CPSC 393 Assignment #2 Report Nikki Poentis

Abstract

As we are learning more about neural networks in machine learning, this assignment serves as an introduction to the coding/application side. Using the keras library in Python, the goal was to follow along with the 3 coding examples provided in our textbook using our own data for binary classification, multi-class classification, and regression analysis using neural nets. After completing each example, the book also gives us "experiments" to carry out so that we can explore the affect changing some of the models' hyperparameters has. By looking at each models' training, validation, and testing loss/accuracy, we can see that the 3 neural networks performs at different levels.

Introduction

Since we only cover the conceptual aspects of machine learning during lectures, these chapters of our textbook serves as guide for writing code for neural networks. In this assignment I chose 3 different datasets that could be used for the coding examples and I chose to write each in its own separate jupyter notebook file so the naming conventions did not have to be overly complicated. For binary classification, I used a dataset from one of my previous classes that looks at credit card customers and whether they defaulted on their debt. For multiclass classification, I found a dataset online including data on customers of an automotive company which were segmented into 7 different categories. Finally, for the regression analysis, I used a dataset measuring the life expectancy for multiple developing countries.

Methods

For all three of the networks, the data being inputted included multiple variables all measuring a different feature. Because of this difference, the data needed to be standardized so that the range for each variable is similar. However, since this was not an issue for the first two examples in the textbook, I did not think to make this adjustment for my two classification networks, which resulted in two training vs validation loss and accuracy curves that made no sense. After copying the code for normalizing the data from the regression analysis file, pasting it in the binary and multiclass classification files, and re-running the code, the curves looked more like what would be expected. Before training each model, I first had to split the data into training and testings sets; however since the credit and customer datasets for the two classification networks were already separated into train and test csv files, I only needed to this for the life expectancy dataset.

Since I was just following along with the textbook, I just used the same amount of epochs as the example in my code, so it most likely didn't have the same affect in mitigating overfitting like the book originally intended. Overall, the same basic steps were followed for defining, compiling, training, and testing for all three models. The only difference in the actual coding process between the three different networks is the actual means of validating the model's performance while training. For the two classification models, I split the training data into two subsets, partial-training data and validation data, and trained the model using the partial-training data as input and the validation data as its own parameter. After fitting the models, I plot the training vs validation loss and accuracy curves to visualize the overall fit of the model. On the other hand, for the regression model, instead of splitting the training data only once, I used k-fold validation so that the model was trained and evaluated at each fold while generating a list of MAE values at each iteration. After validating, I then re-trained the models from scratch using all of the training data, tested the model using the testing data, and generated predictions.

The hyperparameter values for defining and compiling the models are the following…

Binary Classification:

- 3 Dense layers (2 hidden)
- 16 hidden units
- relu & sigmoid activation functions
- rmsprop optimizer
- binary cross entropy loss function
- accuracy metric

Multiclass Classification:

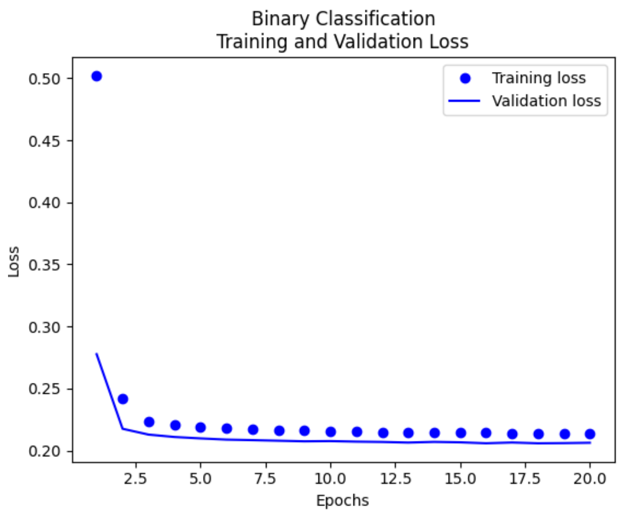
- 3 Dense layers (2 hidden)
- 64 hidden units
- relu & softmax activation functions
- rmsprop optimizer
- categorical cross entropy loss function
- accuracy metric

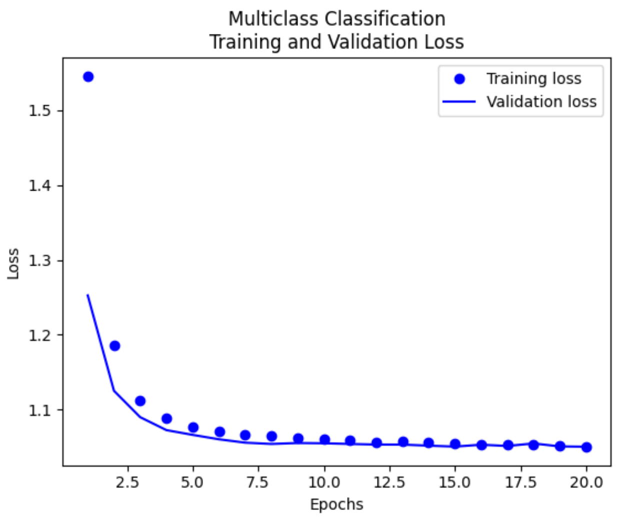
Regression Analysis:

- 3 Dense layers (2 hidden)
- 64 hidden units
- relu activation functions
- rmsprop optimizer
- mean squared error (MSE) loss function
- mean absolute error (MAE) metric

Results/Discussion

Classification Networks:





Using 4 epochs when training the final model, my binary classification model ended up with a test accuracy of 93% which is very good. Although a high accuracy value is usually signs of overfitting, the learning curve (training vs validation loss) seems to indicate a good fit. Also, a higher training than testing accuracy is seen in overfit models, since this model's training accuracy is only less that 1% larger, it is possible that this binary classification model is not overfit and actually performed well. On the other hand, when using 9 epochs, the multi-class classification model did not perform well and had a 67% test accuracy. Since this network also showed the same signs of not overfitting as the binary model, this lower accuracy for both training and testing is most likely due to the model not having enough data to learn from; especially considering the binary classification model–that performed well–had 73,304 data points for training while this model only had 6,665.

Regression Analysis:

Although I can't evaluate the regression model using accuracy like the two classification models, we can still measure its performance using the mean-absolute error. For the final regression model using 80 epochs, since the testing MAE score was 1.998–which is small considering the average life expectancy is around 70–this regression model performed well.

Experiments

The textbook's further experiments were performed on the binary and multiclass classification networks. By comparing the test results from the experiments with the test results from the "base" model, I was able to learn these insights about how choosing different parameter values can have on the two classification models.…

Experiment #1 - Binary Classification:

a) 1 hidden layer (instead of 2) → slightly increases loss
b) 3 hidden layers (instead of 2) → slightly decreases loss

Experiment #2 - Binary Classification:

a) 32 hidden units (instead of 16) → slightly decreases loss
b) 64 hidden units (instead of 16) → slightly decreases loss, but larger than w/32 units

Experiment #3 - Binary Classification:

a) MSE loss function (instead of binary cross entropy) → decreases loss

Experiment #4 - Binary classification:

a) tahn activation function (instead of relu) → slightly decreases loss

Experiment #5 - Multiclass classification:

a) 32 hidden units (instead of 64) → slightly increases loss and accuracy
b) 128 hidden units (instead of 64) → slightly increases loss and accuracy

Experiment #6 - Multiclass classification:

a) 1 hidden layer (instead of 2) → slightly increases loss and decreases accuracy
b) 3 hidden layers (instead of 2) → slightly increases loss and accuracy

** Note that these slight "increases" and "decreases" are very small (< 0.001)

Conclusion

Since keras is pretty intuitive and easy to use, following along with the textbook's code for building the model was a simple task. Actually finding good data and transforming it into tensors that could be used turned out to be the hardest part of this assignment. When working with neural networks, the data must be inputted as tensors meaning that none of the data could be strings so it limited the amount of datasets that I could use. It took some time to find datasets that included majority quantitative variables; even after all that time search the datasets I ended up using had some categorical variables that I just chose to omit. Also, since my datasets were different from the data used by the textbook, not only were there extra steps that needed to be taken to prepare the data, but the models' results were never the same as examples. However, comparing to other classification and regression models I've done in other classes, using neural networks to carry out these analyses is a more complex yet shorter process.