Roro Duran
Assignment 2
LaHaye - CPSC 393

ABSTRACT

Neural Networks, a versatile machine learning architecture, excel at unraveling intricate data patterns and predicting outcomes. They begin with the introduction of x-input data into the input layer, allowing users to configure the number of intermediary layers between the input and output layers. Each layer possesses unique attributes such as dimension size and activation functions, enabling customization to align with specific modeling goals. At the end, the output layer generates predictive results, forming one component of the loss function alongside the actual 'y' value. User-defined nodes in each layer, particularly beyond the input layer, include parameters like weights and biases, enhancing data understanding. Backpropagation, an iterative learning process, propels neural networks by forwarding data sequentially from the input layer to the output layer, scoring with the loss function. It then reverses, calculating derivatives of the loss function with respect to weights and biases, refining these parameters for optimal performance. This iterative cycle continues until convergence, marking the network's attainment of its maximum predictive potential. Neural networks prioritize parameter optimization for enhanced performance, while also encouraging user experimentation with layer count, dimensions, and activation functions to achieve optimal outcomes across diverse applications.

INTRODUCTION

This document outlines the steps taken to build three distinct models, each applied to different datasets, following methodologies in line with those discussed in the fourth chapter of "Deep Learning with Python, 2nd Edition." The initial model is dedicated to predicting whether women's e-commerce clothing items are recommended (labeled as 1) or not (labeled as 0) using customer reviews, which are tokenized into word-based integer representations. The second model extends this concept to categorize reviews on a five-class scale, sourced from Trip Advisor reviews with textual descriptions and aims to predict the numeric rating of those reviews. Lastly, the third model adopts a regression approach, predicting apartment prices in Poland by utilizing numeric features like distances to points of interest, room count, and floor level, with predictions denominated in Polish Zloty. These three examples showcase the versatility of deep learning techniques across various datasets and tasks, offering a comprehensive preview of the report's contents.

DISCUSSION

The process commences with the importation of the requisite libraries, ensuring seamless access to the necessary functions. In all instances, the dataset is retrieved using Pandas' built-in function, enabling the effortless acquisition of datasets via valid URLs. Null values are promptly eliminated from the datasets to ensure the uniformity of samples. Subsequently, the procedures detailed in the textbook examples are meticulously followed. X and Y variables are defined based on the dataset. In the context of binary classification, X represents the user-generated reviews for specific items, while Y pertains to the binary classification label 'Recommended IND.' Notably, all X-values are converted to strings to maintain consistency.

Next, the textual reviews undergo tokenization, assigning unique integers to each word. Following this, a conventional train-test split allocates 70% of the data for training and reserves the remainder for testing. All training and testing variables were vectorized using a textbook-provided function in conjunction with the model. A straightforward model is constructed, comprising three layers with dimensions of 16, 16, and 1, each employing activation functions (relu, relu, and sigmoid, respectively). It's essential to utilize a sigmoid activation function for the output layer in binary classification to ensure the output's binary nature (1 or 0). The model is compiled with the rmsprop optimizer, binary_crossentropy as the loss function, and accuracy as the metric. Subsequently, a validation set is created from the training data, involving 10,000 values. The model is then trained on this validation set, spanning 200 epochs with a batch size of 512. A higher number of epochs offers a more comprehensive view of the model's performance across iterations. The final step with the validation set entails plotting loss and accuracy values against epochs, determining the optimal number of epochs for the testing set, which, in this case, was found to be 38.

The same model, now trained for 38 epochs, yields an accuracy of 81% and a loss function value of 0.48. Subsequently, various modifications to the original model are explored to enhance results, in line with textbook recommendations. These variations encompass adjustments to the number of layers, their dimensions, switching to the mean squared error (MSE) loss function, and experimenting with the tanh activation function instead of relu. Ultimately, the most effective hyperparameters include a two-layer configuration with dimensions of 16 units and 1 unit, using relu and sigmoid activations, respectively, along with the rmsprop optimizer, MSE loss function, 38 epochs, and a batch size of 512. Despite the accuracy remaining at 81%, the loss function decreases to 0.15.

