DOTA 2 Game Result Prediction System

Yiqing Ding (yiqingd)[1], Luyao Hou (luyaoh)[2]

Abstract

Defense of the Ancients (DOTA) 2 is a multiplayer online battle arena (MOBA) game developed by Valve Corporation. The game is played between two teams of five players with each team defending their own base on the map while trying to defeat the adversary by destroying its base. Due to the gameplay's complexity, DOTA 2 is often complained as unfriendly to new players. In this project, we explored different neural network architectures to predict if a player and her team will win in the end. Specifically, we try to generate predictions for game outcomes given current game states.

Introduction

In the gameplay of the Defense of the Ancients (DOTA) 2, each player independently controls a single powerful character, known as a "hero", which has unique abilities and different styles of play. During a match, players collect experience points and gold for their heroes to successfully defeat the opposing team's heroes in player versus player combat.

DOTA 2 currently (version 7.20) has 116 heroes. Each hero has various different abilities ranging from dealing damage to adversaries to healing allies. Heroes gain gold by killing creeps (machine generated soldiers), enemy buildings and enemy heroes. When a hero dies, he/she loses gold and will respawn after certain time. Gold can be used to purchase items or buyback (immediately resurrection). Meanwhile, heroes gain experience ("xp") similarly as gaining gold points. Experience can be used to upgrade heroes which allow them to have better abilities and other properties, such as better health gain, faster moving speed, etc.

A team wins by being the first to destroy a large structure located in the opposing team's base, called the "Ancient". To achieve this goal, it is necessary for players to not only score as much experience and gold as possible, but also utilizing appropriate strategies against adversaries.

This strategy requirement and complexity of the game pose challenges to many players who are not that experienced in the game. On the one hand, it is hard for players to assess the game condition accurately. While professional players often rely on their knowledge from massive game plays, amateur players are quite helpless on determining the current game progress. On the other hand, decision making in the game is often challenging for new players since many factors can contribute. Therefore, we decided to use neural network to train a game prediction system to help players make assess the situation and make the right decision in the game.

[1] Department of Aeronautics and Astronautics, Stanford University
[2] Department of Compute Science, Stanford University

Dataset

OpenDOTA is a volunteer-developed, open source platform providing DOTA 2 match statistics and replays. All the data is collected through Steam WebAPI and OpenDOTA enables developers to collect data using its API. Example match statistics that OpenDOTA provides include but are not limited to when each building is destroyed, the gold and per minute of a player, the item purchase log of each player, and the skill upgrades log.

OpenDOTA provides data for both public and professional matches. Professional matches are games between professional teams while public matches are played by amateur players. In this project, we opted to use only professional matches for several reasons. First, although Steam tries to group players in public matches based on their matchmaking ratings (MMR), it also tries to make sure each team has a similar number of parties, i.e. matchmaker tries to avoid matching a party of 5 against 5 individual players. Therefore, it is not uncommon to match players with different skill levels, which could easily result in landslide victory for one team. Additionally, due to the entertainment nature of public matches, players usually do not take the game seriously. This attitude sometimes results in them making irrational decisions such as picking a really bad hero, etc. Moreover, professional players usually make the optimal decisions under different game states, so using professional matches makes the system more accurate for assessing the progress of each team given a game state.

We retrieved 13640 professional matches from OpenDOTA's API. We derived 10 data samples from each match, where each sample has features related to a specific player and a binary output depending on if the player wins. Some features from OpenDOTA have variable length for different players, for instance, one of the players may get 10 kills in the entire game while another might only have 2. We engineered these features into constant length vectors, by, for example, turning the kill log is into the number of kill plus the time for latest kill. Finally, we divided the 136400 samples to 116400, 10000, 10000 for train, dev and test respectively.

Methodology

We developed two models to predict if a team will win a DOTA 2 game. The first is a plain deep neural network, which allows us to predict game result given either game state at a specific time, for instance, 15th minutes, or game state at any given time plus the time of that state as an additional feature. The second model is a RNN, which allows us to predict game result at all timesteps at the same time.

Architecture 1: Plain Neural Network

The input size for our plain neural network ends up being 3561. This is large because we have to encode the history of game states into a single vector. For instance, to encode item purchase log, we make a one-hot vector of items, but the number of items in DOTA 2 is over 100.

We started with shallow networks. One of which has two hidden layers of sizes 400 and 10 respectively and learning rate 0.0002. However, that did not work as the training loss plateaus quickly and the training accuracy stays around 50%. We in the end used the following model, as shown in Table 1, with a learning rate of 0.00025. Each hidden layer uses RELU activation function except the output layer using sigmoid.

Table 1. Plain Neural Network Model

Layer #	1	2	3	4	5	Output
Hidden Unit #	3000	2000	500	100	20	1

It can be hypothesized that prediction accuracy would be much higher at the end of the game than at the beginning of the game. Therefore, we trained our neural networks on both game states at 15th minute as well as states from mixed times in game.

Architecture 2: Recurrent Neural Network

We used the following model for our RNN. We used the same architecture to train with both LSTMs and GRUs.

