FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Prediction of Dota 2 Game Result
Student:	Filip Beskyd
Supervisor:	doc. RNDr. Pavel Surynek, Ph.D.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of winter semester 2019/20

Instructions

The goal of this thesis is to design and evaluate techniques for prediction of team performance in the Dota 2 game. Teams in the Dota 2 game consist of several heroes each with unique abilities [2]. After teams choose composition of their heroes, a team competes against other team. The outcome of the game depends on the composition of heroes and subsequent decisions of players. A student will study machine learning techniques applicable in prediction of the outcome of the game based on the knowledge of composition of participating teams [1, 5]. A special regard will be devoted to the form of available input data. A selected machine learning technique accompanied with data preparation methods will be implemented as a software prototype and evaluated experimentally on existing game databases. It is expected that the student will implement the prototype with the help of suitable machine learning library [3, 4].

References

[1] Russel, Norvig: Artificial Intelligence: a modern approach, Prentice Hall, 2003.
[2] McDonald: A Beginner's Guide to Dota 2, PC Invasion, 2016
[3] OpenCV team: Open Source Computer Vision Library, https://opencv.org, [February 2018]
[4] MLPack team: Machine Learning Pack, http://mlpack.org, [February 2018]
[5] Mitchell: Machine Learning, McGraw-Hill, 1997.

Ing. Karel Klouda, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague March 1, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Prediction of Dota 2 Game Result

Filip Beskyd

Department of Applied Mathematics
Supervisor: doc. RNDr. Pavel Surynek, Ph.D.

May 14, 2018

Acknowledgements

In this place, I want to thank my supervisor doc. RNDr. Pavel Surynek, Ph.D. for providing guidance, consultations, his time and valuable advices throughout whole time I was writing this thesis. I also want to thank my family for continuous support during my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on May 14, 2018 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Beskyd, Filip. *Prediction of Dota 2 Game Result.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

Teoretická část této práce se zabývá stručným vysvětlením teorií rozhodovacího stromu a umělých neuronových sítí a vysvětlením základních faktorů které mají významný dopad na výsledek hry. Praktická část se zaměřuje na experimentování s parametry použitých technik strojového učení, rozšiřování vstupních dat o informace týkajících se složení hrdinů, porovnávání a vyhodnocení výkonu těchto rozšíření. Toto vše končí implementací experimentálního programu, který vytváří prediktivní ANN model. Tento model může být později použit pro predikci výsledku hry podle počáteční kompozice hrdinů v týmu.

Klíčová slova predikce výsledků hry Dota 2, strojové učení, rozhodovací strom, umělá neuronová síť, Python, scikit-learn

Abstract

Theoretical part of this thesis will focus on briefly clarifying decision tree and artificial neural network theory and explain basic factors that have significant impact on game result. In practical part, focus is set on experimenting with used machine learning technique's parameters, extending input data by information regarding hero compositions, compare and evaluate performance of these extensions. All of this resulting in implementation of experimental program which will produce predictive ANN model. This model can be used later to predict the outcome of the game based on knowing initial team's hero compositions.

Keywords predicting Dota 2 game result, machine learning, decision tree, artificial neural network, Python, scikit-learn

Contents

List of Figures

List of Tables

Introduction

Dota 2 is the most popular MOBA (massive online battle arena) game, being played daily by hundreds of thousands of unique players from all around the world, both amateurs and professionals, therefore it is in the spotlight of e-sport scene.

There exists few online betting websites which allow betting on Dota 2 matches. The goal of this thesis is to create program prototype using suitable AI prediction methods. This program can be used by people who are betting. Using various prediction methods can be useful in helping person who is betting money in giving him an informed opinion on which team has higher chance of winning the game based on initial team hero composition.

Topic of predicting game's outcome is interesting, basically anyone who is about to watch an ongoing Dota 2 match is interested in knowing in advance which team will win, or who has higher chances of winning. Highly spectated matches with significant money prizes between top teams in the world are broadcasted online, the commentators are constantly evaluating situation that is currently happening in the game thus predicting who has better chances at them moment. Technology giant Google's team who is working on many interesting AI projects has also put their interest on this problem as a small subset of very complex problem which they are trying to solve, that is, to let entirely AI-driven team play game against human players, and also how much useful and precise can AI get in predicting game results based on some statistical data.

In this thesis I will discuss relevant theory behind the part of machine learning library I will be using in implementation, present results, draw conclusions and recommend further improvements. I will not provide software analysis, since most of the program will be using library which was programmed and documented by library authors.

In first chapter I will introduce reader to the rules of the game and how Dota 2 game is progressing into later stages of the game, with special regard on first stage. Second chapter will explain theoretical base for prediction models,

which I will employ in my prototype. Third chapter briefly discusses exiting solution. Fourth chapter discusses methods of acquiring data for learning and also preset representation of data. is going to be the most important part, comment on my implementation. Fifth chapter is the core of this thesis as most of experimental work is done and discussed there. Final chapter presents achieved results and selects best model for predictive task of this thesis.

Goals

Goals of the theoretical part of this thesis is to make reader acquaint with basic rules and mechanics of the Dota 2 game necessary to understand some of the factors which contribute to the result of the game. Provide insight into how some of the most common supervised machine learning techniques capable of predicting or classification work.

Practical part's goal is to implement predictive model, but mostly prepare and boost input data with information about initial knowledge of each team's hero composition, compare how these informations help certain techniques in predicting correct result and ultimately compare results with already existing predictors.

Chapter 1

What is Dota 2?

Dota 2 is a free-to-play multiplayer online battle arena (MOBA) game developed by Valve Corporation. The game is based on the original mod, Defense of the Ancients (DotA) for Warcraft III: The Frozen Throne.

Despite some criticism going towards its steep learning curve and complexity, the game was praised for its rewarding gameplay, production quality, and faithfulness to its predecessor, with multiple gaming publications later considering it to be one of the greatest video games of all time. Since its release, Dota 2 has been one of the most played games on Steam, with over a million concurrent players at its peak.