Binary Classification

Hyperparameters	Round 1	Round 2	Round 3	Round 4	Round 5	Round 6	Round 7	Round 8
Number Of Layers (Counting the Final)	3	3	3	3	3	2	2	2
Dimension Layer #1	16	16	16	64	4	16	16	16
Dimension Layer #2	16	1	16	1	1	1	1	1
Dimension Layer #3	1		16					
Dimension Layer #4			1					
Activation Function #1	relu	relu	relu	relu	relu	relu	tanh	relu
Activation Function #2	relu	sigmoid	relu	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid
Activation Function #3	sigmoid		relu					
Activation Function #4			sigmoid					
Model Optimizer	rmsprop	rmsprop	rmsprop	rmsprop	rmsprop	rmsprop	rmsprop	rmsprop
Loss Function	Binary Crossentropy	Binary Crossentropy	Binary Crossentropy	Binary Crossentropy	Binary Crossentropy	MSE	MSE	MSE
Epochs	38	38	38	38	38	38	38	38
Batch Size	512	512	512	512	512	512	512	16
Results								
Loss Function	0.48	0.48	0.48	0.48	0.48	0.15	0.15	0.15
Accuracy	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81

(The yellow cells show the changes done at each round and the red numbers highlight the most optimal model.)

The second model developed was designed for multi-class classification, specifically on a Trip Advisor dataset obtained from Kaggle. In this case, the X values comprised the reviews, while the Y values represented the ratings. Notably, all ratings were adjusted by subtracting one to align with Python's indexing convention, which starts at zero, resulting in values ranging from 0 to 4. Similar to the binary classification example, the text data was tokenized into integers. A conventional train-test split was performed, and the X values were vectorized. However, the Y values took a different approach and were not vectorized as in the previous example.

For multi-class classification, the Y values were processed using a function from tensorflow-keras-utils, which transformed the five possible Y outcomes into categorical data using the to_categorical function. The model itself consisted of three layers. The first and second layers had 64 units each, employing a relu activation function. The final layer, serving as the output layer, had a dimension of 5, matching the number of classes, and utilized a softmax activation function, which is particularly suited for multi-class models. The model was

compiled in a manner similar to the binary classification model, with categorical_crossentropy as the loss function. The validation set consisted of 10,000 values from the training data and was used to evaluate the relationship between loss and epochs and accuracy and epochs. The optimal number of epochs, as determined from the plots, was 25, and this value was subsequently used for testing. Thus, an identical model was constructed, but this time using the test set and 25 epochs. The results showed a loss of 1.36 and an accuracy of 43%. To benchmark the model's performance, it was compared to a random classifier that achieved a 30% accuracy, indicating the superiority of the newly created model.

Several variations were explored, akin to those implemented in the binary classification model; however, none yielded superior results. Consequently, the most efficient model was identified as one that is computationally less demanding – featuring two layers, one with 16 dimension-units and a relu activation function, and the other comprising five dimension-units with a softmax activation function.

Multi-class Classification

Hyperparameters	*Round 1*	*Round 2*	*Round 3*	*Round 4*	*Round 5*	*Round 6*	*Round 7*
Number Of Layers (Counting the Final)	3	3	3	3	2	4	2
Dimension Layer #1	16	16	10	32	16	16	16
Dimension Layer #2	16	16	10	32	5	16	5
Dimension Layer #3	5	5	5	5		16	
Dimension Layer #4						5	
Activation Function #1	relu	relu	relu	relu	relu	relu	relu
Activation Function #2	relu	relu	relu	relu	softmax	relu	softmax
Activation Function #3	softmax	softmax	softmax	softmax		relu	
Activation Function #4						softmax	
Model Optimizer	rmsprop	rmsprop	rmsprop	rmsprop	rmsprop	rmsprop	rmsprop
Loss Function	categorical_crossentropy	sparse_categorical_crossentropy	categorical_crossentropy	categorical_crossentropy	categorical_crossentropy	categorical_crossentropy	categorical_crossentropy
Epochs	25	25	25	25	25	25	25
Batch Size	512	512	512	512	512	512	16
Results							
Loss Function	1.36	1.36	1.36	1.36	1.36	1.36	1.36
Accuracy	0.43	0.43	0.43	0.43	0.43	0.43	0.43

A regression model was developed to predict apartment prices in Poland for the month of October, considering 16 distinct numeric variables. The initial steps mirrored those in the previous examples, encompassing the definition of X and Y values, where X included all the numeric features, and Y represented the price. A standard train-test split was applied. However, since the model dealt with numeric values, feature normalization was necessary to ensure uniform impact across all variables. Each value was transformed into a standard deviation value from the mean. Subsequently, a basic model was constructed with three layers. The first two layers each had 64 dimension-units and utilized a relu activation function. The final layer contained only one dimension-unit and had no activation function since it was dedicated to predicting a single price.

