UNIVERSITÉ LIBRE DE BRUXELLES

BRUFACE BRUSSELS FACULTY OF ENGINEERING

VRIJE UNIVERSITEIT BRUSSEL

Measurement and Data Driven Modelling

Lab 5: Neural Networks

Warre De Winne, Arno De Haseleer

19/05/2023

Pre-Processing Of Dataset

The data is pre-processed by first splitting the data. It is split into a training and validation set. The choice is made to use 20% of the data for validation and the other 80% for training. This ratio is chosen such that enough data is available for training, while also having enough data remaining for validation.

The second step normalises the separate data sets. This is done by using the $MinMaxScaler()$ and the $fit\_transform()$ function in the $sklearnlibrary()$. The $fit\_transform$ calculates the mean and standard deviation of the different features of the dataset and then normalizes them.

The data is after normalisation split into a training and validation set. The choice is made to use 20% of the data for validation and the other 80% for training. This ratio is chosen such that enough data is available for training, while also having enough data remaining for validation.

Neural Network Architecture

The network architecture contains of three main parts: the input layer, the hidden layer and the output layer. The input layer is chosen such that the amount of inputs is equal to the amount of features in the input data. The output layer is then chosen to have only one node, since only the global active power is predicted.

The hidden layer is a more complex problem. Here one has to consider the amount of neurons in each sub layer and also the amount of sub layers. These parameters are chosen mainly on their computational cost. One wants enough layers and neurons per layer to have a good model, but increasing layers and the amount of neurons will lead to a fast increasing computational cost. So one should stop adding layers if the results are only marginally better. Note that increasing the amount of nodes instead of the amount of layers will lead to less extra computational cost compared to just increasing the amount of layers, while still creating a better model. Also note that an overcomplex model can lead to overfitting.

The amount of layers and neurons should be determined by looking at what point the performance of the model stops to improve for increasing layers or neutrons.

Activation Function
The used activation function is the rectified linear unit. This function returns zero if the input is negative or zero and returns the input if the input is positive. There are multiple reasons for using this function. The first one is the computational simplicity. This function does not need to perform any multiplications, divisions, ... This is extremely handy is one tries do build a neural network on a simple computer with limited computational power. The second reason, is that it speeds up training. By using this instead of just sending the output to the next node, a faster learing process is achieved.

There is one drawback and that is that it might causes dead neurons. This is caused by the fact that the function can return zero. This can lead to zero weights in the back propagation and the neuron will then always output zero and thus be dead.

Cost Function

The cost function is calculated using the mean square error. This is done by defining the loss by using $nn.MSELoss()$. Then the loss is calculated by calculating the error between he actual output and the model output using $criterion()$.

Backpropagation

Backpropagation is implemented using the Adam optimizer. The Adam optimizer is a stochastic gradient descent method. The choice for this optimizer is again be made by the fact that the optimizer has a low computational cost.

The backpropagation itself is implemented by using the *backward*() function. This function computes the gradient from the Adam optimizer. Then *step*() is called to update the weights of the model.

Evaluation Of Model

The model is evaluated using the training data set. The metric used for evaluation is the most significant error. The use of the most significant error is obvious, since it is used to optimize the model. The plots for the amount of neurons in function of the Epochs are given in figure 1 and 2. For an increasing amount of Epochs, the MSE decreases and converges to zero. Increasing the amount of neurons in a layer leads to a faster decrease in the MSE. Adding extra layers creates a network that is too complex for the problem and leads to a worse MSE convergence. Note that a lower MSE or faster convergence is always paired with a higher computational cost.

