DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2017

On using Artificial Neural Network models to predict game outcomes in Dota 2

VIKTOR WIDIN

JULIEN ADLER

KTH Datavetenskap
och kommunikation

On using Artificial Neural Network models to predict game outcomes in Dota 2

Julien Adler
Viktor Widin

Degree project in Computer Science, DD142X
Supervisor: Jeanette Hellgren Kotaleski
Examiner: Örjan Ekeberg

June 5, 2017

Abstract

Dota 2 is an online strategy game, played in a five versus five format. Its multitude of selectable characters, each with a unique set of abilities and spells, causes every new match to be different from the last and picking the right characters can ultimately decide whether a team wins or loses a game.

This report investigates if Artificial Neural Networks can be used to predict game outcomes, based solely on the character selection made in each game. Additionally, the report will explore if altering the base parameters of the utilized ANN models can improve predictive performance. The models considered in the thesis will thus vary in number of hidden neurons and hidden layers in the neural network.

The results show that the various models have an average prediction accuracy ranging between 53-59%. We find that using a low number of neurons with many layers improves prediction accuracy. Although the results from this study seem to indicate a correlation between character picks and game outcomes, we recommend a more extensive analysis be conducted in order to reproduce these results and thus ensure external validity.

Sammanfattning

Dota 2 är ett strategispel som spelas online i ett fem mot fem format. Spelets stora mängd av valbara karaktärer, var och en med en unik uppsättning av egenskaper, leder till att varje match är olik den senaste och att välja rätt karaktär kan i slutändan avgöra om ett lag vinner eller förlorar en match.

Denna rapport undersöker om Artificiella Neurala Nätverk kan användas för att förutspå matchresultat, baserat helt på vilka karaktärer som väljs i varje match. Vidare kommer rapporten utforska om ändringar i parametrarna för de ANN modeller som används kan förbättra prediktiv förmåga. Modellerna som hanteras i denna avhandling varierar därför i antalet gömda noder och lager i det neurala nätverket.

Resultaten visar att de olika modellerna i genomsnitt förutspår matcher med 53-59% säkerhet. Genom att använda få neuroner med många lager förbättras modellernas prediktiva förmåga. Fastän resultaten från denna rapport tycks antyda en korrelation mellan vilka karaktärer som väljs och matchresultat, rekommenderar vi att en mer omfattande analys utförs i syfte att upprepa dessa resultat och således garantera extern validitet.

Contents

1 Introduction

Over the past few decades computer scientists have developed a number of machine learning algorithms that can solve problems by learning from examples rather than being explicitly programmed. Artificial Neural Networks (ANN) is a model of machine learning which uses connected neurons, inspired by the axons in a biological brain, to solve complex problems such as image recognition, noise reduction and AI. [1] The network is able to solve these problems by finding patterns and underlying functions in large amounts of data with no prior knowledge or input on how to handle the data set.

One strategy to handle such problems is supervised learning. This works by feeding the network example inputs as well as the desired outcome for each input. [2] The network uses this training data to create an inferred function which it then can apply to unseen data. In this thesis we will apply supervised learning using a Feedforward Neural Network (FFNN) which is discussed in more detail in section two below.

We are using neural networks to predict the outcomes of an online multiplayer strategy game called Dota 2. Played in a 5 versus 5 format, every game starts with each player picking one out of 113 unique characters to play for the duration of the game. Every character has four to six unique spells, used to either deal damage, control enemies, heal allies, or provide support such as teleporting, etc. This results in a plethora of possible combinations of spells, and characters can have unique synergies in multitude of different ways.The countless character combinations cause every game to be different from the last and picking the right characters can ultimately decide whether you win or lose a game.

Dota 2 is not your average computer game. It has a massive player base and ever since the game was released in 2012 professional tournaments with large prize pools have been held. [3] As Dota players enjoyed watching the best in the world play, the competitive scene grew. Today there are dedicated teams with contracts and sponsors that compete all around the world. Notwithstanding the value of being able to predict the outcomes of such a highly advanced game, demand might also exist for a tool that analyzes game data in a predictive capacity.

