Resit Project 1

Jan van Ruijven 2006698

January 23, 2024

1 Question 1

First the data is split up into three parts: training, validation and testing. The training set is set te be 55% of the data, the validation set 25% and the testing set is 20% of the data. The hyperparameters are split up into two catagories, the non-numeric hyperparameters which can be set to a specific value which is not a number. To this category belong the optimization algorithm, initialization and regularization. These hyperparameters are tested with the following values:

Optimization algorithm: Stochastic gradient descent and Adams optimizer

Initialization: GlorotNormal and GlorotUniform

Regularization: L1 and L2

The overall strategy is to first find the optimal combination of non-numerical hyperparameters, and find the optimal combination of numerical parameters afterwards. To test the non-numerical hyperparameters, a Neural Network is needed, to start things off the following Neural Network with numerical hyperparameters and one hidden layer is generated:

Number of nodes: 15

Batch size: 15

Learning rate: 0.01

Regularization rate: 0.01

After testing the non-numerical hyperparameters, the following combination seems to perform best on both the training and validation set: GlorotUniform initialization, Adams optimizer and L2 regularization. With these hyperparameters the numerical hyperparameters are being tested. The overall idea behind the strategy is to increase and decrease our hyperparameters and save the parameters that hhave been perform best so far, each time we do an iteration that is either not improving or seems te be overfitting on the data, the hyperparameters are being reset back to the best combination we have found so far. In each 10 fold of iterations we are decreasing and increasing the amount by which the hyperparameters are being adjusted, for the nodes and batch size, 10, 7, 4, 3, 2 and 1 are the sizes by which we are increasing and decreasing their values each 10 iterations respectively. For the regularization and learning rates the idea is completely similar, but now the amount is multplied by 0.01.

After running the strategy the following neural network is obtained:

Number of nodes: 7

Batch size: 21

Learning rate: 0

Regularization rate: 0.006

This Neural Network is able to produce a score of 0.11 on the testing set. A figure showing the best total score can be found in the appendix at figure 1. In this figure it can be seen that the best performing network is foun dat the end round iteration 50 and that the overal performance has large deviation in the beginning, but becomes more stable after around 30 iterations.

2 Question 2

A

Since we can control the output of the Relu activation function, any output can be produced by $Transpose(W_{j1}^2) * a + b_1^2$. Even setting the weights to just ones only, would not be a problem for outputting any real number, as long as the output of the activation function can be determined.

B

To replicate the Relu activation function we need to apply the max function in linear fashion. This is done using a large number M, a binary variable z, a variable for adjustment S, and the output of the Relu functin as A. To write everything out we first denote the following: the number of nodes is represented by K, the number of observations by N and the number of parameters by P. To write out the Relu function the following is done:

$$A_{i,j} \leq (1 - Z_{i,j}) \cdot M \quad \text{for } i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, \mathrm{K}$$
$$A_{i,j} \geq 0 \quad \text{for } i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, \mathrm{K}$$

$$S_{i,j} \leq M \cdot Z_{i,j} \quad \text{for } i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, \mathrm{K}$$
$$S_{i,j} \geq 0 \quad \text{for } i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, \mathrm{K}$$

$$\sum_{p=1}^{P} W_{\text{eights1},j,p} \cdot X_{\text{train},i,p} + B1_j - A_{i,j} + S_{i,j} = 0$$
$$\text{for } j = 1, 2, \ldots, \mathrm{K}, \quad i = 1, 2, \ldots, N$$

Now that the first activation function is defined, the step function can be created, also for the step function the large number M is needed to make sure that the output of the step function is less than 0 when our predicted value is 0 and the output is greater or equal than 0 when the predicted output value equals 1. To do this, the following constraints are created:

$$\text{for } i = 1, 2, \ldots, N:$$
$$\sum_{j=1}^{\mathrm{K}} W_{\text{eights2},j} \cdot A_{i,j} + B2 - Y_{\text{pred},i} \cdot M \leq 0$$

To produce the absolute value in the objective function, an auxilary variable must be created which is always greater or equal than the absolute value of the true value minus the predicted value.

$$\text{for } i = 1, 2, \ldots, N:$$
$$Y_{\text{train},i} - Y_{\text{pred},i} \leq \text{aux}_i$$
$$-Y_{\text{train},i} + Y_{\text{pred},i} \leq \text{aux}_i$$

In an effort to stop the model from overfitting, L2 regularization is added. Therefore, the following objective value is obtained:

$$\text{minimize } \sum_{i=1}^{N} \text{aux}_i + \lambda \left(\sum_{j=1}^{\text{K}} \sum_{p=1}^{P} W_{\text{eight1},j,p}^2 + \sum_{j=1}^{\text{K}} B1_j^2 + B2 \right)$$

3 Question 3

After training and testing the neural network a training score of around 0.22 is found and a testing score of around 0.11. As for the MILP, when searching for the optimal solution, this is found relatively quickly and the predicted values equal the true values. But after testing the model either only 0's are returned, only ones, or random zeros and ones. I have tested a lot of different settings to try and reduce the overfitting, this would sometimes give very small improvements, but these seemed quite random. The improvements seemed random due to the fact that increasing my overfitting value lambda from 0.01 to 0.001001 would give a small improvement, but increasing it again by 0.000001 would have a negative effect. The overall advatage from the neural network over the MILP seems to be the fact that its easier to control for overfitting and underfitting while also producing good estimates. An MILP seems better in scenarios where (almost) identical data is inserted into the model each time, since the model can quickly find the optimal solution for one perticular set of input data.

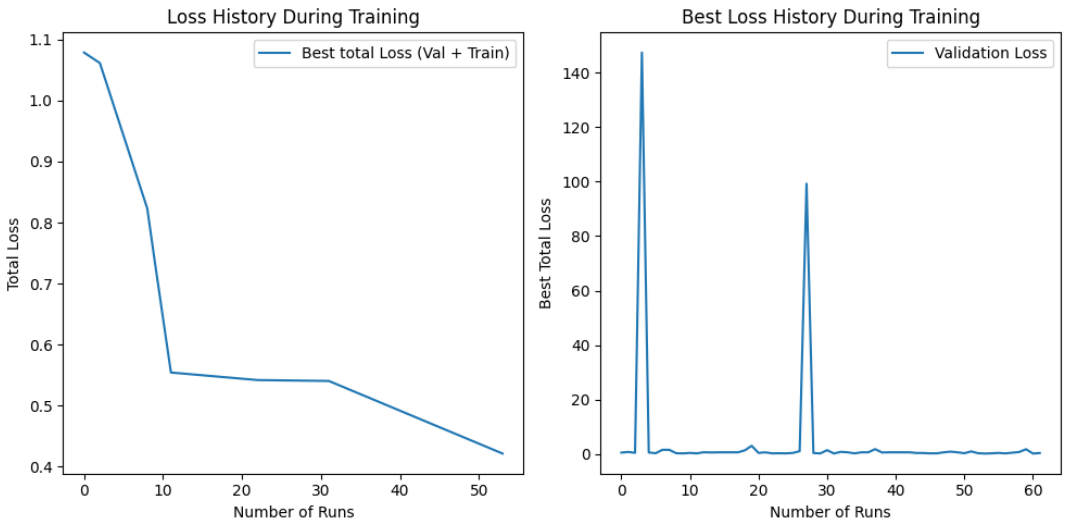

Figure 1: Best total loss and overall validation loss over time