot, we can see the hero's win rate is between 40% to 60%, which is typical for a well-balanced game. However, hero win-rates can differ by 20%, suggesting that hero picks are indeed important.

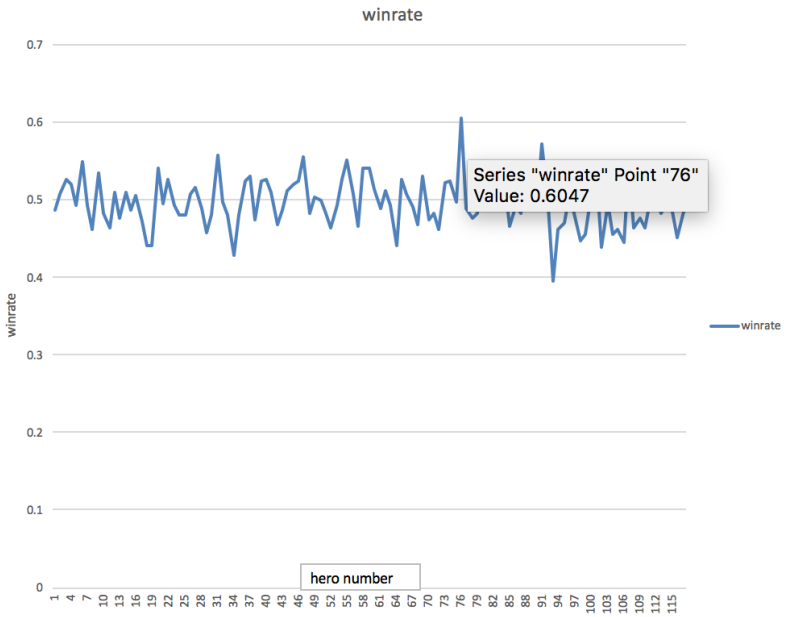

Figure 5 Average win rate vs hero number

Besides, we calculated the win rate of each hero when facing against other heroes and store the value into a matrix. Each entry of the matrix represents the win rate of the hero against another based on our collected data. Most of the win rate is still between 40%- 60%, but some are above 70%. That means one hero can be 'a counter' of another one, and picking counter can increase the win probability for his team. Therefore, hero selections can be an essential factor in the winning chances of a team.

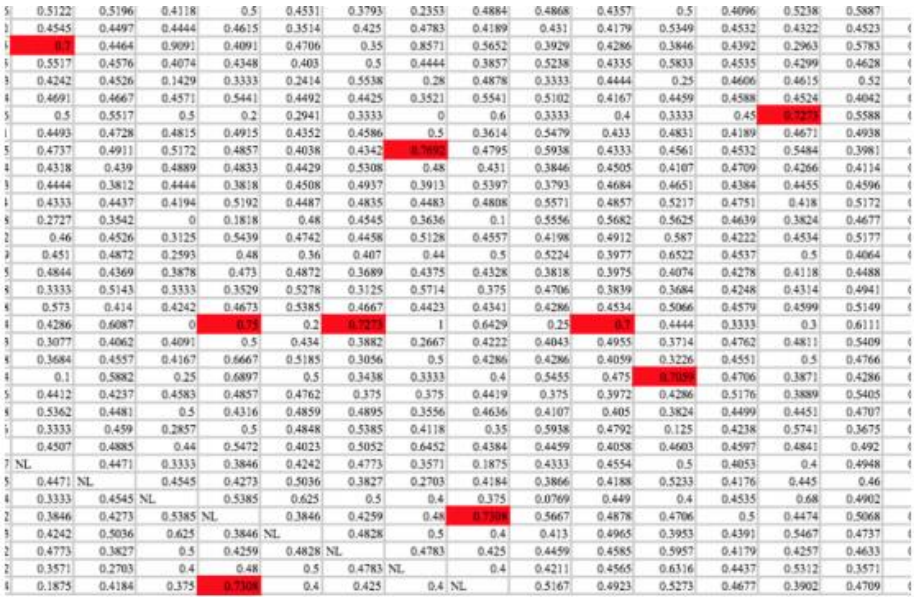

Figure 6 Matrix about win rate of each hero against others

Part 2

We plotted 1k samples of the gold and experience differences between two teams under two circumstances: Radiant team winning (Blue lines) and loss (Green). The average difference for both circumstances is marked (red and purple) in the graph. The two graphs show the same pattern, and it is evident that the blue lines are overall slightly higher than the green ones, which means that gold/experience differences are greater when Radiant team wins. The average value for the winning case is above zero, while the other is below. It follows the intuition that if Radiant team has more gold than the opponent team (positive gold difference), it has a higher chance to win. Moreover, the higher gold difference will lead to a higher winning probability.