In the complicated game of Dota it is impossible, even for the best players, to recognize all the possible patterns that can lead to victory. We aim to investigate how the computational power of a Neural Network instead can be used to predict the outcome of a game. We are solving a classification problem, identifying which sub-group new data belongs to by analyzing training data. In our case we are determining what team will win based on the picks of the game. The obvious problem is that there might be too many variables that determine game outcome, thus picks becoming too insignificant. Our goal is to explore if it even is possible to find a correlation between character selection and game outcome.

1.1 Problem Statement

This thesis aims to investigate if Artificial Neural Networks can be used to predict Dota 2 game outcomes. We intend to create a model that uses character selection data as input to predict the outcome of a game as either a win or a loss. Our thesis aims to explore the following:

- Can Artificial Neural Networks determine a correlation between character selection and game outcome of a Dota 2 game?
- How does predictive performance of the ANN vary when applying different ANN models?

1.2 Scope

A lot of variables could potentially affect the outcome of a Dota 2 match, such as a team's general level of skill or players' moods. As such our approach to purely use character selection to predict game outcomes will likely cause a decrease in classification accuracy. However, our intention with this thesis is not to maximise overall predictive power of our models, but rather explore the possibility of using ANNs to find correlations between character selection and game outcome in a conceivably imbalanced data set. It is rather within this subspace of ANN research that we will try to maximise predictive power.

Due to resource constraints such as computational power and time availability, we will be using a high level yet lightweight framework to implement the ANNs. This framework is called Fast Artificial Neural Network and will be outlined in further detail below. While it is possible that a more methodical and efficient way to find an optimal ANN model might exist, part of our goal is to compare the potential of various ANN models and the only way to achieve that is by training many ANN models.

1.3 Thesis Outline

This report is organized as follows: Section 2 aims to familiarize the reader with ANNs, algorithms and other relevant terms discussed in this thesis. In section 3 we motivate our choice of dataset, describe how we trained and tested our network as well as how the results were interpreted. The methodology described in section 3 is used to provide results that are presented in section 4. In section 5 the results and limitations are analyzed and discussed. In the last section we explore reasons for the obtained results as well as summarize and conclude the study.

2 Background

Below we give an outline of relevant definitions and terminology needed for further understanding of the report, as well as a background to Artificial Neural Networks and predict

2.1 Definitions and terminology

Artificial Neural Network (ANN) and ANN Models: A network of artificial neurons. The model describes the structure of the ANN.

Training data: A data set $S_{training} = \{(Inputs_1 ,\ Output_1\ ), \ldots, (Inputs_n ,\ Output_n\ )\}$ used to train the ANN. In this project the training data represents the “past games” constituted by $n = 92650$ games.

Testing data: A data set $S_{testing} = \{(Inputs_1 ,\ Output_1\ ), \ldots, (Inputs_m ,\ Output_m\ )\}$ of Input-Output data used to test the ANN. In this project, this set represents the “future games” constituted by $m = 10294$ games.

Inputs: A set of attributes, or a vector **x**. In this project every Input set will consist of exactly 113 attributes, where every attribute is a discrete value from {-1, 0, 1}. Each attribute in this set represents a character and the value means the following picking status: -1 = selected by first team, 1 = selected by second team, 0 = not selected for this game.

Output: Every Output is a singleton discrete value from {-1, 1}.

The ANN Output: A singleton output in the continuous domain [-1, 1]. The ANN Output is what the ANN returns in an attempt to predict/classify the expected output from the given set of inputs.

Epoch: One training iteration of the whole training data set.

Error: The error is the difference between the ANN output and the actual expected output.

MSE: Mean squared error.

2.2 Artificial Neural Networks

Inspired by the structure of the biological brain, the ANN is a network of artificial neurons. These neurons, or perceptrons, make up the building block of the ANN. Every neuron takes a number of input signals and produces an output signal. Depending on the connections between the neurons, these output signal are in turn used in the weighted input of other neurons. [1][4] While a single neuron cannot do much, a lot of neurons joined together with the help of the backpropagation algorithm, can carry out very complex computations.