The game is played in matches between two teams consisting of five players. Every player independently controls a "**hero**".

Objective of both teams, is to destroy other team's most important structure on the map, known as "Ancient".

1.1 Experience and gold

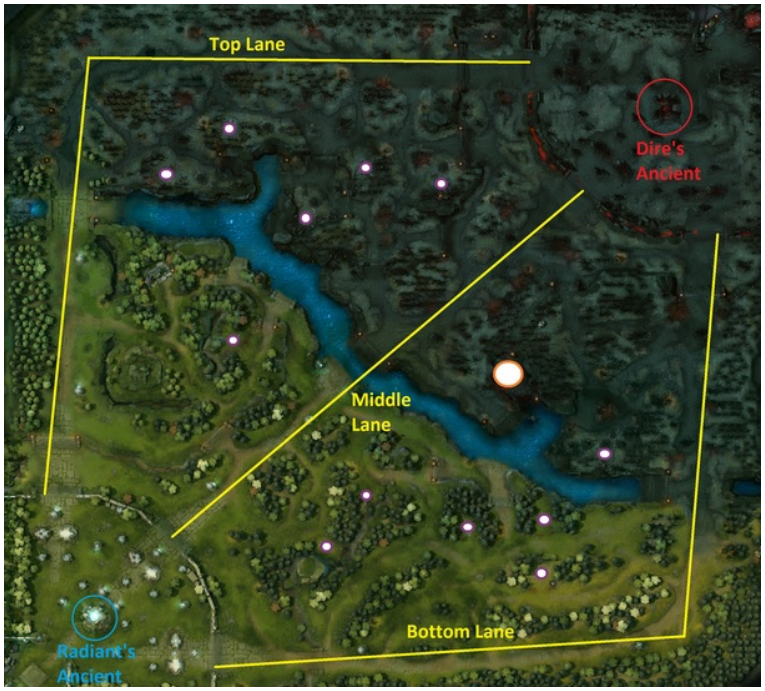
Every hero gets stronger over time by achieving higher level. Leveling-up is done by collecting experience points by killing either enemy heroes or neutral "creeps" on the map. To earn experience in Dota 2 heroes must be within a certain range of a dying enemy creep, this experience is shared between all allied heroes within that range.

For every new level a hero's attributes increase, and usually hero learns new ability which players use in fights to their advantage. Players spend gold they earned to buy various "items" for their heroes making them more durable, swifter and stronger by further increasing their attributes, some of the items also grant hero additional abilities.

1.2 Stages of the game

Each match of Dota usually runs through four stages. In each stage, players focus on different objectives. Different kinds of heroes shine in different stages.

Figure 1.1: Dota 2 map [1]



1.2.1 First stage

The very first stage of the game is called ”pick phase”, its purpose is that both teams choose from the pool of available heroes. This stage is the most important one from the perspective of this thesis. The pool of available heroes depends on the game-mode this match is being played in. Choosing hero often called ”picking” is turn-based, this means teams are alternating and only one team is able to pick at the time.

1.2.2 Second stage

A ”laning phase”, Dota 2 map consists of 3 main lanes, top lane, middle lane and bottom lane as shown in 1.2. Players chose on which lane they will be playing first stage that is around 10-15 minutes usually. There are some occasions in which players change their lane to use their hero in best possible way and maximize hero potential. This stage is usually calmer than other, meaning that not too many fights happen, players focus on gathering as much gold and experience as they can, to convert their advantage into kills later.

1.2.3 Third and fourth stage

Middle game and late game are similar, players are usually equipped with powerful items and well leveled, most of the fights happen during these stages, I would argue, most of games are more less decided in middle game already, especially in games with higher skilled players, as in late game players usually focus on capitalizing on their team advantage and destroy enemy base buildings and ultimately "the ancient" building.

1.3 Game-modes

Currently there are twelve game-modes in the game, they serve purpose to add extra variety to matches, allow players get more comfortable, since some players prefer to play only few heroes in hundreds of their games and become specialists with certain heroes, other prefer to try new things and experiment a bit. In this section I will only describe four out of all twelve game-modes, most popular, other game-modes differ only slightly from these four. Additionally matches in "all pick", "random draft" and "captain's draft" game-modes have **ranked** versions. Ranked match differs from their unranked counterparts, by the fact that players are collecting points for winning the match. This allows players to play games against players of similar skill making games more even, skill-wise at least and be featured in global Dota 2 leader-boards, in other words, ranked games are taken more seriously and players usually perform better.

1.3.1 Captain's draft

This game-mode is played in professional games, it allows teams to not only pick but also "ban" 12 heroes, banning means removing hero from the pool of available heroes. Teams use this to build strategies, banning heroes which are especially good against heroes they have already picked or planning to pick to protect certain heroes from being countered later in the game.

1.3.2 All pick

Every player chooses any hero player desires to play from the pool of 115 heroes. This is the most played game-mode in DotA. At the beginning every player can nominate one hero for banning. Half of the nominated heroes is randomly selected and banned

1.3.3 All random

Players are assigned random hero they will play, additional option is available in this game-mode, players are allowed to "re-pick" hero when they dislike their hero. Re-picking means they will be assigned another hero randomly. Only

one re-pick is available for player. In this game players are given additional starting gold to buy items.

1.3.4 Single draft

Random pool of only three heroes is created for every player, he then chooses one of these three heroes.

1.4 Heroes

Heroes are the most important aspect of the game. Different compositions of heroes create different possibilities, combos in team-fights, allow players to use some of the hero's abilities in conjunction with other abilities to create powerful "chain reactions".

Every hero has three attributes, strength, agility and intelligence, and of course many other but I will not provide all the details for the sake of simplicity. Every hero has one "primary attribute" out of these three, that means, their primary attribute grows faster rate than other two attributes as hero gains higher levels, that gives a sketch or an idea of what his strengths are going to be. For example, some heroes are better at defending, other are better at attacking. Certain heroes are more suitable for certain play-styles, as time advanced people playing Dota started realizing that, and divided heroes into few basic groups.