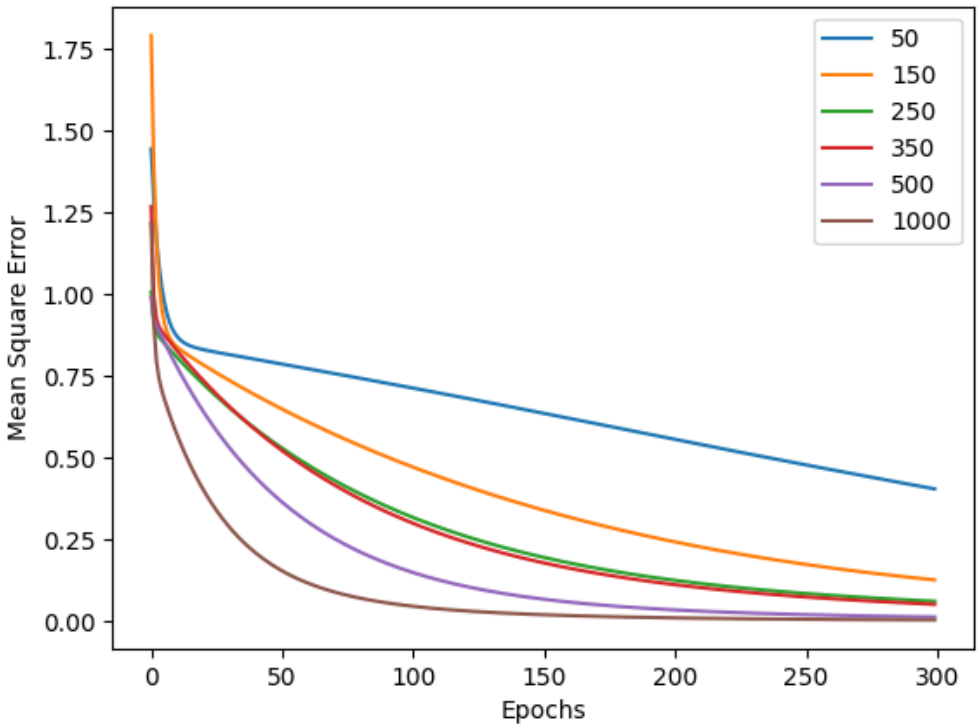

Figure 1: Neural network with 1 hidden layer

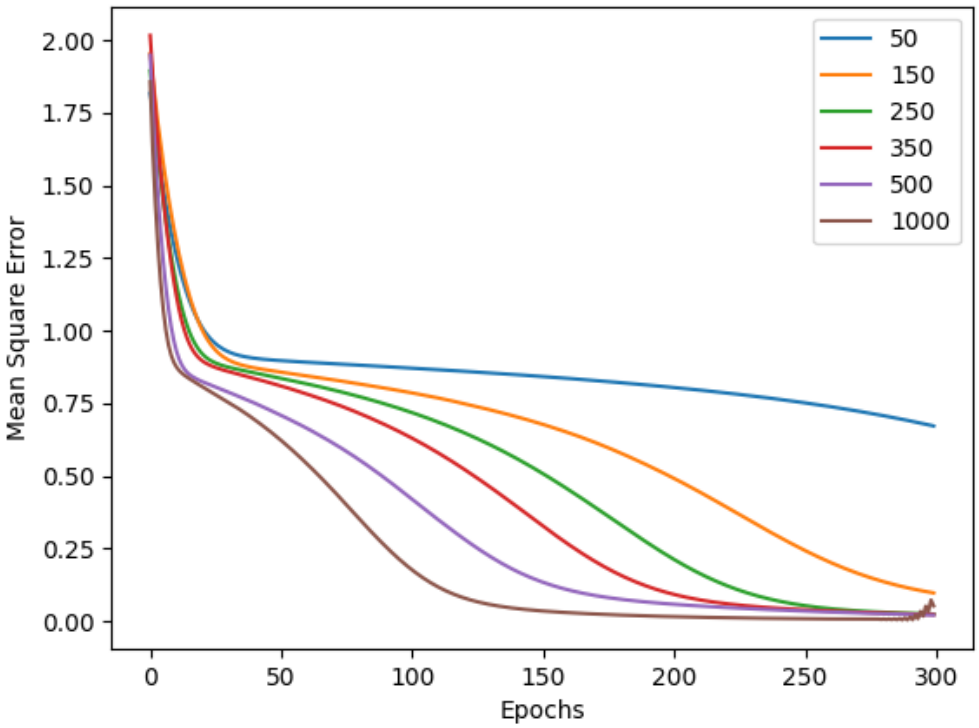

Figure 2: Neural network with 3 hidden layers