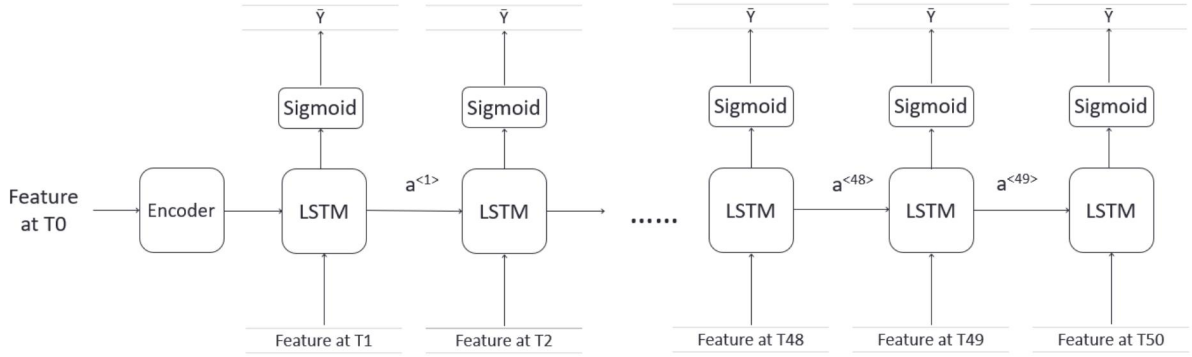

Figure 1. Recurrent Neural Network with LSTM cells

Each step in the RNN represents one minute of the game. We used Tx = 50 as most games end around 40 minutes. If a game ends at t < 50, then the last (50 – t) inputs are the same as the input at step t. The input size to each RNN cell is 408 because we do not need to encode the history of game states. For example, we assumed that each player buys at most 3 items each minute, which avoids the one-hot vector we used in plain network. We used weight matrix of size 64 for both LSTM and GRU. The LSTM model has learning rate 0.01 and GRU model has learning rate 0.001.

Results & Analysis

Plain Neural Network

We plot a training and test accuracy versus training epoch graph for both 1) when the network is trained on only game states from 15th minute and 2) when the network is trained on game states from mixed times ranging from 5th minute to end of game. The result is shown in figure 2.

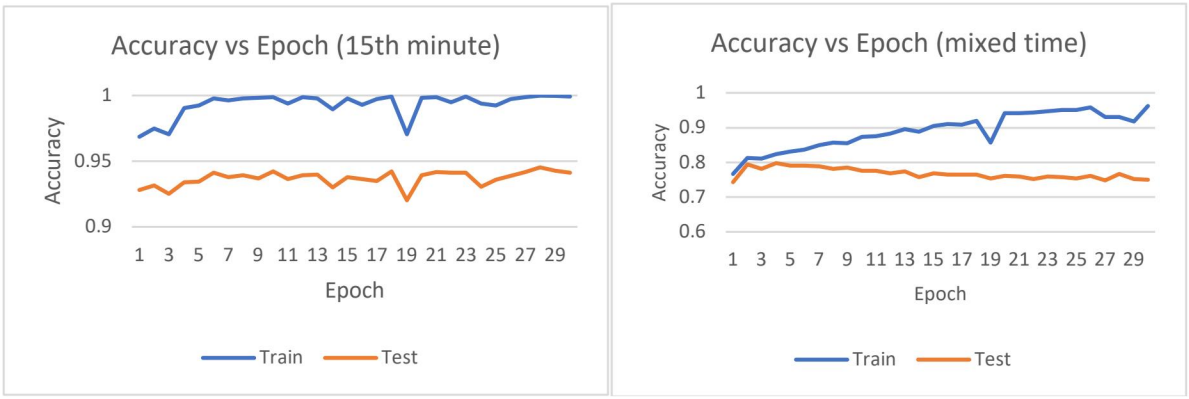

Figure 2. Accuracy vs. Epoch for game states t = 15 min and mixed time

Figure 2 shows that the train and test prediction accuracies are both above 90% given the state information at 15 mins, even though there is notable variation. We suspect that the high accuracy could result from professional players knowing how to build on their advantages early in the game.

In Figure 3 where the inputs are games states from mixed times, the train accuracy continues improving, but the test accuracy seems to be stagnating and even slightly decreasing. This shows a sign of overfitting and we suspect that this issue is primarily due to the difficulty of learning the new time feature, which makes it hard for the neural network to predict unseen matches. The distribution of training and test data can introduce overfitting as well.

Recurrent Neural Network

We tested LSTM and GRU with the sequential feature vector defined in previous sections. Since batch gradient descent was used in optimizer, some fluctuation of accuracy can be seen in Figure 3 and Figure 4. As the epoch increases, both algorithms arrive at stable equilibriums.