Next, I will only briefly describe some of the common classifications or "roles" of the heroes, because I think deep details are not necessary for the sake of understanding this thesis, but understanding that some heroes are more "important" than others is needed.

There is no hero in game which perfectly fits any of following definitions. Hero is usually combination of many roles, this is one of many other reasons why hero composition makes game so complex, it is never clear at the beginning of the game which hero will be prioritized by the team, enemies must figure this on their own as the game progresses and act accordingly to do their best to counter enemy strategies.

1.4.1 Carry

Carry is considered position #1, because it's role is to collect more gold than the other players on his team, the primary goal of a carry is to increase their strength as quickly as possible. The primary reason why it is important is because their hero ability scale very well as items ar purchased and levels are gained. Purchasing items that are even slightly more efficient than an opponent's can provide a major advantage during team-fights.

The carry's teammates will work to ensure the carry is able to collect gold efficiently and stay safe. Almost all of the carry's time in the game should be

spent leveling up. The only times the carry might not be doing that are when teams are fighting. Durning a teamfight it is very important for a carry to be present, they can dramatically impact the battle, and if the fight goes well, they earn gold and experience from killing enemy heroes. [6]

Carry heroes truly come into play later when they converted gold into powerful items which increase their potential of killing enemy heroes. Strongest carry's primary attribute is agility, as each agility point equals increase in their damage and attack speed.

A carry's ability to kill opponent heroes usually exponentially increases as a function of their items and this is reflected in what is known as the "snowball" effect in the DotA2 community i.e., a well played carry hero often leads his team to victory almost single-handedly. [7]

1.4.2 Support

Supports serve an aid purpose, provide map vision for team by buying special items known as "wards".

They usually come with helpful abilities for their team, such as healing spells or abilities that disable enemy heroes, thus making them unable to move and escape from dangerous situations, and generally are more centered around utility than damage.

They do not rely on gold as much as carries, but they do reasonable amount of experience. They usually assist carries in early stages of the match. It is common to see supports sacrifice themselves in order to let heroes with higher importance escape.

1.4.3 Off-lane

Off-lane role also known as position #3, role is one of the most dangerous in all of Dota 2 roles because of the dangers that an off-lane player must face.

Heroes chosen for this role are often those that have high health and armor abilities which allow them to escape and most importantly, benefit the most from experience gains.

Typically the off-lane player is placed alone against three enemy heroes usually including the enemy carry and support players, while this means it can be dangerous to move within experienced range. It also means that because they are alone, off-lane player will be able to earn experience much faster than their enemies.

Off-lane players actions can have a cascading impact on the enemy carry and support. The off-lane player attacks the enemy carry, limiting the carry's ability to collect gold freely and efficiently. If it's done correctly this will help provide them with the gold they were unable to earn previously and give them an extra boost to their experience and levels as a match progresses.

Off-lane players will participate in almost all team-fights, using their abilities to pin an enemy or turn a battle to their team's advantage. [8]

Chapter 2

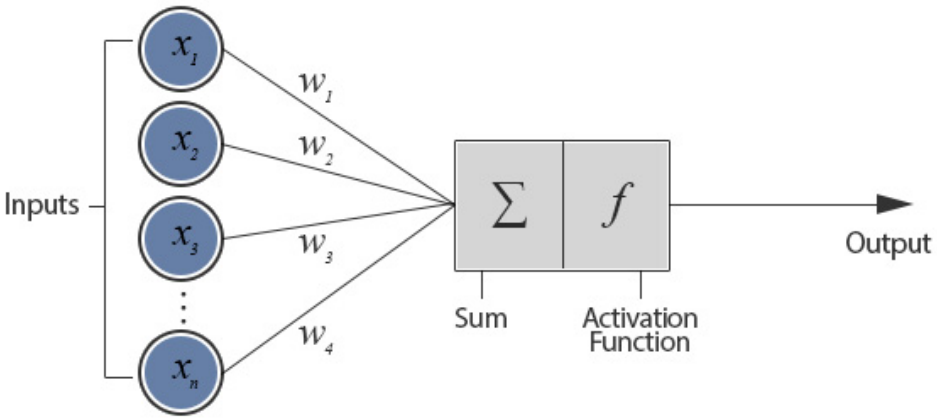
Theory

2.1 Artificial neural network

Artificial neural network (**ANN**), does not have any formal definition, yet ANNs are one of the main tools used in machine learning.

ANNs were inspired by human brain's neural networks, in which we have complex systems of smaller interconnected computational cells called "neurons". Neurons can have thousands of inputs, these inputs are somehow processed in a neuron, which represents a "function, a sum". Each neuron produces one output, but this output is only active, or taken into account, if the neuron's function output meets a certain "threshold".

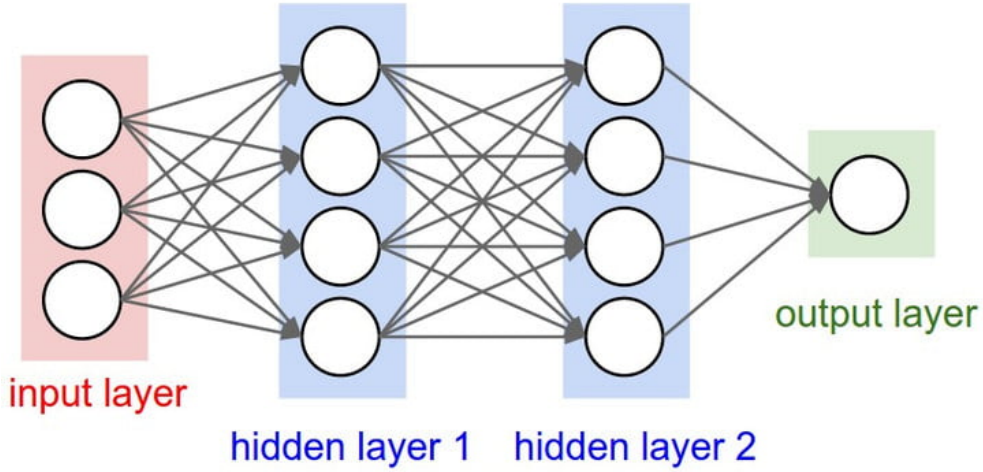
Figure 2.1: Artificial neuron [2]



As shown in the picture 2.1, "activation function" serves this purpose. In computer science it is usually non-linear function such as hyperbolic tangent or logistic function, although also the "ReLU", $f(x) = \max(0, x)$ which is non linear is used. It can be seen that every input is weighted my some values.