2.2.1 Feedforward Neural Networks

By using feedforward neural networks (as opposed to recurrent neural networks) we can ensure that information always moves in one direction. The feedforward neural network can be described as an acyclic directed graph whose vertices correspond to neurons and whose edges correspond to the connections between them. [5]

Let the graph G = (V,E) and a weight function over the edges, w : E → R. Let σ : R → R be the activation function for each neuron. In our case we have picked σ to be the sigmoid function, that is σ(a) $= \frac{1}{1+e^{-a}}$, which is a smooth approximation of the step function, a commonly used activation function. [6] This choice means that every neuron will be able to output a continuous value in the range [-1, 1].

The network is organized in layers. In order to get an expression of the relationship between inputs and outputs, we let the layers be denoted V_0 , V_1, ..., V_n. We call the V_0 the input layer, V_1, ..., V_{T-1} the hidden layers and V_T the output layer. That is, V = $\bigcup_{t=0}^{T} V_t$ such that every edge in E connects some node in V_t to some node in V_{t+1} for some t ∈T. The bottom layer V_0 is called the input layer and it contains n+1 neurons where n is the dimensionality of the input space. In our case n = 113 and V_0 has 114 neurons.

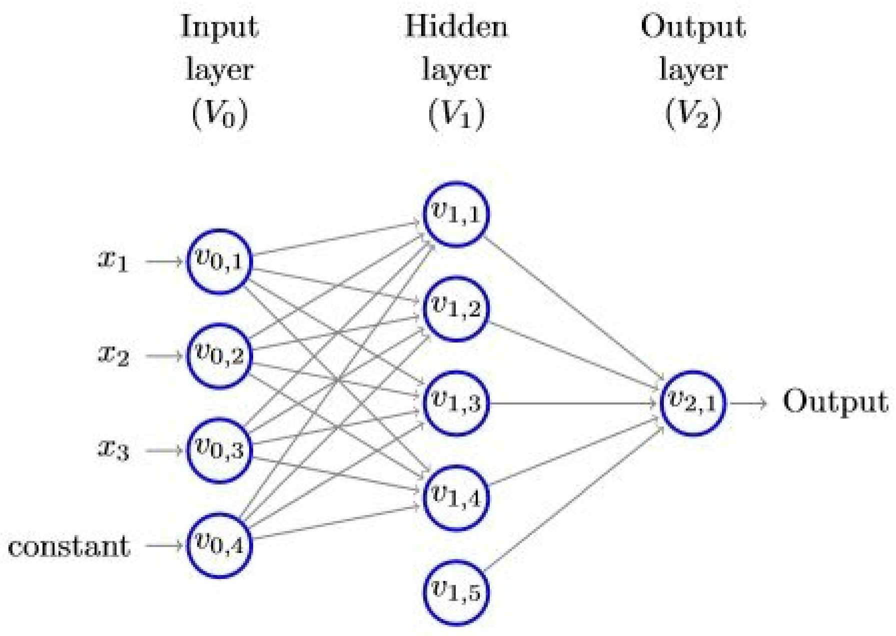
The 114th neuron is called a bias neuron and its output is always 1. For every i ∈ [n] the output of neuron i in V_0 is x_i. We denote the neurons $v_{t,i}$ to be the i:th neuron of the t:th layer and $o_{t,i}$ (**x**) to be its corresponding output when the network is fed with the input vector **x**. Therefore, for i ∈ [n] we have $o_{0,i}$ (**x**) = x_i, as well $o_{0,n+1}$ (**x**) = 1 (bias neuron). This provides us with a base case that allows us to proceed with the calculations layer by layer.

Suppose we have calculated the outputs of the neurons at layer t. Then the outputs of the neurons at layer t+1 can be calculated as follows: For any fixed $v_{t+1,j} \in V_{t+1}$, let $a_{t+1,j}$ (**x**) denote the input to $v_{t+1,j}$ when the ANN is fed with the input vector **x**. Then

$$a_{t+1,j}(\mathbf{x}) = \sum_{r:(v_{t,r}, v_{t+1,j}) \varepsilon E} w((v_{t,r}, v_{t+1,j}))\, o_{t,r}(\mathbf{x}) \text{ and } o_{t+1,j}(\mathbf{x}) = \sigma(a_{t+1,j}(\mathbf{x})).$$

Simplified, the input of the neuron $v_{t+1,j}$ is the weighted sum of the output neurons in V_t connected to $v_{t+1,j}$ where the weighting w is unique for each connection between the neurons, and the output of $v_{t+1,j}$ is the sigmoid activation function σ on its input. [7] [6]

Figure 1: A small simple ANN with a single hidden layer. It has 3 input attributes and 1 output.



