Department of computer science
Umeå University

March 10, 2021

Machine Learning
5DV194 Spring 2021
Assignment 2

Name	**Email**
Klas Holmberg	hed16khg@cs.umu.se

March 10, 2021

Teacher: Lili Jiang

Contents

1 Introduction **1**

2 Algorithm 1: Linear regression **1**

3 Algorithm 2: Neural Networks - Regression **1**

4 Reflections **2**

References **3**

Department of computer science
Umeå University March 10, 2021

1 Introduction

In machine learning many different approaches to solving problems by utilising data are out there, they all strive to do different things depending on the variation found within the data. This assignment is built on exploring the `Tensorflow` library `Keras` [1] and practice using `sklearn` [2], to do this the `Boston Housing`[3] dataset is utilised.

2 Algorithm 1: Linear regression

The basic premise of Linear regression is to create a linear prediction model out of the training data that is the result of the sum of the least minimum square errors (MSE). The results of building a basic linear regression model of the dataset and plotting the prediction against the true labels can be seen in Figure 1.

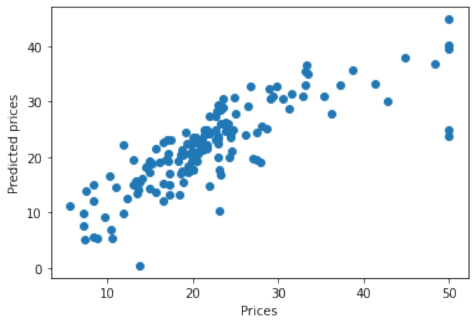


Figure 1 – A scatter plot of the results of a basic `sklearn` linear regression model on the boston dataset.

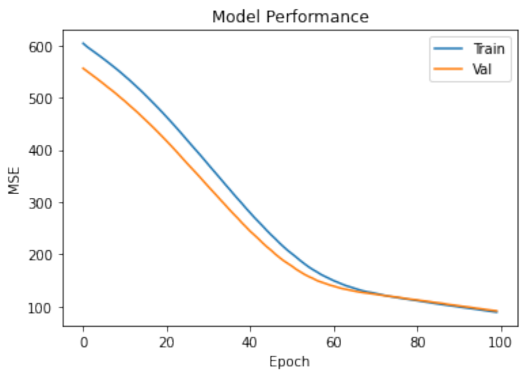
The importance of having a non-biased dataset cannot be overstated, but there are different types of Linear regression methods to combat bias in the data and thus also combating overfitting. One of the types of methods are `Ridge Regression`, a linear regression method that utilises a bias penalty term to decrease the high variance outcomes. The difference in `RMSE` (root mean square error) can be seen in Table 1.

Table 1 – The difference in RMS between a regular linear regression model and a ridge regression model.

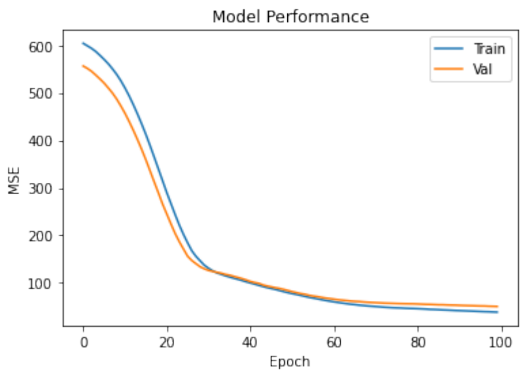
Regression Type	**RMSE**
Linear regression	5.214975
Ridge regression	5.268987

3 Algorithm 2: Neural Networks - Regression

`Keras` by `Tensorflow` is a popular Python library used for AI and Machine Learning, there are many types of building blocks for Neural networks in the library which all do different things and have different settings, thus leading to a really big set of possibilities in configurations of networks. The resulting differences of two very simple architectures can be seen in Figure 2. At this level, by simply adding a hidden layer between the input and the ouput layer improves the `MSE` by quite a lot, at this amount of training the extra time it takes to train the more complex model is insignificant.



(a) 2 layer (in: 8, out: 1)



(b) 3 layer (in: 8, hidden: 16, out: 1)

Figure 2 – The Mean Square Error of two different Neural Network architectures over 100 epochs. `a)` is a simple 2 layer network with an input layer of 8 nodes and an output layer with one node. `b)` is a 3 layer network with the same input and output layer setup but with a hidden layer in between consisting of 16 nodes and having the `relu` activation function.

Another type of layers are so called 'Dropout layers', these layers try to help the model avoid overfitting through "resetting" edges in the architecture at a certain probability. In Figure 3 the MSE result of a `NN` with 4 layers can be seen and in comparison to the earlier versions it is clear that this type of model gives voice to more "noise/static", which is generated by the dropout layers (which is the general idea). Furthermore the speed of which improvements are made is faster than both the earlier models, dropping to roughly 100 at 20 Epochs.

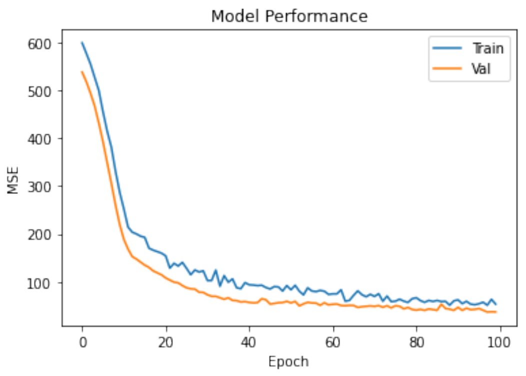


Figure 3 – The MSE of a 4 layer *NN* with dropouts set to a probability of 0.2. In: 8, Hidden 1: 16, Hidden 2: 16, Out: 1.

4 Reflections

The sheer size of configurations one could make for the Keras library is amazing, and somewhat daunting. It can be hard to see what type of setups are used for what situations, the linking between this is not that intuitive, BUT, i also see that exploring is probably the only way of actually getting some sense of what models are good for what type/shape/variance of dataset. So, i wouldn't fault the assignment for that. It was really interesting to see how fast improvements are made by adding layers. It would be really interesting to see the trade-off of training time though over this in a more "real" way, at these sizes it's a second or two, but getting an idea of the magnitude of training time of larger networks is hard to see without actually trying it out.

A really fun assignment, i really like this format of assignment, Jupyter + Python + ML is a great way to learn about these things. It makes the purpose of aspects of the assignments quite easy to comprehend, why one would add dropout layers and such.

References

[1] *Tensorflow Keras library*
https://keras.io/

[2] *Sci-kit learn library*
https://scikit-learn.org/stable/

[3] *Boston Housing Dataset*
https://www.cs.toronto.edu/ delve/data/boston/bostonDetail.html