Complex problems problems require far more than just one neuron, as in our brains, artificial neurons can be arranged to form huge networks, these arrangements are called topologies on the ANN.

Figure 2.2: Artificial network with hidden layers [3]



ANN can have many hidden layers and many neurons in each layer. While neural networks (also called "perceptrons") have been around since the 1940's, it is only in the last several decades where they have become a major part of artificial intelligence. This is due to the arrival of a technique called "back-propagation", which allows networks to adjust their hidden layers of neurons in situations where the outcome doesn't match what the creator is hoping for. [3]

Backpropagation or backward propagation of errors, this algorithm does two things, propagate errors, and then update weights. Input is propagated forward through all the ANN's layers until it reaches output layer. Output is then compared to desired output, error is calculated using "loss function" and this error is then propagated back into the network and each neuron which contributed to initial output will adjust input weights accordingly.

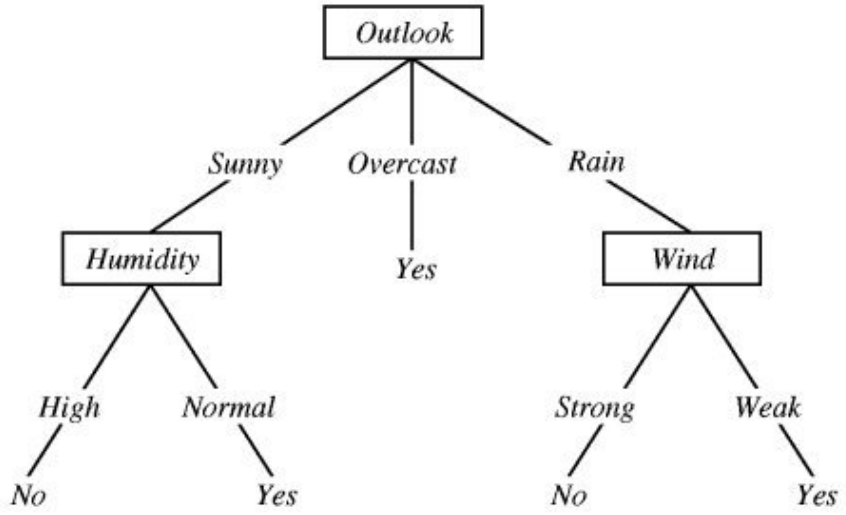
For mor information about Backpropagation algorithm see [9].

2.2 Decision tree

Decision tree learning is one of the most widely used and practical methods for inductive inference. It is a method for approximating discrete-valued functions, in which the learned function is represented by a decision tree, that is robust to noisy data and capable of learning disjunctive expressions. [9]

Nodes of decision tree can be divided into two classes, leaf nodes and inner nodes (nodes with at least one child node). Leaves are terminated and contain final "decision" or a classification of the input vector. Each of the inner nodes contains a "decision rule". This rule is then tested for one of the attributes of input vector which is further classified in lower levels of the tree.

Figure 2.3: Decision tree for classic "Play Tennis" classification [4]



The are many measures by which decision tree selects attributes for classification, two most widely used measures are **information gain** and **gini index**. Information gain, a statistical property that measures how well a given attribute separates the training examples according to their target classification, information gain is based on measure commonly used in information theory called "entropy" [9].

Figure 2.4: Information gain for the attribute

$$InformationGain(S,a) = \underbrace{Entropy(S)}_{\text{Entropy of parent}} - \underbrace{Entropy(S|a)}_{\text{Entropy of child node}}$$

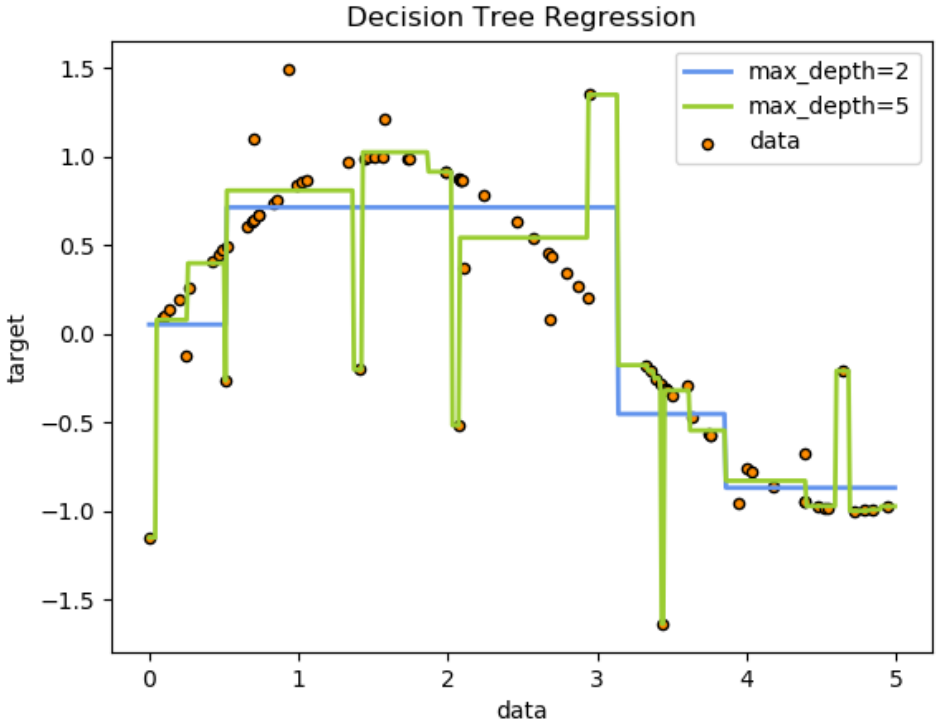
$$InformationGain(S,a) = -\sum_{i=1}^{j} p_i \log_2 p_i - \sum_{a} p(a) \sum_{i=1}^{j} -p(i|a) \log_2 p(i|a)$$