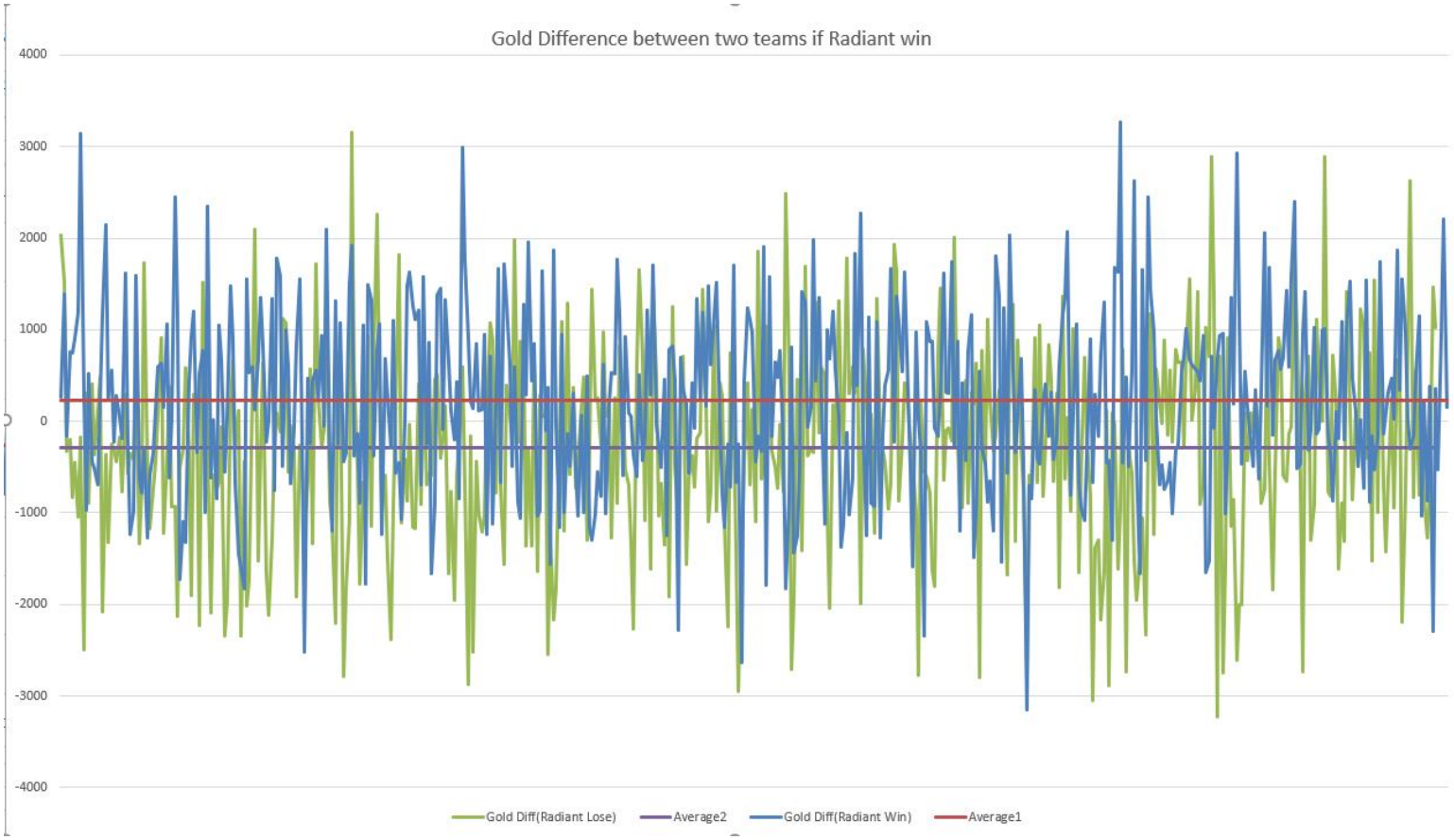

Figure 7 Gold differences between two teams

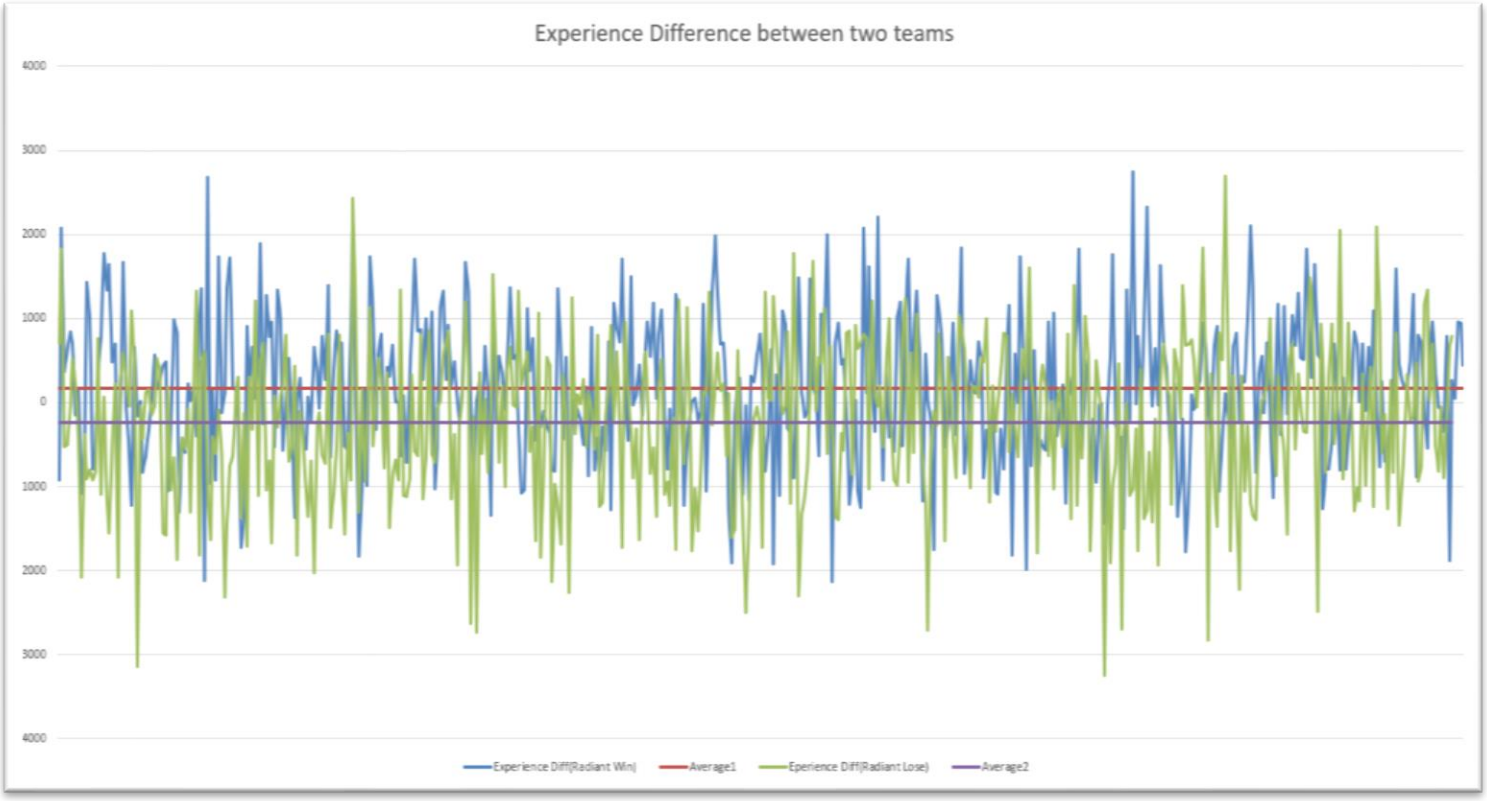

Figure 8 Experience differences between two teams

From the two graphs, we can state that gold and experience are features that do influence a game result. Furthermore, adding the gold/ experience difference as our input may result in a more accurate output.

