Predicting Video Game Rating from Genres using Multilayer Perceptron

15th March 2021

1 Introduction

Playing video games can be a way to relax and experience diverse emotions through player–game interaction, a brief respite from reality. However, deciding what video games to play in an age of endless technological distractions and social fear of missing out (*FOMO*) [1] can be an arduous task. Will a game be worth putting hours into, if one could be doing something else? One way, albeit a superficial one, to assess whether a game is worthwhile is to check reviews. These generally include a numeric score of some type but, unfortunately, not all games are blessed with abundant review data.

Steam is a game distribution service maintained by Valve Corporation [2]. The service holds arguably the biggest market share of PC gaming services with 120 million monthly active users in 2020 [3]. Steam allows users to add text-formatted tags to describe a game and shows the most popular tags publicly. In addition, the service shows a numeric user score between 0–100% for a given game. This represents the percentage of positive reviews given by the community.

In this project, we present a multilayer-perceptron-based machine learning method to predict the numeric user score of a game using the genres the game is classified as. While predicting video game score from esoteric features is not new [4], using this method, one can assess whether new games are—in a sense—good or not by examining the genres the game represents.

2 Problem Formulation

We use games available on the Steam store as data points. The employed features are the user-classified tags representing genres and the label to predict is the numeric user score expressing the ratio of positive and negative reviews given by the community. Therefore, the features are the multiple categories, genres, a game can belong to.

The data provided by Steam is public but not easily exportable. Therefore, light data mining is necessary. Fortunately, this has already been done by Weinbaum [5], providing a Google Sheets formatted spreadsheet containing the tags, user score, and revenue of all games on Steam up to early 2020. Using all of the data is not wise, as Steam includes many rarely played games—outliers. As a consequence, we only consider games that have a projected revenue of at least \$1000. This leaves us with 8868 data points.

3 Method

One-hot encoding is used to encode the string-formatted tags as a numeric matrix of ones and zeroes. This is done because the tags are categorical variables, which the employed machine learning (ML) models cannot handle directly. In this encoding, the unique categories, genres, a game can have are each assigned an index. Consequently, the feature vector of a data point consists of ones and zeros – ones representing that a game belongs to the category represented by the index [6].

Numerical regression from categorical one-hot-encoded data is a difficult problem we believe to be inherently non-linear in this case. Therefore, we exclusively implement a multilayer perceptron (MLP) neural network (NN) consisting of rectified linear activation $f(x) = \max(0,x)$ (ReLU) nodes and a single linear $f(x) = x$ node in the output layer. Layers of ReLU can approximate any non-linear continuous function [7]. For this reason, we train the ReLU to effectively discover the non-linear logic behind the data. Thus, we do not require other hypothesis spaces. Although, more data is required in practice for attaining an ideal MLP. Subsequently, we employ a single linear node to output a numeric variable, the user score. The implementation is done in Python using the TensorFlow and Keras libraries and the specifics of the model are presented in Section 3.1. The NN is trained for 800 epochs, i.e., the weights $\mathbf{w}$ are improved for 800 iterations.

Our loss function of choice is Huber loss, as it is less sensitive to outliers by including a penalty term for residuals larger than a set threshold δ. This loss is defined in Eq. (1).

$$\mathcal{L}(y,f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \le \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{1}$$

Additionally, the TensorFlow implementation of the Adam algorithm is employed for optimisation or learning. It is a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments [8].

We employ a split of 20% to the data, providing training and test data using `train_test_split` from the scikit-learn library [9]. Secondly, we employ 5-fold cross validation of the training data to attain rolling learning and validation data. The mean losses from this cross validation are used for error analysis. This narrows the learning data points of a cross-validated model to 5675 from 8868.

Hyperparameter tuning is used to find optimal neural network composition and Huber loss delta δ. We use Keras Tuner [10] for this purpose and visualise the results as a parallel coordinates figure with HiPlot [11] and `keras-tuner-hiplot` [12]. The objective of the tuner is to minimise the loss of the validation data.