2.2.2 The backpropagation algorithm

We have described how the information flows through the ANN and concluded that the output for any input vector **x** depends on the weights associated with the connections of the neurons. For the ANN to be useful it needs to be able to learn. It does this with a technique called backpropagation that allows it to quickly update the weights between the neurons in the ANN. [8] The backpropagation algorithm utilizes the gradient $\frac{\partial E}{\partial w}$ of the loss function E in respect to any weight w. [4] After each epoch, the weights are modified so as to minimize the mean-squared error between the neural network's prediction and the actual target value.

2.3 Predictive modelling

This section outlines relevant considerations and methodology for predictive modelling.

2.3.1 Classification

The goal is to utilize neural networks and supervised learning to predict the results of future games based on past games. This process can be described as a classification process and is outlined as follows: [9]

1. The ANN learns to classify past games based on the training set. This is the learning step.
2. We then determine the accuracy and train the ANN to be as accurate as possible at classifying the training data. The accuracy of the ANN is determined by the average error on the whole set.
3. The ANN is then used to classify new data from the testing set. The result is that the classification of the testing set essentially becomes a prediction of the future games based on the previous games. We can now determine how good the given ANN is at

predicting the future games and compare the efficiency of different ANN models for the task.

It is important to note that no ANN model will be able to perfectly predict the outcome of a game simply based on these attributes. This is because predicting the outcome of a game depends on many unknown factors besides character selection such as information about the players on either team.

2.3.2 Classification accuracy

There is a reason for splitting up the data into a training set and a testing set. Not doing so would allow us to draw false conclusions about the predictive accuracy of the ANN. This is because the ANN tends to overfit data. [10] For example, an ANN can achieve a very small error when predicting the training games yet perform much worse when attempting to predict the future games. The cause of this is inconsistencies in the general data; the ANN may learn to incorporate some particular anomalies in the training data that are not present in the overall data set. We expect it to be unavoidable to avoid the problem of overfitting for this thesis. Predicting a dota game outcome is a complex task containing many variables. Many of these variables are unknown and thus not accounted for in the dataset, therefore creating a seemingly random element that the ANN cannot predict.

3 Method

This section explains the method used in this thesis. Initially we discuss the dataset used in this thesis to both train and test the various neural networks. Next, we outline the framework for the networks and how it was trained and tested. Lastly we discuss how to assess their performance.

3.1 Dataset

The data set we use has been produced by players. For each input 10 human players have affected the outcome of the game. The more complex a game becomes, the number of variables that affect the game outcome increases and thus the relative significance of character selection should decrease. We predict that this will affect the ANNs' ability to find hidden trends and will likely cause bad performance. We still believe that the network will be able to foresee the outcome to some degree, only scaled.

The ANNs were trained with the help of a dataset containing 102,944 Dota 2 games coming from the UCI Machine Learning Repository. [12] The dataset was split into a training set containing 92,650 games and a testing set containing the remaining 10,294 games. Each game in the dataset is associated with an input vector and an output. Every input vector contains 113 attributes, one for every character that can be selected in the game. Each input attribute has a discrete value from {-1, 0, 1} that represents the selection of the corresponding character. -1 Means selected by the first team, 0 means not selected and 1 means selected by the second team. The output singleton for every game is a discrete value from {-1,1} and represents the winning team; -1 for first team and 1 for second team. The training data also inputs victory data, -1 or 1, depending on which team won the game.

3.2 ANN Models

Several different ANN models were trained and tested. Every one of these ANN models are Feed Forward Neural Networks utilizing epoch-based backpropagation. ANNs are covered more in depth in section 2.1. Every ANN has some particular structure of input neurons, hidden neurons and output neurons. In section 3.2.2 we describe the notation used to describe the ANNs in the result section.

3.2.1 Framework

The various FFNN:s were created, trained and tested using the Fast Artificial Neural Network Library (FANN) implemented in c++. It is lightweight and open source, which made it ideal for our tests. [11]

3.2.2 Parameters

We denote an ANN model in this report as $"a, b"$ where
a = number of hidden neurons in every hidden layer,
b = number of hidden layers,
and every model has 113 input neurons and 1 output neuron.