Architecture

We used Multi-layer perceptions for both parts, as our data is simply labels and numbers. Our input data only contains 229 category classes (ten hero picks) for part 1 and two continuous class (gold/experience) added for part 2. We used hidden layers to help our network 'learn' the relationship between the features and outcome. Moreover, output data is a win probability for the Radiant team.

Below are the architecture diagrams for each part. We selected the layers from testing.

Part 1

2 fully connected hidden layers with 50 and 10 neurons, each followed by Relu function. Activation function used for output is sigmoid (to convert the output to a probability),

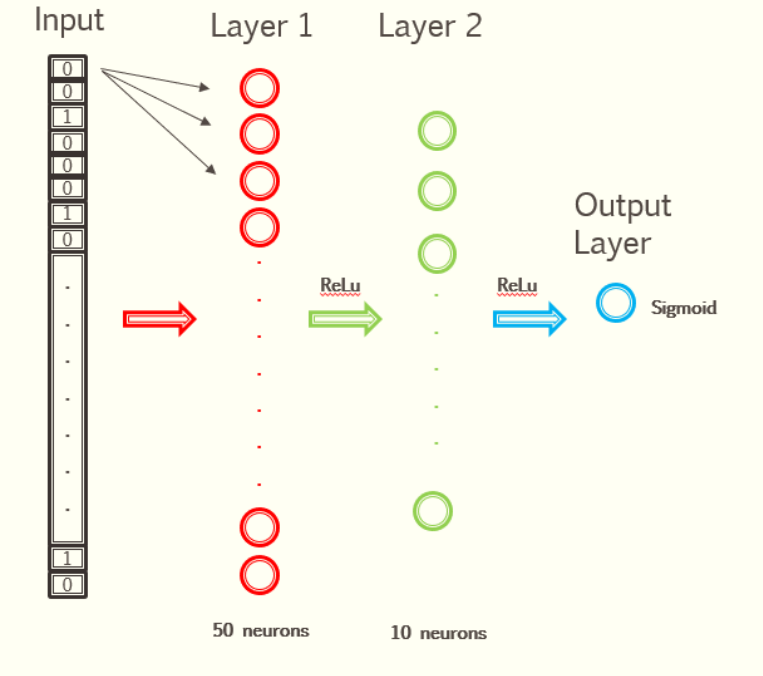

Figure 9 Model diagram for part I

Part 2

There is one hidden layer with 50 neurons followed by a Relu function and a output layer followed by a Sigmoid function.

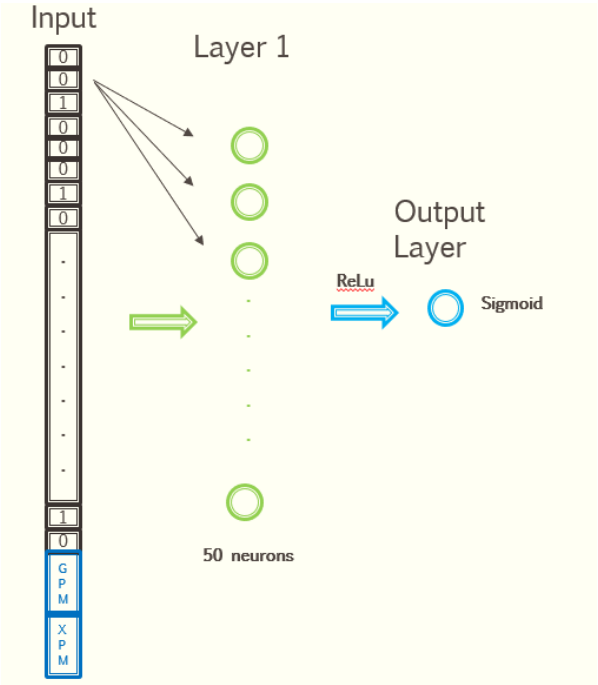

Figure 9 Model diagram for part II

Baseline model

We used Logistic regression as our baseline model. Logistic regression predicts the probability of a categorical dependent binary variable as a function of the input variable. It is simple to implement with Pytorch using one linear layer followed by 'softmax' function. The 'softmax' is an activation function that will turn the input into probabilities sum to one. In our case, the model will finally output two prediction values, each representing the probability of the winning/losing for the radiant team. Since our label is either 1 or 0, we can say a prediction is correct if the outcome with higher probability matches the label. Also, we used the Cross-Entropy Loss function to calculate the loss and train the baseline model to get a proper result. Logistic regression is a suitable baseline model because it can predict a probability that is comparable with our label, and it can efficiently deal with a large amount of input data. Moreover, the model is simple, fast, and is guaranteed to output a reasonable outcome according to the background research we did previously.