In contrast to the previous examples, a K-fold validation model was employed, utilizing a validation dataset to identify the optimal number of epochs. A total of 500 epochs were run using the validation data, and results were saved in a dictionary. This data was then used to create a plot illustrating the relationship between the number of epochs and mean absolute error (mae). To enhance the plot's clarity, a second version was generated, excluding the first 10 epochs, which often included outliers. This analysis suggested that around 70 epochs was optimal for training the final model with the test data. A subsequent model was built with 70 epochs and a batch size of 16, resulting in a mae of $281,251. This figure holds significance, given the price range between $16,900 and $2,500,000 and a mean price of $757,935.

Further experiments were conducted to enhance the model's performance, revealing that as more layers with high dimensions were added to the neural network, the mae consistently decreased. Various optimizers, including adam, nadam, and adagrad, were also explored. Ultimately, the most efficient hyperparameters for this model comprised five layers, with four of them featuring 1024 dimension-units and a relu activation function. The adam optimizer was adopted, resulting in a reduced mae of $125,176, a significant improvement over the original performance.

Regression

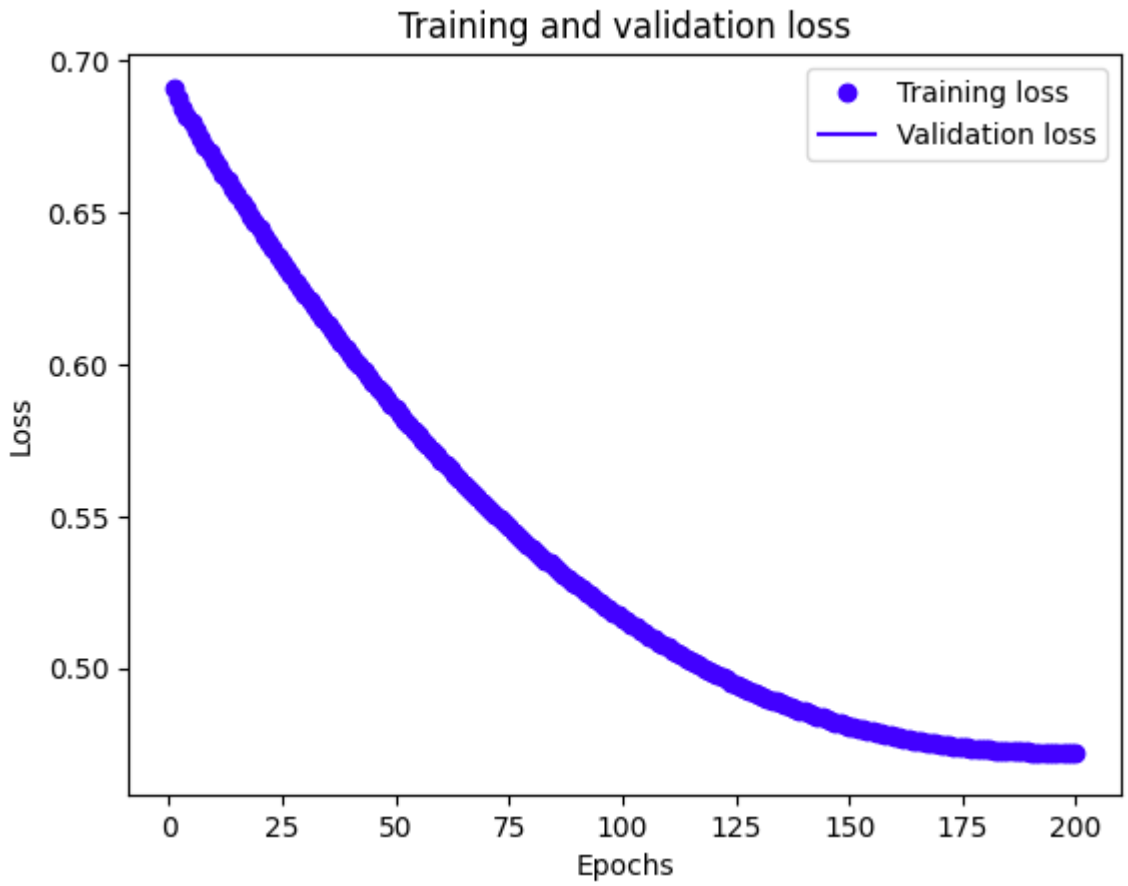
Hyperparameters	*Round 1*	*Round 2*	*Round 3*	*Round 4*	*Round 5*	*Round 5*
Number Of Layers (Counting the Final)	3	3	3	4	5	5
Dimension Layer #1	64	1024	16	1024	128	128
Dimension Layer #2	64	1024	16	1024	128	128
Dimension Layer #3	1	1	1	1024	128	128
Dimension Layer #4				1	128	128
Dimension Layer #5					1	1
Activation Function #1	relu	relu	relu	relu	relu	relu
Activation Function #2	relu	relu	relu	relu	relu	relu
Activation Function #3				relu	relu	relu
Activation Function #4					relu	relu
Activation FUnction #5						
Model Optimizer	rmsprop	rmsprop	rmsprop	rmsprop	rmsprop	adam
Loss Function	mse	mse	mse	mse	mse	mse
Metrics	mae	mae	mae	mae	mae	mae
Epochs	70	70	70	70	70	70
Batch Size	16	16	16	16	16	16
Results						
MAE	281,251	182,243	684,710	160,332	132,337	125,176

