Predicting Global Video Game Sales Using Multilayer Perceptrons and Multiple Linear Regression

Marc Marsden (1437889)
School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
1437889@students.wits.ac.za

***Abstract*—This report aims to develop two models which can predict the global sales of video games (in millions) based on the year of release, the company which developed the video game, the publishing company, the genre, and the platform on which the video game is played. The first resulting model was a multilayer perceptron with 2 hidden layers with 300 nodes each, and an output layer with one node, which was implemented with a learning rate of 0.001, 100 epochs, and tanh as the activation function. The second model was a multiple linear regression model with a learning rate of 0.3 trained for 5000 iterations. The final results of the models were not very accurate due to overfitting, however, they did provide insight into the relationship between the features and the global sales of video games.**

***Index Terms*—multilayer, perceptron, linear, regression, game, sales**

I. Introduction

In an age where the video game market is saturated with various gaming companies, consoles, and genres, no one truly knows whether a game will sell well until it is released. This report presents two models which predict the global video game sales based on a variety of features related to the video game itself. With the ability to predict whether a video game's sales are high, creators or investors can decide whether to continue with the video game's development.

The two models presented in this report are the multilayer perceptron and the multiple linear regression. The following section will present background information on the two models as well as their key components. Section III describes the methods used to extract and preprocess the data as well as selecting the optimal variation of each model. Section IV will discuss and analyse the results of the models. The last section presents a summary of the main points of the report along with possible future work.

II. Background

A. Perceptron

A perceptron is a linear discriminant model which corresponds to a binary classification model. The input vector $\mathbf{x}$ is first passed through a non-linear transformation function to produce a feature vector $\phi(\mathbf{x})$ and then this is used to create a generalised linear model

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x}))$$

where f is a non-linear activation function given by

$$f(a) = \begin{cases} +1 & \text{if } a \geq 0 \\ -1 & \text{if } a < 0 \end{cases}$$

and $\mathbf{w}$ is the weight vector. $\phi(\mathbf{x})$ includes a component $\phi_0(\mathbf{x}) = 1$ known as the bias [1] which allows for the decision boundary to be shifted away from the origin. The goal is then to find the optimal weights which produce a small error based on the specific cost function used. However, this is only useful if the data is linearly separable.

A more advanced form of the perceptron, known as the multilayer perceptron, was developed to overcome the shortcomings of the single perceptron such as the requirement that the data be linearly separable. This model consists of multiple layers of computational units (neurons), which have direct connections to the neurons of the subsequent layer. Contrary to the name, the perceptrons which make up the network are not true perceptrons in the sense that where true perceptrons perform binary classification a neuron in a multilayer perceptron may perform either a classification or a regression based on the activation function chosen [2].

B. Activation Function

One of the most important components of a multilayer perceptron is the activation function [3]. The purpose of an activation function is to decide which neuron should be activated by calculating a value based on the weighted sum of input plus a bias. This can be linear or non-linear; however, the use of a non-linear activation function allows for more capable learning of complex tasks [4]. The only layer which may use a linear activation function is the output layer depending on the problem.

The three most used activation functions are Sigmoid function, Hyperbolic tangent (tanh) function, and Rectified Linear Unit (ReLU) function. The Sigmoid function is a bounded differentiable real function, which takes in real input values,

with non-negative derivatives across the entire domain [5]. This function is defined as

$$f(x) = \frac{1}{1 + e^{-x}}$$

and is mainly used for probability-based output and in shallow networks. A drawback to the implementation of the Sigmoid function is that the gradient vanishes (tends to zero) as x grows infinitely large. In addition to the vanishing gradient, the Sigmoid function also produces a zero-centred output which causes the gradient to propagate in different directions during backpropagation [5].

The tanh function is given by

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

which can be simplified to

$$f(x) = 2 \cdot \sigma(2x) - 1$$

where $\sigma(x)$ is the Sigmoid function. The tanh function is preferred over the Sigmoid function since it giving better training results for multilayer percetrons. This is due to the tanh function producing zero centred output which aids the backpropagation process [5]. However, similar to the Sigmoid function, tanh also suffers from the vanishing gradient problem.

