DOTA2 Game Prediction with Deep Neural Network

Luyao Hou, Yiqing Ding
Stanford University

ABSTRACT

DOTA 2 is a popular multi-player online battle arena game. Players form two five-man team and control different hero avatars trying to defeat adversary team and destroy their fortress.

Complexity of the game implies that it is quite unfriendly to new players. On the one hand, new players are unable to assess the game situation as seasoned players. Professional players can predict game outcome event at the very beginning of the game. On the other hand, new players cannot make a right decision to facilitate the game progress. Among all the decisions to be made, the most important one in DOTA 2 is item purchase options.

In this project, we use different deep neural network architectures to investigate problems in DOTA 2 gameplaying. We start by building a winning rate prediction system using current game states. We then utilize this prediction system to explore different item purchase options to find out which items work the best.

INTRODUCTION

In DOTA 2, Each player controls a single hero. Heroes can 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).

DOTA 2 currently (version 7.20) has 116 heroes. Each hero has various different abilities ranging from dealing damage to adversaries to healing allies. Meanwhile, heroes gain experience ("xp") similarly as gaining gold. Experience can be used to upgrade heroes which allow them to have better abilities and other properties, such as health gain, moving speed, etc.

Meanwhile, DOTA 2 has a huge map where many game factors are included such as neutral creeps and runes (can be picked up by heroes to give them extra time limited capabilities).

To build the winning rate prediction system, it is necessary for us to modelling the known game state (to one team) as fully as possible. Therefore, we used all the above mentioned factors in our state representation to build the neural network.

DATASET

OpenDota is an open-source website that provides data of game matches from both public and professional games. Because of the discrepancy of gameplaying that often happen between teams in public matches, we decided to use only the professional matches where most players are on a similar level. In total, we pulled 13640 professional matches from OpenDota's API. We then divided these matches into sets of [11640/1000/1000] for [train, dev, test] purposes.

FEATURES & MODELS

We define our state vector and output as follows

$$\begin{cases} x = [Current\ game\ state\ information, such\ as\ gold, kill\ logs, etc] \\ y = Winning\ rate \end{cases}$$

Current game state information includes all the information known to the player on allied team and the state definition changes a little when training using different networks. Some features have variable length in raw data, for instance, one of the heroes may get 10 kills in the entire game while another might only have 2. We engineered these features into constant length vectors. For instance, the kill log is turned into the number of kill plus the time for latest kill.

Architecture1: Plain Neural Network

It is apparent that results of plain neural network setting vary significantly with timestamp of input state vector. Prediction accuracy will be much higher at the end of the game than at the very beginning. Therefore, we tested using both states at a fixed time versus states at a random time slice.

Layer #	1	2	3	4	5	Output
Hidden Unit #	3000	200	500	100	20	1
Activation	RELU					Sigmoid
Initialization	Xavier initialization for W and zero initialization for b					

Table1: Plain Neural Network

Architecture2: RNN

With RNN, we input the state vector at every minute of a game to predict the winning rate at each minute.

We fixed the length of the sequence to 50 as to most of the game duration and tested both GRU and LSTM.

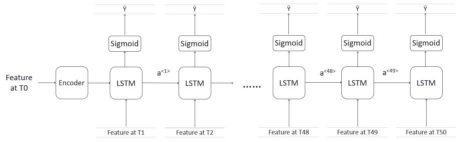


Figure1: Recurrent Neural Network with LSTM Activation

RESULTS

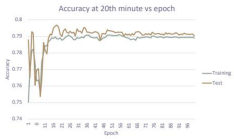
Plain Neural Network:

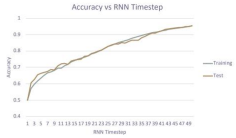
- 15 minute state

- random time slice

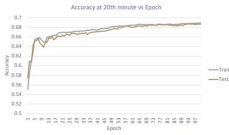
RNN:

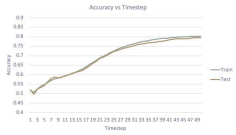
- LSTM Accuracy – Epoch and Timestep





- GRU Accuracy – Epoch and Timestep





Analysis:

- There is room for improvement in plain networks as there is high variation in the random time slice case.
- GRU in general has lower accuracy than LSTM. This is expected as LSTM maintains more state.
- Both training and test accuracies are relatively low at early stage of the game, probably due to limited amount of information and possible variations later in the game.

FUTURE WORK

- Try to reduce the variance for plain networks so that test accuracy could approach 95 to 100 percent.
- Try RNNs with more than 1 layers to boost accuracy, especially for early stage predictions.
- Utilize the winning rate prediction system to recommend items to purchase at different times in game.