CONCLUSION

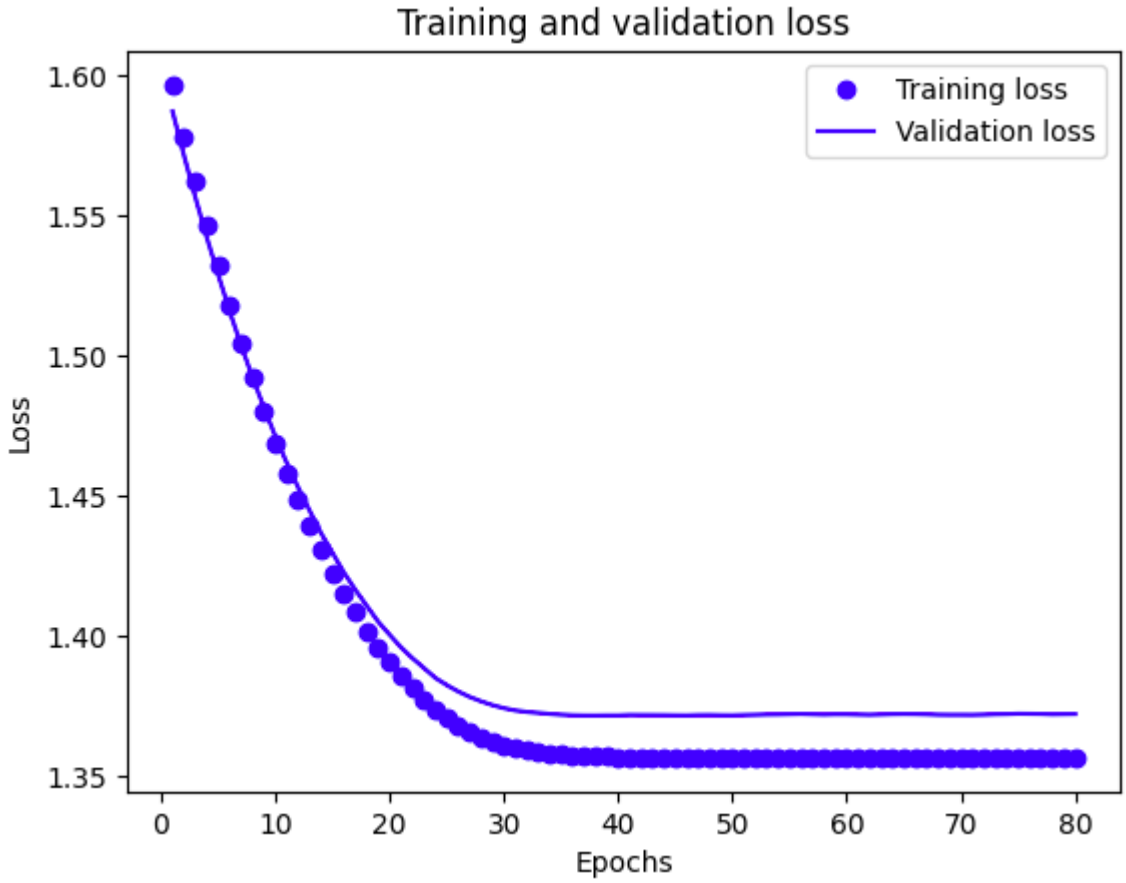
Overall, the performance of these models was notably strong. In binary classification, the model achieved a minor loss function of 0.15 and an accuracy of 81%. The multi-class classification model, while displaying a higher loss function and lower accuracy (1.36 and 43%, respectively), still outperformed a balanced random classifier. Lastly, the regression model demonstrated substantial improvements in mean absolute error (mae) by incorporating additional layers with increased dimensionalities and leveraging the adam optimizer.

APPENDIX

Binary Classification:



Multiclass Classification:



Regression Model: