AML Assignment 1

Francesco Stranieri 816551

f.stranieri1@campus.unimib.it

Introduction

The assignment consists in the prediction of the price of a private room or an entire apartment using a *neural network*.

The dataset includes all the necessary information to learn more about hosts, geographic availability, and necessary metrics of Airbnb apartments in New York City in 2019.

Preprocessing

First of all, I removed the columns that I did not consider useful, as the *Unnamed: 0* column.

Then, with respect to the percentiles, I removed all the rows with *outliers* in correspondence of *minimum_nights* and *price* columns. ~~For example, I removed all those rows with a *latitude* value smaller than -90 or greater than 90.~~ Finally, in correspondence of columns with a high variance, I scaled data in range $[0, 1]$ through the Min-Max technique.

Subsequent analysis have shown that the dataset has neither duplicates nor missing values.

NN Architecture

The input layer has a number of neurons equivalent to the number of features, which corresponds to 9 in this case.

The output layer has a single node, since this is a *regression* task.

According to (Heaton, 2008)[1] I chose two hidden layers, because they can represent functions with any kind of shape. The "rule-of-thumb" method that I followed for the number of neurons in the hidden layers says that there should be less than twice the size of the input layer. So I chose 16 for the first layer and 8 for the second one.

[1]Heaton, J. (2008). Introduction to neural networks with Java. Heaton Research, Inc..

Optimizer and Learning Rate

With the *"Learning Rate Range Test"* Package[2], I tried to find an estimate of the optimal *optimizer* and its relative *learning rate*.

The technique used, described in (Smith, 2018)[3], consists to train a network starting from a low learning rate and increase it exponentially for every batch.

For this task, I chose to test *Adam* which performs similar to *RMSprop* and *Adadelta* as well as *momentum* [4] and *SGD*, with a learning rate ranging from $10e-4$ to 1. The results, shown in Figure 1, made me bet on *Adam* optimizer with a learning rate equal to $10e-3$.

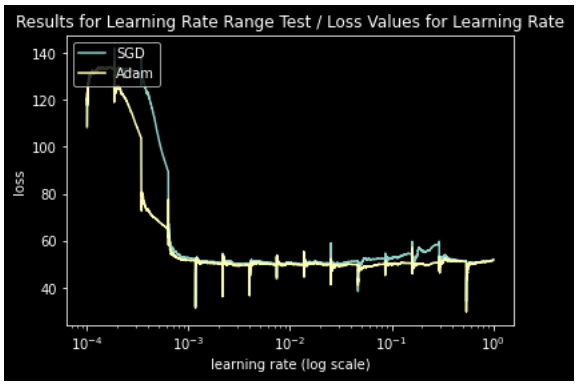


Figure 1: Loss values for Adam and SGD, according to the learning rate.

Loss Function

I chose the *Mean Absolute Error* (MAE) over the *Mean Squared Error* (MSE) because the average error was not at all small. In fact, with the MSE large errors introduce a much larger cost because the differences are squared and larger errors produce much larger squares than smaller errors.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

Activation Function

The activation functions that I tested are: ReLu, Leaky ReLU, ELU, GELU, SELU and Swish. Looking at Figure 2, *GELU* seems to be the most stable,

[2]Versloot, C. (2017). Keras_lr_finder. Retrieved October 26, 2020, from `https://github.com/christianversloot/keras_lr_finder`

[3]Smith, L. N. (2017, March). Cyclical learning rates for training neural networks. In 2017 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 464-472). IEEE.

[4]Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

with a RMSE near to 75 for the *validation-set* (this latter equivalent to 15% of the entire dataset).

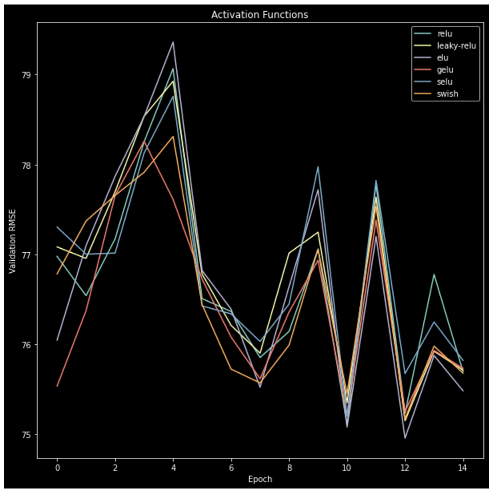
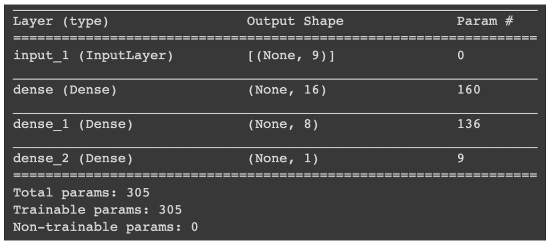


Figure 2: RMSE on *validation-set* for the activation functions.

For the output function I chose the *linear* one, since my codomain is $[0, +\infty)$ and ReLU is not particularly used as output function.

Results

The final architecture is shown in Figure 3.



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 9)]               0
_________________________________________________________________
dense (Dense)                (None, 16)                160
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 136
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 9
=================================================================
Total params: 305
Trainable params: 305
Non-trainable params: 0
```

Figure 3: Final architecture of the neural network.

The *batch size* selected is 32, while the number of epochs is 15. The loss value obtained with this configuration on the entire *training-set* is around 48, with an RMSE about to 78.

Lastly, the mean of the predictions on the *test-set* is 126.1 ± 56, which happens to be near to the mean of the price on the *training-set*, that is 134 ± 94 (this makes sense under the assumption that the data come from the same distribution).