The sigmoid function is used as an activation function, this is because our inputs/outputs between [-1,1] and the sigmoid function is a smooth approximation to the step function with the range [-1,1].

3.2.3 Training

Every ANN was trained for 5000 epochs using the training data. After each step of 1000 epochs we noted the current Mean Squared Error for the training set and tested the ANN.

3.2.4 Testing

The test consisted of running the ANN against the testing set, noting the Mean Squared Error and amount of correct predictions.

3.3 Performance measures

Since the ANNs are trained to minimize the mean squared error one ought to assess its performance by looking at the mean squared error. The lower the error, the better the performance. However, there is an important distinction to be made between the error for the training data and for the testing data.

The error for the training data should only be used to interpret how well the ANN has learned the training data; simply because the ANN performs well against the training data, it does not mean that it has learned to solve the general prediction problem. [10] The real measurement of success should be the error of the testing data. [9] If an ANN has a low error against the training data but a high error against the testing data, this is an indicator of overfitting.

However, just looking at the mean squared error may not be very interesting either. For classification problems, it is common to interpret the ANN output to mean whichever class it is the closest to. In our case the ANN is attempting to predict the output to be one of two cases: -1 or 1. If the ANN output is a number above 0 then that is closer to 1 so we can interpret it as predicting 1. Similarly, if the output is less than 0 then that is closer to -1 so we interpret it as predicting -1. This allows us to count how many successful predictions it makes and to gain an intuition for likely it is to make a good prediction.

4 Results

In this section we will present the results produced in this thesis.