Results and Discussions

Part 1 (Quantitative)

We were able to achieve 56% testing accuracy while overfitting on the training set and using early stopping. Although this result can seem unsuccessful (only 6% more than randomly guessing: making a prediction "1" for all data will achieve 50% accuracy), it is reasonable because player performance plays a more crucial part of the game. Our results showed that hero picks alone can influence the game outcome, making us able to build the hero suggestion tool.

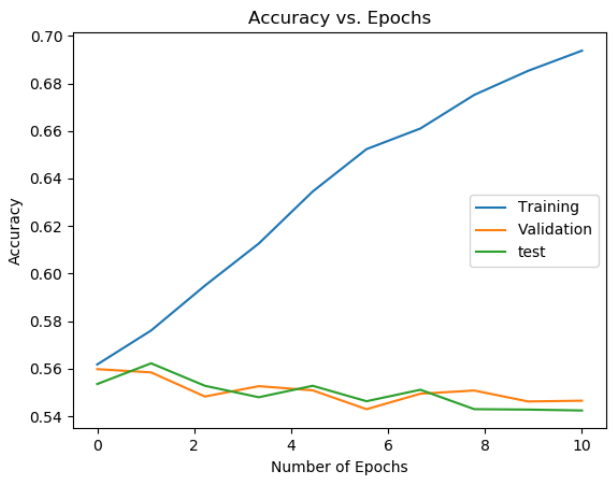

Figure 10 Accuracy vs Epochs

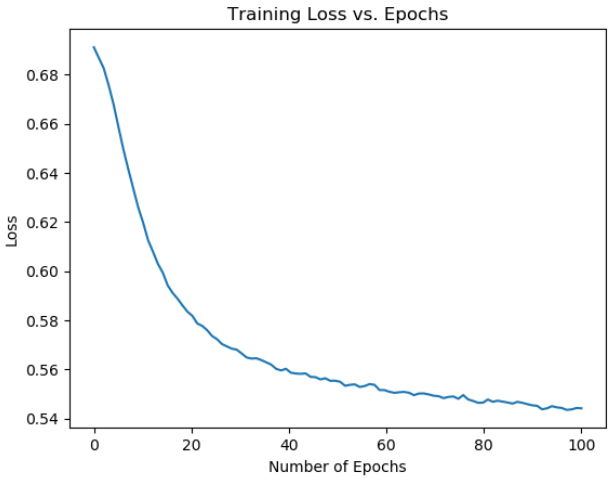

Figure 11 Training Loss vs Epochs

Comparing to the baseline model it performed ~1% better; but comparing to previous works it was not working as well as expected. This is likely due to our constraints set while collecting data.

Hero Suggestion (Qualitative)

Using our best model in part 1, we wrote a win predictor and a hero suggestion tool. Using a dictionary storing the corresponding hero indices to names, the win predictor returned the model's prediction using hero names as inputs.

The suggestion tool uses the nine heroes picked in-game and suggests picks for the 10th hero. The progress was made by running the model on all possible picks and returning the three heroes with the top predicted win probability.

Part 2 (Quantitative)

Adding the gold and experience difference for input, we achieved a test accuracy of 64%. We were also able to overfit on training data and did early-stopping. This shows that with some measurement of player performance during the early/mid-game, the outcome can be more accurate. This result can be possibly utilized to predict game outcomes during mid-game.