The ReLU function receives the input x and produces the maximum of either x or zero. This can be shown mathematically as follows:

$$f(x) = max(0, x)$$

This replaces any input less than zero, which eliminates the vanishing gradient problem seen in the Sigmoid function and tanh function [5]. The main advantage of this activation function is that no exponentials or divisions are required in the calculation, which implies faster computation. However, the ReLU function has a high risk of overfitting the training data when compared to other activation functions such as the Sigmoid function [5].

C. Linear Regression

Simple linear regression models a linear relationship between a scalar response variable and an explanatory variable. In the case of more than one explanatory variable, this process is known as multiple linear regression, mathematically defined as

$$Y = X\beta + \epsilon$$

where Y is the response variable with size $n \times 1$, X is a $n \times p$ matrix containing the explanatory variables, β is a $p \times 1$ vector containing the weights for each explanatory variable, and ϵ is the $n \times 1$ vector known as the error term [6]. The goal of multiple linear regression, in machine learning, is to determine the optimal weight vector β, which minimises the error between the predicted value and the actual value. To achieve this, gradient descent is implemented to minimise the error.

D. Gradient Descent

Gradient descent is an iterative optimisation algorithm used to minimise some function by stepping in the direction of steepest descent, defined by the negative of the gradient, with the size of the steps depending on the learning rate α. The parameter of the function is updated by adding the product of the negative of the gradient and learning rate to the parameter itself. [7]. In machine learning, the function to minimise is the cost function defined and the parameter to be updated is the weights of the particular model. Below is an example of the machine learning updating scheme

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{N} \sum_{i=1}^{N} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

where θ_j is the weight of the model, α is the learning rate, $h_\theta(x)$ is the predicted value of the, and y is the actual value.

III. Methodology

A. Data

The dataset used for this report was sourced from Kaggle. There are 55,792 records in the dataset which contains a list of video games with sales, critic scores, and information about the video games such as which company published and developed it. This was generated by scraping the data from *vgchartz.com* as of 12 April 2019 [8]. The features extracted are the year of release, the company which developed the video game, the publishing company, the genre, and the platform on which the video game is played. Along with these features, the global sales (in millions) for each game will be extracted as the target value.

A problem that occurs when choosing these features is that the majority are categorical. To solve this issue, we implemented *One Hot Encoding* to represent the data with columns of ones and zeros. The year of release attribute was normalised using *Min-Max Normalisation* so that the attribute did not dominate the calculations. With all records containing null values removed and *One Hot Encoding* implemented, the dataset contained 18,030 rows × 4,302 columns. This was split into training, validation, and testing data in the ratio 57:10:33, respectively.

B. Multilayer Perceptron

The chosen design for the multilayer perceptron was an input layer the size of the number of features (4,302), two hidden layers, and an output layer of size 1. The complication with the hidden layer was determining how many nodes should the layers contain. Due to the amount of time it takes to train a multilayer perceptron, we chose two different values for the size of the hidden layers, 300 nodes and 500 nodes, with a learning rate of 0.001 and the hyperbolic tan function as the activation function. These two hyperparameters were trained and then tested using the validation with 100 epochs. The best one was chosen for the final model. The results are described in the Analysis of Results section.

C. Multiple Linear Regression

The simplicity of the multiple linear regression allows for fast training relative to the multilayer perceptron. This allows for the testing for the optimal learning rate. To achieve this, we ran the algorithm on the training data using various learning rates, for 10 iterations, and plotted the costs for each iteration. The best learning rate was chosen based on this visualisation. We then retrained the model using the chosen learning rate for 5000 iterations. The results are discussed in the following section.