3.1 Neural network model

```python
import tensorflow as tf
import keras
def create_model():
    with tf.device('/GPU:0'): # Force GPU usage
        model = keras.Sequential() # Define stack of NN layers with one input-output tensor
        model.add(Dense(4, input_dim=X_train.shape[1], activation='relu')) # First ReLU-layer
        model.add(Dense(80, activation='relu')) # Sweeped as neurons
        model.add(Dense(20, activation='relu')) # Sweeped as neurons2
        model.add(Dense(1, activation='linear')) # Linear output layer

        model.compile(loss=keras.losses.Huber(delta = 3.0), # Huber loss with delta of 3.0
                      optimizer=keras.optimizers.Adam(), # Adam algorithm for learning
                      metrics=['MeanSquaredError']) # Output also MSE
        return model
```

4 Results

Two layers of the model, `neurons` and `neurons2`, were tuned for sizes between 10–90 and 10–50 with a step size of 10, respectively. In addition, the Huber loss delta δ was tuned between 0.5–4. The results from this tuning are shown as parallel coordinates in Fig. 1. The resulting optimal parameters were `neurons` = 80, `neurons2` = 20, and Huber $\delta = 3.0$. After tuning the hyperparameters, the total amount of parameters in the model was 3909, which is graciously smaller than the number of training data points (5675). The amount of unique categories, genres, was 377.

The 5-fold cross-validated training and validation error of the model, defined as the Huber loss, are presented in Fig. 2(a) in addition to the test error. Accordingly, the validation error is larger than the training error yet less than twice as substantial. The same applies to the test error, which ultimately is the most trustworthy metric, as it was not used in any part of the learning process. This indicates relatively decent performance.

An intuitive way to test whether the model is realistic is to check the distribution of user scores predicted from the test data. This is presented in Fig. 2(b), from which we can see a realistic majority of scores around 80, a rough drop to the larger scores, and a softer drop to smaller scores. Nevertheless, some outliers are seen as scores above 100. The true distribution of the validation data is apparently more skewed to the right. In contrast, the outline of the predicted histogram corresponds better to the distribution of critic scores on Metacritic [13]—a site providing weighted averages of video game review scores [14].

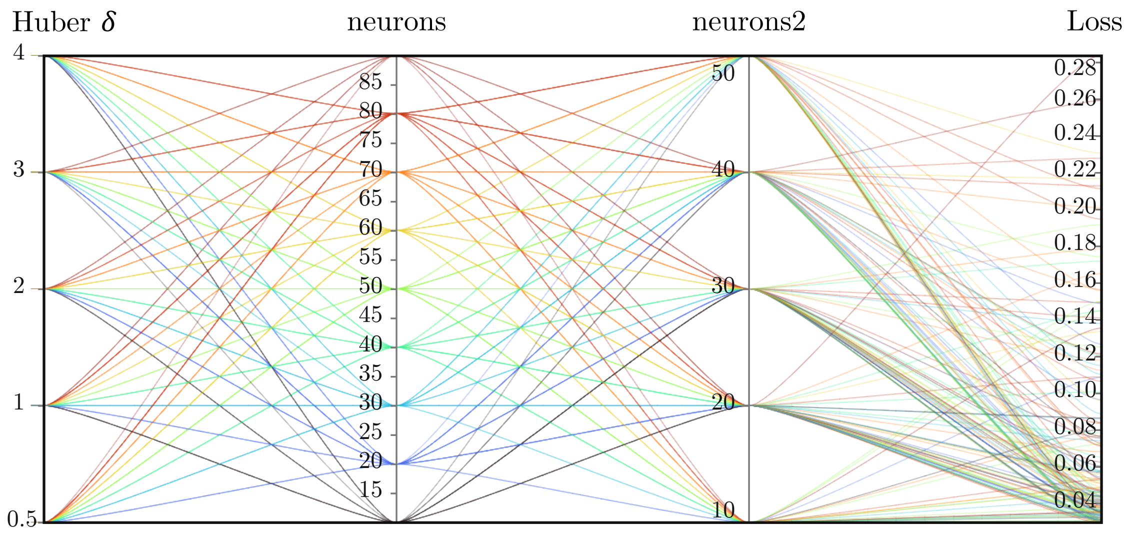

Figure 1: Parallel coordinates plot of model hyperparameter tuning generated using HiPlot and `keras-tuner-hiplot`. In this case, lower loss is better.