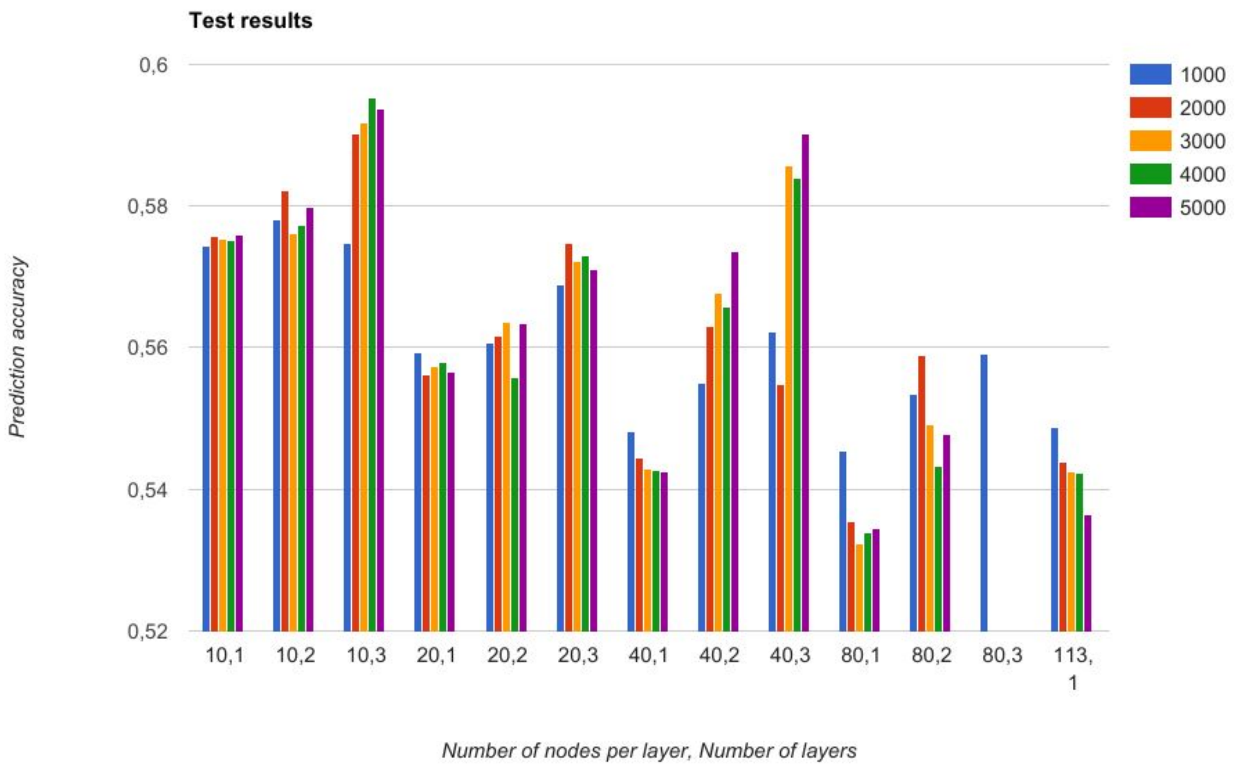
4.1 Accuracy tests

The different models are represented by two integers separated by a comma. The first integer represents the amount of nodes in a hidden layer and the second integer represents the number of hidden layers. For example: “20,2” means that the model uses 113 nodes in the input layer (this is always the case) followed by two layers of 20 nodes in each hidden layer, as well as 1 node in the output layer. Worth noting is that the model that uses 3 layers of 80 nodes has incomplete data due to a crash while training the network.

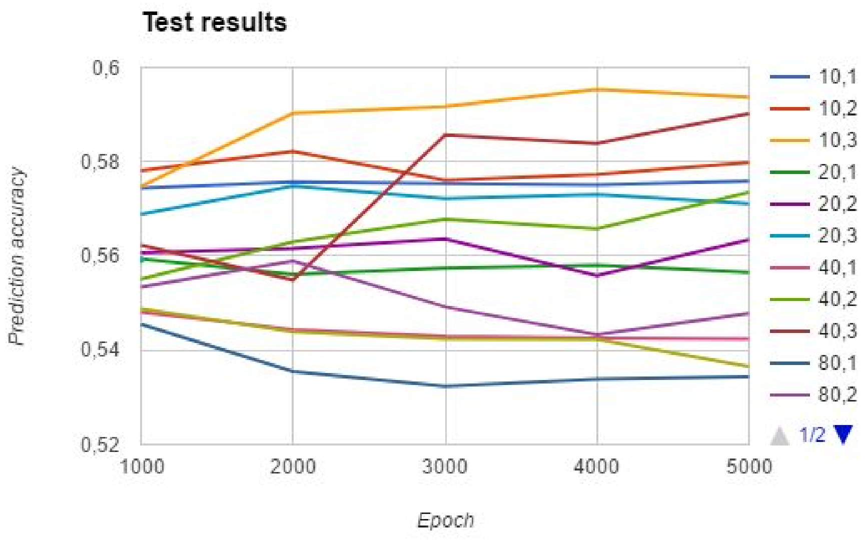
Table 1: Testing results after training for different models. This data is also represented in graphs 4.1 and 4.2. The values represent the ratio of correct predictions of the test data.

Epochs:	1000	2000	3000	4000	5000
10,1	0.5755	0.5757	0.5754	0.5751	0.5759
10,2	0.5781	0.5822	0.5761	0.5773	0.5798
10,3	0.5748	0.5903	0.5917	0.5954	0.5937
20,1	0,5594	0,5561	0,5574	0,5580	0,5565
20,2	0,5607	0,5616	0,5636	0,5558	0,5635
20,3	0,5689	0,5748	0,5722	0,5731	0,5711
40,1	0,5481	0,5444	0,5429	0,5426	0,5424
40,2	0,5551	0,5630	0,5678	0,5658	0,5735
40,3	0,5623	0,5549	0,5857	0,5839	0,5902
80,1	0,5455	0,5355	0,5324	0,5339	0,5344
80,2	0,5534	0,5589	0,5492	0,5433	0,5478
80,3	0,5592				
113,1	0,5488	0,5439	0,5424	0,5423	0,5365

Graph 1:: The accuracy of the different models. The different colors represent the amount of epochs used in the tests.



Graph 2: The same data as in graph 4.1 in a line format to more clearly visualize the differences between the models.



4.2 Performance comparison

The results show that all models have a prediction rate above 50%. The average accuracy for the models ranges between 53.44% and 59.54%, with the best and worst performing models being “10,3” and “80,1” respectively.