Other measure is **gini index**, defined as $Gini(S) = 1 - \sum_{i=1}^{j} p_i^2$ and it is roughly same as informational gain measure.

Decision trees can be used either for classification or regression problems. Classification trees are used when we want to classify input into known classes.

Regression trees are used when the predicted variable can be a real number.

Figure 2.5: Example of regression tree [5]



Chapter 3

Existing solutions

3.1 Support vector machine

Git-hub project written in Python is using support vector machine (SVM).

A final project in an undergraduate machine learning class. It is a simple win/loss predictor that uses a hinge loss classifier to try to predict the outcome of a match before it has started. [10]

It's best result achieved was around 59% accuracy at the time I ran this project on my computer on fresh data. Reported accuracy was 60%.

I have decided to use this project as base from my thesis, although heavily reworked part for compiling, acquiring data and extending data-set with additional vector features.

Original project is using individual hero win-rates and hero role information. This predictor is using information about location where certain match was played geographically as every match is hosted on remote servers around the world, I do not quite understand why this was taken into account as I think this information does not/should not have any significant impact on match results. Additionally predictor is provided information about game duration, this in my opinion is very significant match parameter, since by experience is it often obvious from the composition of heroes whether teams will try to finish the match fast before enemy gets their strongest hero out-of-control, since their heroes might be oriented to quickly force fights and try to finish the game, because their strength in late game will fade away and will be insufficient to compete with their counterpart.

3.2 Logistic regression

Project from Stanford "How does he saw me? A Recommendation Engine for Picking Heroes in Dota 2" [11] constructed simple predictor based on logistic regression, test accuracy reported on data-set of 18000 games was 69.8%. Their data-set was consisting of only games played in "very high skill" bracket.

The skill level of the players is very-high,which corresponds to roughly the top 8% of players. We believe utilizing only very-high skill level matches allows us to best represent heroes at their full potential. [11]

In this thesis I have included all matches regardless of skill bracket it was played in. In this paper they also present hero recommendation engine based on k-nearest neighbors algorithm, although I prefer not to comment on this one as recommending hero is slightly different task from predicting result.

3.3 Augmented logistic regression

Project written in Python, although I was not able to find source code of the program bound to this work. Model is based on above mentioned work [11], which used logistic regression with very nice results around 69% accuracy.

Since we consider only the pre-game state for our predictive task i.e., the game state upto and including the hero picks, our analysis is only upon the different hero compositions and not on any other aspect of the game or the players of the game. It is true that a game as complex and unpredictable as DotA2 is influenced by hundreds of factors other than the team composition, such as, player skill, gold and experience growth, the ability of the team members to coordinate among one another, any sudden and critical team-fights that could change the game etc. We have used simple logistic regression, without regularization from the sklearn.linear model library. [7]

Extension lies in computing co-occurrence network which is a graph, then applying Louvain optimalization of Girvan-Newman community detection algorithm on this graph was used to discover "hero communities".

The Louvain method is an efficient, greedy optimization based community detection algorithm which has been used with considerable success for very large networks (for up to 100 million nodes and billions of links). It uncovers hierarchies of communities and allows fine grained control over the size of communities, number of communities and the discovery of sub-communities. [7]

Combining these two methods revealed pairs of heroes which stand out, when these pairs occur in same team, they perceptibly increase their team's probability of winning. This extension brought significant increase in overall accuracy of the basic logistic regression model to 74%. Data-set used included only very-high skill matches.

3.4 What is missing?

None of aforementioned solutions has employed neither artificial neural network nor decision tree for predicting Dota 2 results. Therefore I think it is interesting to try these two techniques in this thesis and observe how they compare to classifiers which are usually used to classify problems where output is one of two values, such as SVM's and logistic regressions are.

Chapter 4

Data

4.1 Acquiring data

The data I worked with in this thesis were obtained from public Dota 2 Steam API. Steam API provides data in JSON format, each GET request on API provides data of 100 matches starting from *match_seq_id*, which is something similar to match ID. Program downloads data and needs to be stopped manually depending on how much data I wish to use. Downloading and parsing data is part of the program. I downloaded approximately 3.5 GB of data, which after filtering out bad data makes approximately 140000 matches. This data-set needs to be further processed because it often contains unfinished, incomplete, corrupt matches or matches which duration is too short and I consider them not being representative enough to be used as training examples.

4.2 Data representation

For learning classifier from scikit-learn library [5], downloaded JSON data first need to be put into matrices. Raw downloaded data contain lots of useless information, I will only be using attribute *radiant_win*, which is binary attribute, if it is set to be 1, it signify that Radiant team won, otherwise Dire team won, next I will keep *hero_id* attribute of each player, this integer attribute ranges from 1 to 120, these numbers are the IDs of heroes available for picking in game, some of the IDs are unused, but this should not have affect on prediction result. Currently there are 115 heroes in the game. As suitable data representation for machine learning I have chosen vector of float numbers.

4.3 Filtering matches

In two of previously mentioned projects, data were first filtered, both by game-mode, and skill-bracket of games. I decided to only focus on most played game-mode "All pick", and include games from every skill bracket. This might have big impact on the resulting predictor accuracy, since I am allowing data to include lots of games in which new players are in process of understanding the game, learning basics and thus will be making mistakes which often dismiss initial advantage of better hero composition.