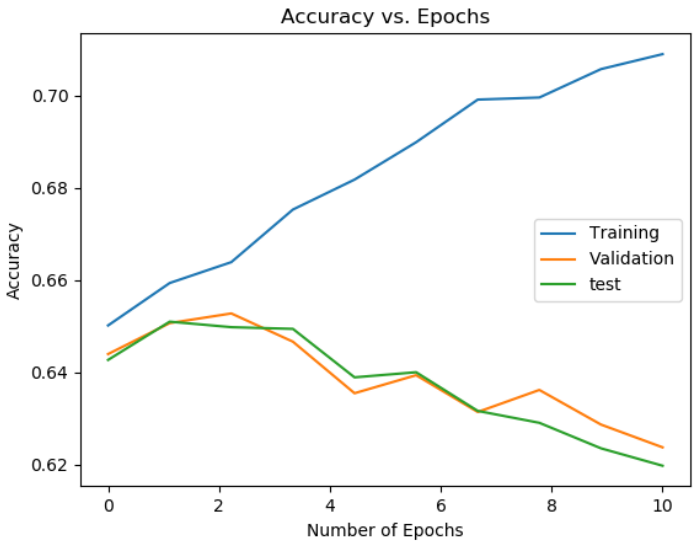

Figure 12 Accuracy vs Epochs

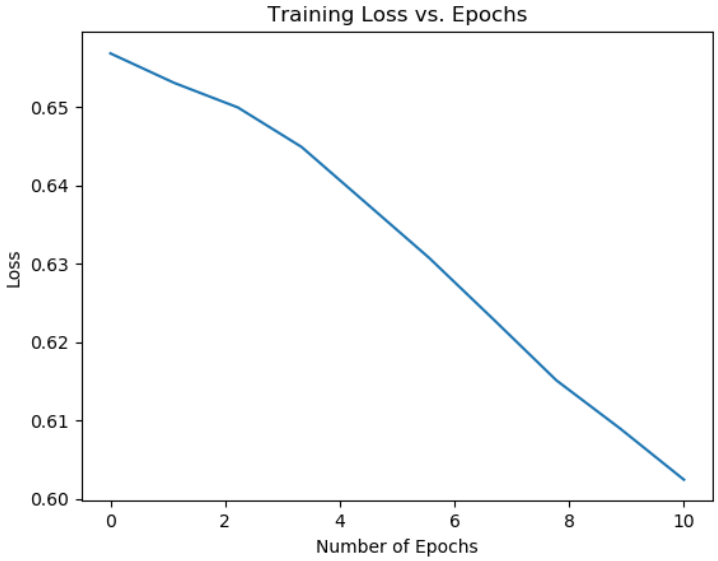

Figure 13 Training loss vs Epochs

Lessons

Our project is quite simple in architecture; data collection was the most difficult part. If starting a similar project, we would do more preliminary data analysis and propose a project that is expected to produce better results (we expected test accuracy to be ~60%), while using a more complex architecture (multiple CNNs, RNNs, etc.).

Ethic Framework

Considering players as stakeholders, we analyze the ethics of our hero suggestion/game prediction tool with reflective principlism:

Beneficence: Our hero suggestion tool can be used as a side consideration for hero-picking when players play DotA2.

Autonomy: If players are likely to follow the suggestion tool, it will limit their thoughts on hero-picking; creating ideas on hero synergies and counters is an enjoyable part of the game. Using the tool might restrict the players' freedom on establishing, or even creating their ideas.

Justice: The project can potentially make the game unbalanced. The prediction tool can benefit players who use them but may not be fair to those who refuse to use it. There is often no clear line between using an aiding tool and cheating in multiplayer video games.

Nonmaleficence: This project can minimize the risk of players (especially new players) to pick heroes that are weak for the game. It is often happening on new players or players in lower rank tiers. Our tool may limit the risks of them doing so.

Permissions:

Both of the team member : Ke Zhang and Haobang Wu grant permission to:

- Permission to post video
- Permission to post final report
- Permission to post source code

Reference

[1] Github:andreiapostoae/dota2-predictor [online], viewed Oct 24th 2019, Available: https://github.com/andreiapostoae/dota2-predictor

[2] OpenDota API (V17.7.0) [online], viewed Oct 30th 2019, Available: https://docs.opendota.com/

[3] Kaggle: mlcourse.ai:Dota2 Winner Prediction, [online],viewed Nov 16th 2019, Available: https://www.kaggle.com/c/mlcourse-dota2-win-prediction