The accuracy seems to decrease as the number of nodes in each layer increase. The accuracy for 10,X is higher than 20,X and so forth. The exception being 40,3 showing better performance than 20,3. A similar observation is that the accuracy seems to improve as the number of layers in the models increase. This is true for all the models, regardless of how many nodes there are in each layer. Interesting to note is that the models with 40 nodes in each layer had a more significant improvement in performance when adding more layers than the models with 10 or 20 nodes. The amount of epochs used to train the networks seems to have little effect on the accuracy of the models. This is made clear by graph 4.3.

Table 4.2 shows a larger ANN model, the “113,1”. Note that for this model, the error in training becomes smaller with more epochs whereas the error in testing becomes larger.

Graph 3: Results for the different epochs.

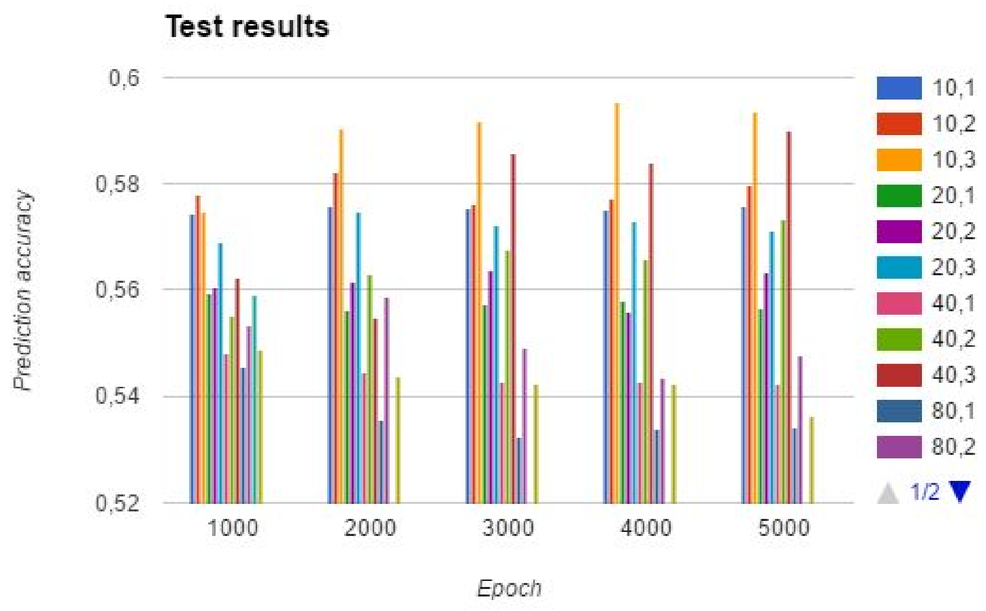


Table 2: Results of a larger ANN model, the “113,1” (that has 113 hidden neurons)

Epoch	Training error	Testing error	% predictions
1000	0.1391384155	0.323600	54.88%
2000	0.1131955758	0.360515	54.39%
3000	0.0989583582	0.378731	54.24%
4000	0.0908215046	0.386797	54.23%
5000	0.0855906084	0.393203	53.65%

5 Discussion

Below we analyze the results and their potential limitations.

5.1 Result analysis

As shown in the results, all the chosen ANN models have a better chance at predicting outcomes correctly rather than incorrectly, with the worst one predicting at a 53.44% accuracy. This is at least an indication that there may indeed exist a correlation between character selection and game outcome. If there was no correlation, the expected mean accuracy among the models would be 50%, however the mean accuracy among the fully trained ANNs in this report was is 56.38%. Even the most simplistic chosen ANN model, “10,1”, with only a single 10 neuron hidden layer had an accuracy at 57.60% after 5000 training epochs. While this hints at our predictive models being non-random, the degree to which our predictions deviate from 50% is low and we therefore urge caution when when analysing our models. We can thus not prove or disprove our hypothesis, but the results show some data in favor of it.

Another result to note is that none of the chosen models seem to be able to predict at any higher accuracy than 59.54%. This was well expected as the only attributes accounted for are the characters selected. For example, if there existed any model that could predict at 100% then it would mean that every game would be predetermined after character selection and there would be no point for the players to keep playing the game. Considering all the unaccounted factors, 59.54% is a strong indication of predictive power.