Chapter 5

Experiments

In this chapter I will perform various experiments. Experiment's core consists of extending input data, more precisely, extending each basic data vector by adding more features which could be useful in predicting match results with higher accuracy.

I will be gradually expanding input data matrix, observe how additional information help machine learning models to predict with higher precision, and summarize results of such data expansion. Expansion will take place in following order:

- Basic dataset
- Adding individual hero winrate
- Adding each hero match-up winrate

5.1 Basic data-set

Data-set of basic vectors contains only information about which team won and team's compositions. 1st column signify winning team with values only 1 or -1. 2nd - 121th column value is always 1 or -1 or 0. 1 meaning Radiant team has picked this hero, -1 meaning Dire team picked hero, 0 meaning hero is not being played in the game.

Shortened basic vector explanation:
(1,0,1,-1,1,0,0,0,1,-1,0,-1) mean that Radiant won, Radiant team consisting of heroes 2, 4, 8, dire 3, 9, 11.

Figure 5.1: Example of input data matrix

$$\begin{bmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ -1 & -1 & 1 & \ddots & & & & \vdots \\ 1 & 0 & 1 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 1 & & & \ddots & \ddots & 0 & 0 & 0 \\ 1 & & & & \ddots & 0 & 0 & 1 \\ -1 & \dots & \dots & \dots & \dots & 0 & 0 & 0 \end{bmatrix}$$

5.1.1 Decision tree

Learning process was performed on data-set consisting of 10000 games. Tuning tree parameter max_leaf_nodes brought different results, the higher number I used the better results on training data I achieved, this is understandable because max_leaf_nodes manages tree model size, if the parameter is close to number of games in training set, the tree will likely remember every game. The difference between train and test scores shows, that tree does not generalize well on basic data.

Table 5.1: DT train/test scores on basic data-set

max leaf nodes	Train score	Test score
50	0.567	0.525
100	0.599	0.513
200	0.647	0.519
500	0.737	0.517
1000	0.835	0.483
5000	1.0	0.503

5.1.2 ANN

In case of ANN model has seemingly better results in comparison to DT as table 5.2 shows. It is interesting that in general, after adding more than one hidden layer model shows worse train scores and thus reducing train-test score difference, however test scores remained lower, within range 0.5-0.56.

Table 5.2: ANN train/test scores on basic dataset

hidden layer sizes	Train score	Test score
10	0.721	0.522
20	0.816	0.506
30	0.875	0.532
50	0.835	0.517
30,30	0.712	0.53
50, 50	0.627	0.567
50, 50, 50	0.526	0.511
20, 20	0.726	0.539
20, 20, 20	0.526	0.511
20, 20, 20, 20	0.526	0.511

5.1.3 Discussion

With basic vector, it seems that information about winner and team's heroes composition are insufficient for achieving any reliable prediction. Test scores close to 0.5 shows that model does not generalize well on basic data vector and is unable to learn much from simple data. Higher train score in case of DT can be explained by argument that model simply remembers train data. In summary ANN achieved better result than DT, 0.56 versus tree's 0.52.

5.2 Dataset enriched with individual hero winrates

Definition 5.2.1 (Hero win-rate) *Hero win-rate is ratio of number of games where certain hero was present and won, and number of all games in which this hero was played.*

Basic vector is now extended by two values, Radiant and Dire team projected win-rates respectively. It is computed separately for both teams as average of individual win-rates of each hero in team. Hero win-rates data were obtained from [12].

Figure 5.2: Example of input data matrix with win-rates

$$\begin{bmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 & 0.522 & 0.496 \\ -1 & -1 & 1 & \ddots & \ddots & \ddots & & & & \vdots \\ 1 & 0 & 1 & \ddots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 1 & & & & & \ddots & \ddots & 0 & 0.554 & 0.512 \\ 1 & & & & & & \ddots & 1 & 0.539 & 0.482 \\ -1 & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 0.481 & 0.503 \end{bmatrix}$$

5.2.1 Decision tree

An improvement of roughly 3% can be observed on test scores, which was expected since win-rate is very important measure of expected team success. Best improvement of test scores is observed on relatively smaller trees. It also seems that trees with smaller number of leaf nodes generalize better as difference between train and test scores decreases as well.

Table 5.3: DT train/test scores on data-set extended by hero win-rates

max leaf nodes	Train score	Test score
50	0.601	0.547
100	0.629	0.541
200	0.668	0.54
500	0.766	0.525
1000	0.866	0.52
5000	1.0	0.501

5.2.2 ANN

With hero win-rate information added train scores now increase rapidly with number of nodes added in hidden layers in comparison to basic vector data, but test scores actually decreased. At first glance, it is quite paradoxical as one would expect results to increase after adding relevant information. My hypothesis is that ANN respects these newly added values by a large margin to others and fails to generalize on unseen data, because small change in hero composition can reflect in huge change in win-rate values, in other words, I think that ANN is making classification based on win-rate values more than it should. Maybe normalization of win-rates could help.

Table 5.4: ANN train/test scores on data-set extended by hero win-rates

hidden layer sizes	Train score	Test score
10	0.791	0.523
20	0.96	0.536
30	0.999	0.503
50	0.999	0.54
30,30	0.933	0.521
50, 50	1.0	0.516
50, 50, 50	0.956	0.508
20, 20	0.893	0.501
20, 20, 20	0.524	0.52
20, 20, 20, 20	0.524	0.52

5.2.3 Discussion

Hero win-rate information does have an impact on overall precision of the models, it definitely helped decision trees, but in case of ANN's it is unclear whether this information has positive impact. Model does not generalize on given data well and it is necessary to provide additional information for better results.