IV. Analysis of Results

It was previously stated that we needed to test for the optimal hyperparameters for the multilayer perceptron. After training the variations, the validation data was passed to each variation of the multilayer perceptron and their Mean Square Error (MSE) scores were calculated. The smaller model (300 nodes) had an MSE score of 1.0057, while the larger model (500 nodes) had an MSE score of 2.1792. Since we want the error to be as small as possible, we selected the 300 hidden node model. The first 10 predictions for the multilayer perceptron can be found in Table II

With the hyperparameters selected, the model was passed the testing data and scored an MSE of 1.0655. Comparing this to an MSE score of 0.5297 on the training data, the model may be overfitting the training data. Looking at a different metric for confirmation, we look at the coefficient of determination regression (R2) score. The training R2 score of 0.2603 is substantially better than the testing R2 score of -0.3761. This reinforces the idea of overfitting. The first 10 predictions for the multiple linear regression can be found in Table III.

After analysing the plotted errors for each iteration of multiple linear regression, the decided learning rate was 0.3. This model, with 5000 iterations, produced an MSE score of 0.8306 for testing data and 0.5052. Similar to the multilayer perceptron, it seems as though the model has overfit the training data as the R2 score for the testing data is -0.0727. This value is extremely close to zero, which indicates an almost constant model that always predicts the expected value of y, disregarding the input features. A summary of these results can be found in Table I.

When comparing the scores of the multilayer perceptron and the multiple linear regression, the immediate choice would be the multiple linear regression model due to its smaller MSE score for testing data. Even though the accuracy of both models are relatively inaccurate, they both give an interesting look into the relationship between the features used and the global sales of video games.

V. Conclusion

The purpose of this research was to develop a multilayer perceptron and a multiple linear regression model which could predict the global sales of a video game based on information about that video game. The features chosen for this report were the year the video game was released, the company who developed the video game, the publishing company, the genre and the platform on which the video game is played. After tuning the hyperparameters, a multilayer perceptron with 2 hidden layers with 300 nodes each, and an output layer with one node, which was implemented with a learning rate of 0.001, 100 epochs, and tanh as the activation function. The best learning rate for the multiple linear regression model was 0.3, based on the plotting of the errors for each iteration of the various models. The final result was two models which may have overfitted the training data but still provided some insight into the relationship between the specific features and the global sales.

In the future these models may be improved on by creating more complex versions and using different features which may relate more to the global sales than the ones used in this report.

TABLE I
Scores of Multilayer Perceptron and Multiple Linear Regression

	Multilayer Perceptron	Multiple Linear Regression
MSE Score Training	0.529752	0.505263
MSE Score Testing	1.065595	0.830668
R2 Score Training	0.260273	0.294469
R2 Score Testing	-0.376098	-0.072715

VI. Appendix

TABLE II
Results of First 10 Predictions of Multilayer Perceptron

Predicted	Target	Error
0.823493	0.02	0.803493
0.658469	0.19	0.468469
0.583934	0.18	0.403934
1.522947	1.41	0.112947
0.296917	0.62	-0.323083
0.407635	0.40	0.007635
0.712798	0.01	0.702798
1.167628	0.90	0.267628
0.252646	0.32	-0.067354
0.669394	0.03	0.639394

TABLE III
Results of First 10 Predictions of Multiple Linear Regression

Predicted	Target	Error
0.002553	0.02	-0.017447
0.677047	0.19	0.487047
0.905527	0.18	0.725527
1.405005	1.41	-0.004995
1.045534	0.62	0.425534
0.326285	0.40	-0.073715
0.836126	0.01	0.826126
1.653436	0.90	0.753436
0.110906	0.32	-0.209094
0.014057	0.03	-0.015943

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] K. C. Dunuku and V. Saritha, "Reducing error signal in multilayer perceptron neural networks using mlp for label ranking," *Internal Journal For Research In Applied Science and Engineering Technology*, 2013.
[3] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized mlp architectures of neural networks," *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2011.
[4] A. S. V, "Understanding activation functions in neural networks," 2017, https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0 (Last accessed: 28/06/2020).
[5] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.
[6] D. A. Freedman, *Statistical models: theory and practice*. cambridge university press, 2009.
[7] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261, 1944.
[8] A. Alqunber, "Video games sales 2019," 2019, https://www.kaggle.com/ashaheedq/video-games-sales-2019 Last accessed: 28/06/2020).