As noted, the simpler models such as the “10,1” model performed better in the prediction test than many of the more complex ANN models despite not showing any real progress against the training data past the first 1000 epochs. In fact, it seemed quite random as for which model would do well in the prediction test, apart from the very large neural nets (relatively speaking) which performed the worst.

The randomness regarding the ANN models improvement or lack thereof as epochs progressed past 1000 has two possible explanations:

1. The ANN already learned to the best of its potential by epoch 1000
2. The ANN needed many more epochs to learn the problem

We suspect explanation 1 is more reasonable after looking at the training error seeing as the smaller ANN models stagnated/shifted randomly while only the bigger ANN models improved in regards to the training data. What might come as a surprise though is that the best result was achieved by the “10,3” model, one of the ANN models with quite few hidden neurons.

Meanwhile, the trend for the more complex ANN models is that even though they were more proficient at learning the training data they just performed all the worse in the prediction test. This is a clear sign of overfitting the data, even at only 113 hidden neurons.

Considering these results in combination, we are tempted to infer that not all selectable characters affect the games equally and there may be a few possible explanations. One might be that some characters have neither a positive nor a negative effect for the winning chances of the team and result in unnecessary input data. Another explanation is that not all characters are equally popular, therefore not picked as often resulting in imbalanced data.

5.2 Limitations

The data set used for this thesis is the most limiting factor in our report. We only use a single data set without knowing if the data is unbalanced. The data was collected during a single day, and we have no way of knowing if the data we use is representative of an average Dota game.

6 Conclusion

This thesis has explored the following:

- Can Artificial Neural Networks determine a correlation between character selection and game outcome of a Dota 2 game?
- How does predictive performance of the ANN vary when applying different ANN models?

With regards to character selection and game outcome, our data seem to indicate that game outcome has some degree of dependence on character selection. However, the degree to which accuracy of our models deviates from pure randomness at 50% is low. Urging caution and an element of conservatism when analysing our results, we are therefore unable to prove our hypothesis that character selection affects the outcome of the game.

For many predictive models there will be omitted variables that limit the ability to accurately forecast future outcomes. The ANN models utilized in this paper are no exception. The limiting factor here was not computational power, but the seemingly random outcome of Dota games when only considering character selection as input. Choosing an appropriate ANN model with the right amount of hidden neurons is of outmost importance when attempting to predict any outcome. What we can conclude from this prediction problem is that not many hidden neurons are needed and that it does not take much for the ANN to start overfitting the data.

As a last word of thought, we believe that we have shown a clear indication that it is possible to quantify Dota 2 game outcomes. However, while this paper might offer a valuable glimpse into the predictive power of ANN networks, we believe that this is a lesson to be learned, not by means of computer science, but through playing the game in itself.

References

[1] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. Artificial neural networks: a tutorial. Computer, 29(3):31–44, Mar 1996. ISSN 0018-9162. doi: 10.1109/2.485891.

[2] Carlos Gershenson. Artificial neural networks for beginners. arXiv preprint cs/0308031, 2003.

[3] Dota 2 gamepedia, Competitive Scene (2017) http://dota2.gamepedia.com/Competitive_scene (accessed 2017-04-16)

[4] D.K. Montana and L. Davis. Training Feedforward Neural Networks Using Genetic Algorithms. BBN Systems and Technologies Corp. *url: https://www.ijcai.org/Proceedings/89-1/Papers/122.pdf*

[5] S. Shalev-Shwartz, S Ben-David. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014.

[6] W. Duch, N Jankowski. Survey of Neural Transfer Functions. Nicholas Copernicus University, 1999.

[7] D. Shiffman. The Nature of Code. Free Software Foundation. http://natureofcode.com/book/chapter-10-neural-networks/ (accessed 2017-04-11)

[8] M. A. Nielsen. Neural Networks and Deep Learning. Determination Press, 2015.

[9] J Han, M Kamber, J Pei. Data Mining Concepts and Techniques. Elsevier, 2012.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014) 1929-1958

[11] Fast Artificial Neural Network Library, FANN. *url: leenissen.dk*

[12] S. Tridgell. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA, 2016. https://archive.ics.uci.edu/ml/datasets/Dota2+Games+Results (accessed 2017-04-10)

[13] G. Blom et al. Sannolikhetsteori och statistikteori med tillämpningar. Studentlitteratur AB, 2004.