5.3 Baseline algorithm

After seeing performance numbers of machine learning models on data-set including win-rates, I was expecting better results and found it suspicious, thus I have decided to construct this baseline algorithm without machine learning which would "predict" outcome of the game only by looking at win-rates alone. I summed hero win-rates for each team, obtaining 2 numbers, W_r for Radiant and W_d for Dire team.

Baseline algorithm compared these 2 values, higher W_r indicates Radiant team should win and compared this prediction with actual game result, this way baseline algorithm counted how many correct predictions over whole data-set a human would guess, considering only individual hero win-rates.

Data-set with 10000 games gave precision of surprising 57%. On bigger data-set consisting of 30000 games it was little bit lower, 55%. This might be indicating that win-rates are "worth" 5% of useful information, in comparison to random predictor with accuracy of 50%, therefore I expected machine learning to improve accuracy of at least 5% compared to random predictor.

Comparing these results with ANN results is proving that on these data it is better to use this baseline algorithm. Better results could be achieved if I could add "weights" to hero win-rates sum, because some heroes are more important than others and their win-rate should matter more.

5.4 Data-set further extended by match-up scores

Definition 5.4.1 (Hero advantage [12]) *Advantage measures the match-up between two heroes regardless of their normal win rate. It is calculated by establishing their win rates both in and outside of the match-up and comparing the difference against a base win rate. The calculation is procedural and advantage/disadvantage results are not designed to be symmetrical.*

Since there are 5 players in each of the 2 teams, for every player there are 5 hero advantage scores, thus in the following experiment input vector is extended by these 50 values (input matrix preview is omitted). It is worth noting that now explicit team's hero picks are redundant in the input matrix, because information whether concrete hero is present in the match is already included in form of relationships defined by hero match-ups.

5.4.1 Decision tree

Additional vector attributes helped train scores but had minimal effect on test score. Decision tree has trouble generalizing on these data.

Table 5.5: DT train/test scores on data-set extended by hero match-up scores

max leaf nodes	Train score	Test score
50	0.629	0.548
100	0.668	0.539
200	0.729	0.54
500	0.856	0.525
1000	0.962	0.514
5000	1.0	0.515

5.4.2 ANN

Hero match-up scores brought improvements to test scores, but it is obvious that model is heavily over-fitting data. This is understandable, because hero compositions are so unique for every game, that it is nearly impossible to obtain same game twice in practice.

5.4.3 Discussion

After this experiment I noticed that ANN's with fewer nodes in the hidden layers and more hidden layers bring better results. My hypothesis is that in previous experiments I have used incorrect ANN topology. With DT it seems to be similar issue, I have made trees too large and thus it is over-fitting.

Table 5.6: ANN train/test scores on data-set extended by hero match-up scores

hidden layer sizes	Train score	Test score
10	0.853	0.544
20	0.991	0.54
30	1.0	0.513
50	1.0	0.501
30,30	1.0	0.528
50, 50	1.0	0.553
50, 50, 50	1.0	0.527
20, 20	0.999	0.547
20, 20, 20	0.999	0.53
20, 20, 20, 20	0.998	0.55

5.5 Baseline algorithm with match-ups

To be able to do simple comparison based on which I can classify match vector as Radiant or Dire win, I needed to aggregate this vector into 2 values. I remind reader that positive value means hero is in disadvantage.

I chose to try simple summation of match-ups for each team. This might not be the best approach though, because it will likely lose some information, for example the advantage score of very important hero can be erased my huge disadvantage of hero which does not contribute to team win that much. Summation was only 50.3% accurate on data-set with 30000 games.

Second idea was to try to select and compare only team's "worst" hero, for this I used max function, from the array of Radiant and Dire hero match-ups I summed match-ups for every hero, obtaining his "expected" advantage against whole enemy team, and then selected maximum from these summations for both teams. Accuracy was 50.5% on same data-set as in previous experiment.

Lastly I tried to select "best" hero, in other words, compare two heroes which have lowest match-up score against enemy team. Accuracy of this experiment was 52.5%, 2% is significant increase hinting that selecting minimum might be good try to boost accuracy in ANN model.

Chapter 6

Final model selection

Regarding decision trees, after experimenting with various decision tree's parameters, I was not able to find right configuration to achieve results close to ANN, therefore at this point I abandoned further research on decision trees and their ability to predict Dota 2 match results and focus on ANN.

Experiments from previous chapters gave me an idea that for this problem it is more convenient to use smaller ANN's, this means, more hidden layers with sizes of 1-10 nodes. It seems that adding more than 7 layers, model starts to over-fit the data and test score decreases.

Since there is practically infinite number of ways one can configure parameters of an ANN, For final model selection I decided to write a simple script that generates all possible topologies, but limiting number of hidden layers to seven (because of computational time issues), each including 1-10 neurons. I randomly selected 100 topologies for each number of hidden layers and selected the topology with best result as "winner".

Data-set used for this final model selection included team's hero compositions, individual hero win-rates, match-up scores and lastly aggregated match-ups for "most significant hero" as in 5.5.

Following table 6.1 shows best achieved results for every number of hidden layers. It is interesting to see that hidden layers sizes are in ascending order in 6 out of 7 cases. Train-test score difference is relatively high, ranging from 5% to 11%, compared to existing solutions where the differences were 2%-3%.

Table 6.1: Best selected models per number of hidden layers

# hidden layers	Train score	Test score	Topology
7	0.65	0.598	(2, 3, 5, 7, 8, 9, 10)
6	0.658	0.588	(2, 2, 1, 3, 6, 6)
5	0.639	0.587	(2, 2, 5, 7, 7)
4	0.668	0.594	(3, 6, 8, 9)
3	0.693	0.58	(4, 4, 7)
2	0.649	0.586	(2, 8)
1	0.648	0.57	(2)

Conclusion

It was difficult to find correct decision tree configuration for this problem to achieve any consistent results, from my experiments with the input data it seems that ANN is more suitable machine learning technique for predicting Dota 2 game results.