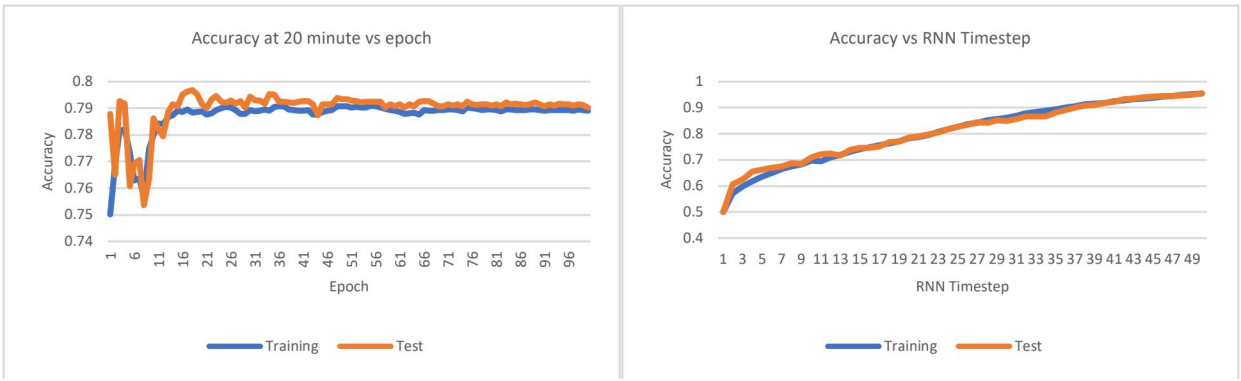

Figure 3. LSTM Accuracy - Epoch and Timestep

Two important observations can be made from Figure 3 and Figure 4. First of all, it is apparent that as the time increases, accuracy increases gradually. When reaching the end of the game, RNN can reach a nearly 100% accurate prediction. Second, we found that LSTM generally performs better than GRU given that we tuned the hyperparameters for each. The result matches our expectation because LSTM maintains more state than GRU and is able to capture more complex functions given the same weight matrix size. Comparing the results from two architectures, the RNN has lower accuracy at a fixed time early in the game probably because of diminishing gradients.

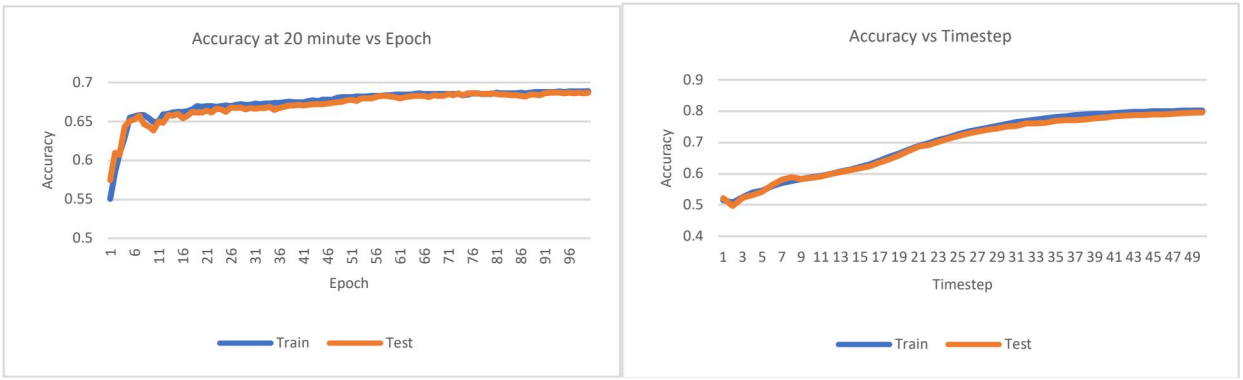

Figure 4. GRU Accuracy - Epoch and Timestep

Conclusion & Future Work

In this project, we tested different neural network architectures trying to predict if a player will win a given DOTA 2 match. The plain network performs great on game states at a fixed time, but its test accuracy suffers when inputs have states from mixed times of games. The RNN shows great accuracy towards the end of the game but slightly lower accuracy at a fixed time early in the game probably due to diminishing gradients.

For the next steps, we would like to reduce the variance for plain neural networks so that test accuracy can be improved, especially for the mixed time case because of the large space of improvement. Besides that, we want to test RNNs with more than one layers to boost accuracy, especially for early stage predictions. In the long term, we would like to use this winning prediction system to make decision recommendations to players such as which items to purchase by optimizing the winning rate with respect to different decision variables.

Code

https://github.com/hlycharles/dota2-result-prediction

References

1. Conley, Perry (2014). How Does He Saw Me? A Recommendation Engine for Picking Heroes in Dota 2. http://cs229.stanford.edu/proj2013/PerryConley-HowDoesHeSawMeARecommendationEngineForPickingHeroesInDota2.pdf
2. Kalyanaraman (2014). To win or not to win? A prediction model to determine the outcome of a DOTA2 match. http://jmcauley.ucsd.edu/cse255/projects/wi15/Kaushik_Kalyanaraman.pdf
3. OpenDOTA API. https://docs.opendota.com

Contributions

Each team member made different and significant contributions to this project. Yiqing contributed the initial idea, much of the background and data search, the GRU model as well as most of the poster and report writing. Luyao worked on data processing and feature extraction, the LSTM model and result visualizations. Both members worked on the plain neural network collaboratively.