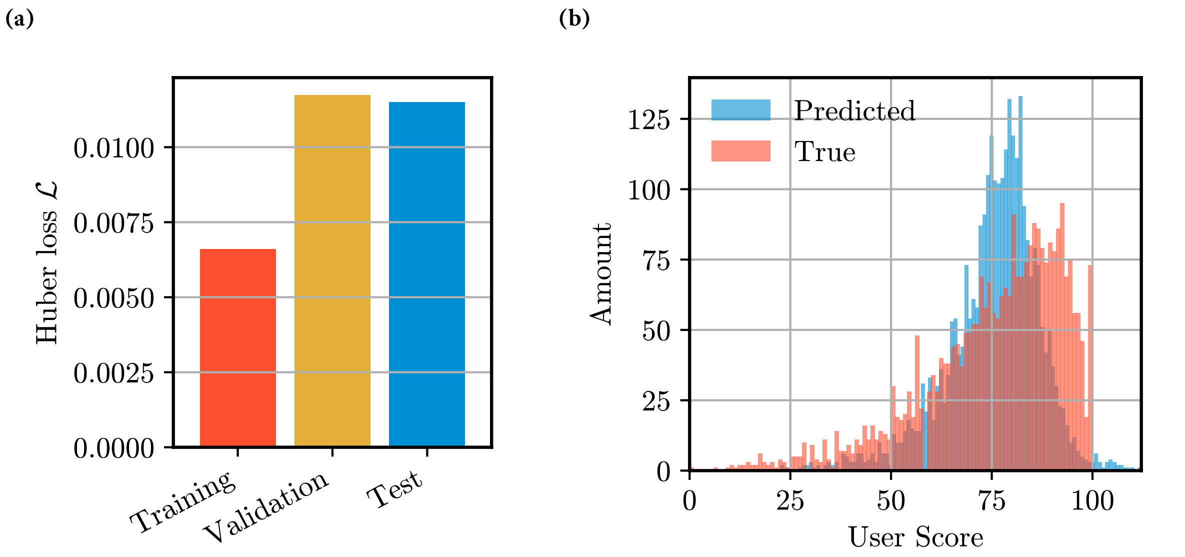

Figure 2: **(a)** 5-fold cross validated training, validation, and test error computed from Huber loss. **(b)** Histogram representing the distribution of user scores predicted from the test data and the true distribution.

For curiosity, the model can be used to predict the *best* game on Steam, which we find to be *Changed* by the developer DragonSnow [15] with the tags: *2D, Action, Adventure, Anime, Dating Sim, Difficult, Great Soundtrack, Horror, Indie, Pixel Graphics, Psychological Horror, Puzzle, RPG, RPGMaker, Sexual Content, Singleplayer, Story Rich, Survival Horror, and Cute*. The game was predicted to have a user score of 119.77333% while the real score was 89%.

As the true test data in Fig. 2(b) suggests, the model could be improved. One possible avenue for improvement is to use a non-linear output layer instead of the linear currently in use. Limiting the possible scores to the range 0–100 is to be considered as well.

5 Conclusion

We examined the possibility of predicting the numeric rating a video game receives from the community using solely the genres describing a game as features. We used publicly available data mined from Steam for this purpose and trained a MLP consisting of mainly ReLU layers. Fascinatingly, the model predicted the *best* game to be *Changed*, a difficult action puzzle game featuring anthropomorphic animal characters.