Expanding data with win-rates brought improvements of about 5% in test scores to both models in comparison with random predictor, this proves that if teams choose heroes that are globally successful, they increase their winning chances. Head-to-head hero match-up information helped models as well with about 2% increase in accuracy, but it is not as much significant as I was initially expecting, it is much more powerful when hero pair's win-rates are considered as in [7].

My solution is not directly comparable to earlier mentioned existing solutions, mainly because data-sets used for learning differ greatly, other solutions included only games played by highly skilled players, these players do not make major mistakes often, and are able to capitalize on initial hero composition advantage, while in my solution data-set included games where new players make mistakes and even what is supposedly a "won" game, by means of hero composition, result can be opposite. I think this is the major factor in accuracy difference compared to my solution.

As regards to goals of this thesis, two machine learning technique's performances were compared and evaluated. Software prototype was successfully implemented and served experimental purposes very well. Input data were gradually expanded and merits of these expansions were discussed.

Bibliography

[1] betarena. DOTA 2 pravidla a informace o e-sportu. [image]. 2017, [Cited 2018-05-12]. Available from: https://www.betarena.cz/rubriky/sportovni-clanky/dota-2-pravidla-a-informace-o-e-sportu_2053.html

[2] the Project Spot. Introduction to Artificial Neural Networks - Part 1. [image]. 2013, [Cited 2018-05-13]. Available from: http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7

[3] DIGITAL TRENDS. What is an artificial neural network? Her's everything you need to know. 2018, [Cited 2018-05-13]. Available from: https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/

[4] The Null Pointer Exception. A Tutorial to Understand Decision Tree ID3 Learning Algorithm [image]. [Cited 2018-05-13]. Available from: https://nullpointerexception1.wordpress.com/2017/12/16/a-tutorial-to-understand-decision-tree-id3-learning-algorithm

[5] Pedregosa, F.; Varoquaux, G.; et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, volume 12, 2011: pp. 2825–2830.

[6] Dota Roles - Hard Carry [video]. Dota 2 Official youtube channel. 2015, [Cited 2018-05-11]. Available from: https://www.youtube.com/watch?v=BKE72pDhLfc

[7] Kalyanaraman, K. To win or not to win? A prediction model to determine the outcome of a DotA2 match [online]. University of California, San Diego. Class of CSE255: Dahta Mining and Predictive Analytics. 2015, [Cited 2018-04-21]. Available from: https://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Kaushik_Kalyanaraman.pdf

[8] Dota Roles - Offlaner [video]. Dota 2 Official youtube channel. 2015, [Cited 2018-05-11]. Available from: https://www.youtube.com/watch?v=9U2ap6Sy7WI

[9] Mitchell, T. M. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 3 1997, ISBN 0070428077, 52–60, 97–100 pp.

[10] Blomqvist, F. A machine learning project to predict match win/loss in DOTA 2 [online]. Undergraduate ML class project. June 2017, [Cited 2018-04-21]. Available from: https://www.github.com/fgblomqvist/dota2-ml-predictor

[11] Kevin Conley, D. P. How does he saw me? A Recommendation Engine for Picking Heroes in Dota 2 [online]. Stanford's CS229 course semestral project. 2013, [Cited 2018-05-06]. Available from: https://cs229.stanford.edu/proj2013/PerryConley-HowDoesHeSawMeARecommendationEngineForPickingHeroesInDota2.pdf

[12] Elo Entertainment LLC. DOTABUFF - Dota 2 Statistics. [online]. 2018, [Cited 2018-04-21]. Available from: https://www.dotabuff.com

[13] Python Software Foundation. Python 3.6.4 [online]. 2017. Available from: https://www.python.org

[14] Jones, E.; Oliphant, T.; et al. SciPy: Open source scientific tools for Python [online]. 2001–. Available from: https://www.scipy.org

[15] Beautiful Soup 4.4.0 [online]. 2018. Available from: https://www.crummy.com/software/BeautifulSoup/bs4

Appendix A

Acronyms

ANN Artificial Neural Network

DT Decision Tree

DotA Defense of the Ancients

MOBA Massive Online Battle Arena

JSON JavaScript Object Notation

API Application Programming Interface

SVM Support Vector Machine

Appendix B

Software description

Implementation includes simple downloader of input data from official Dota 2 Steam API in JSON format, taken from [10], converting and preparing JSON match data into more friendly format for machine learning library I used, training model module and lastly implementing module containing support functions which expand and manipulate match vector data.

In the following sections I list all third party libraries, languages and tools which I used in this thesis.

B.1 Python

For implementation I chose programming language Python [13], because it offers wide spectrum of machine learning libraries to chose from. Python applications are compatible with most of modern operating systems. Its syntax is very easy to understand and allows programmers to write easy to understand applications while focusing of code's functionality instead of language features. Python's slicing arrays syntax is very helpful for various vector manipulations which which were necessary for this project.

B.2 NumPy

For easier matrix manipulation I used Python's NumPy library [14]. NumPy also offers lots of other useful mathematical operations with arrays for example.

B.3 scikit-learn

Machine learning library which I chose is scikit-learn [5]. Technique's interfaces are easy to use and understand in conjunction with scikit-learn's in-depth documentation with demostrative examples.

B.4 Beautiful Soup

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work. [15]

I have used this library for parsing HTML page containing hero win-rates from [12].

Appendix C

Contents of attached CD

Attached CD contains data used for experiments in this thesis for possible result reproduction.

- `sources` ..... directory with source codes
  - `prototype` ..... implementation sources
    - `other` ..... directory with other necessary files
    - `python` ..... python sources
  - `thesis` ..... latex source codes of the thesis
    - `images` ..... directory with images included in the thesis
- `data` ..... directory containing input data used for learning the model
- `2018_Beskyd_Filip_thesis_oneside.pdf` .... onesided PDF of the thesis
- `2018_Beskyd_Filip_thesis_twoside.pdf` .... twosided PDF of the thesis