The distributions of user scores shown in Fig. 2(b) did not overlap satisfactorily. Therefore, there are clearly improvements to be made. For example, the NN model can possibly be designed to perform better by not using a linear output layer for attaining the numeric variable. Furthermore, using data from Metacritic as scores instead of Steam user scores could yield a more interesting model, as the current user scores are only the percentage of positive reviews. Contrary to Steam, Metacritic compiles well-thought-out critic scores.

References

[1] G. Harrison and M. Lucassen, *Stress and anxiety in the digital age: The dark side of technology*, 2019. [Online]. Available: https://www.open.edu/openlearn/health-sports-psychology/mental-health/managing-stress-and-anxiety-the-digital-age-the-dark-side-technology (Accessed: 3 February 2021).

[2] Valve Corporation, *Steam*, Bellevue, WA, 2021. [Online]. Available: https://store.steampowered.com/ (Accessed: 3 February 2021).

[3] R. Stanton, *Steam had 120 million monthly users in 2020*, 2021. [Online]. Available: https://www.pcgamer.com/steam-had-120-million-monthly-users-in-2020/ (Accessed: 3 February 2021).

[4] V. Batchu and V. Battu, "“How to Rate a Video Game?” — A Prediction System for Video Games Based on Multimodal Information," in *Frontiers in Pattern Recognition and Artificial Intelligence*, World Scientific, Jun. 2019, pp. 265–280. DOI: 10.1142/9789811203527_0014.

[5] D. Weinbaum, *Genre Viability on Steam and Other Trends - An Analysis Using Review Count*, November 2019. [Online]. Available: https://www.gamasutra.com/blogs/DannyWeinbaum/20191115/353349/Genre%7B%5C_%7DViability%7B%5C_%7Don%7B%5C_%7DSteam%7B%5C_%7Dand%7B%5C_%7DOther%7B%5C_%7DTrends%7B%5C_%7D%7B%5C_%7DAn%7B%5C_%7DAnalysis%7B%5C_%7DUsing%7B%5C_%7DReview%7B%5C_%7DCount.php (Accessed: 3 February 2021).

[6] J. Brownlee, *Why One-Hot Encode Data in Machine Learning?* Jun. 2020. [Online]. Available: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/ (Accessed: 4 February 2021).

[7] K. I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, no. 3, pp. 183–192, January 1989, ISSN: 08936080. DOI: 10.1016/0893-6080(89)90003-8.

[8] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, December 2014. arXiv: 1412.6980.

[9] Scikit-learn developers, *Sklearn.model_selection.train_test_split*, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model%7B%5C_%7Dselection.train%7B%5C_%7Dtest%7B%5C_%7Dsplit.html (Accessed: 5 February 2021).

[10] T. O'Malley, E. Bursztein, J. Long *et al.*, *Keras Tuner*, 2019. [Online]. Available: https://github.com/keras-team/keras-tuner (Accessed: 5 February 2021).

[11] D. Haziza, J. Rapin and G. Synnaeve, *Hiplot, interactive high-dimensionality plots*, 2020. [Online]. Available: https://github.com/facebookresearch/hiplot (Accessed: 4 February 2021).

[12] V. Schettino, *keras-tuner-hiplot, HiPlot visualization data for a Keras Tuner hyperparameter optimization project*, 2020. [Online]. Available: https://github.com/vbschettino/keras-tuner-hiplot (Accessed: 4 February 2021).

[13] E. Oulster, *Average Is Not the Middle: Ratings & Their Distributions*, 2018. [Online]. Available: http://www.ericoulster.com/2018/08/10/average-is-not-the-middle-ratings-their-distributions/ (Accessed: 6 February 2021).

[14] Metacritic, *How We Create the Metascore Magic*, 2021. [Online]. Available: https://www.metacritic.com/about-metascores (Accessed: 6 February 2021).

[15] DragonSnow, *Changed*, 2018. [Online]. Available: https://store.steampowered.com/app/814540/Changed/ (Accessed: 7 February 2021).