Hogeschool van Arnhem en Nijmegen
HAN University of Applied Sciences

Investigating the correlation between weather and domestic heat demand

Minor Project Report

By:

Aliya	613274
Karl	617931

Ehssaan

Abdul	602016

Guided by:

Carolien Stroomer

Marijn Jongerden

Paul Van Kan

Contents

Contents of Figures

Introduction

There are 2 broad techniques that are used to predict the future in Big data:

1. Regression

- Linear
- Logistic

2. Machine learning

- Neural networks
- Support vector machines

Regression establishes a mathematical connection between 2 or more variables under consideration.

Machine learning is a study of different algorithms and models such that it learns the patterns of events and inferring what would happen next. It is an integral part of AI (artificial intelligence). Based on current and historical data, a mathematical model is derived and based on the relationship it understands, it makes a predictive analysis. (Wikipedia, Predictive Analytics, 2019) (Wikipedia, Machine Learning, 2019)

Neural networks and Support Vector Machines will be used for this project.

Artificial neural networks (ANN) are systems that are inspired by actual brains and the functions of the neurons. These systems are trained to learn and perform tasks which we teach it to. They can be used for image recognition, medical diagnosis, network filtering, speech recognition and so on.
It is based on a collection of neurons. The sensory neurons (dendrites) are the input neurons, the hidden neurons and then output layer. (Wikipedia, Artifical Neural Networks, 2019)

Support vector machines (SVM) is a supervised learning method like neural networks where a training set is given to the algorithm and it creates a new hyperplane to then categorise new data. It is a black box model. SVMs work as large margin classifiers. It solves classification problems and has a flexible representation of class boundaries.

Objective

Using neural networks:

- Derive a relationship between weather and heating demand.
- Make predictions of heating demand based on the weather forecast.
- Determine the optimum learning rate, number of iterations (or epochs) and number of hidden neurons.

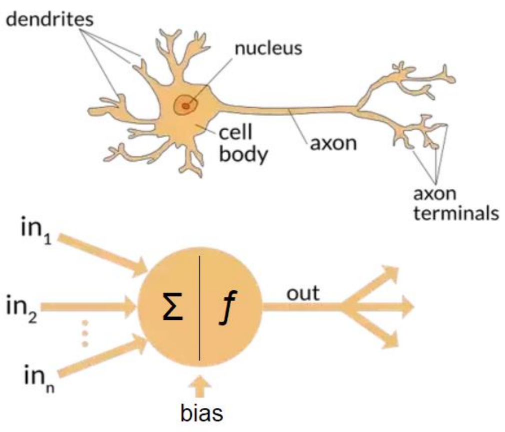
Literature survey

Neural Networks

Performing linear functions are not always preferable. If we have features that have nonlinear terms, calculations get complex and time consuming. Neural networks offer an alternative way to perform machine learning when we have a complex system: many features and complex hypotheses.

They basically mimic our brains to the best of our technology. Just like our brain uses one "algorithm" to learn for different abilities, the neural network attempts to do the same.

Every neuron has dendrites which take inputs from the surroundings as electrical impulses and transmits them to the output through the axon.



(Richárd, 2019)

There are 3 basic layers: input, hidden and output. There can be multiple hidden layers.

The dendrites are input features $(x_1, x_2 \dots x_n)$ and a bias unit is added (x_0) which is always equal to 1.

An activation or a sigmoid function is added which is the same as in the logistic function.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Here, θ =weights

The output from the input layer is the hypothesis function. The middle layer is also called the **activation layer**. They are represented by $\left(a_1^{(2)} \dots a_n^{(2)}\right)$ and another bias unit $a_0^{(2)}$ is added. The (2) in the superscript is because it's the second layer.

So now, the flow looks like:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} \rightarrow h_\theta(x)$$

The activation node is obtained by:

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

Here, Θ= matrix of weights controlling function mapping from one layer to the next.

The feedforward implementation using vectors is a faster technique and can be achieved as follows:

Assume: $a^{(1)} = x$

$$z^{(2)} = \Theta^{(1)}a^{(1)}$$
$$a^{(2)} = g(z^{(2)})$$

After the row of biases is added:

$$a_0^{(2)} = 1 = a_2^{(2)}$$
$$z^{(2)} = \Theta^{(2)}a^{(2)}$$

Now finally the hypothesis is calculated with:

$$h_\Theta(x) = a^{(3)} = g\left(z^{(3)}\right)$$

The cost is calculated using:

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2$$

The last term is called **regularisation** and it is included to avoid overfitting. It does this by removing larger weights.

The cost function should be minimised to the best of capabilities and to achieve this, we use **back propagation**. This can be done using the **gradient** function.

These errors or mismatches are there in the weights (θ)

First, set $\Delta_{\mathrm{ij}}^{(l)} = 0\ for\ all\ i, j, l$

For $i = 1\ to\ m$

Set $a^{(1)} = x^{(i)}$

This is calculated also during forward propagation where the *delta* is the difference between the expected and actual output.

$$\delta^{(L)} = a^{(i)} - y^{(i)}$$

For each node, the delta is calculated and the g-prime can be calculated. The g-prime is the derivative of the activation function evaluated with the input values given by z_2 and multiplied elementwise.

$$g'(z^{(l)}) = a^{(l)} .* \, (1 - a^{(l)})$$

The delta for each activation layer is calculated either using a vectorised or a non-vectorised function:

$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta^{(l+1)}$$

The deltas multiplied with their respective activations and dividing by the number of samples, with regularisation gives the gradients.

$$D_{ij}^{(l)} := \frac{1}{m}\left(\Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}\right) \; if \; j \neq 0$$

And

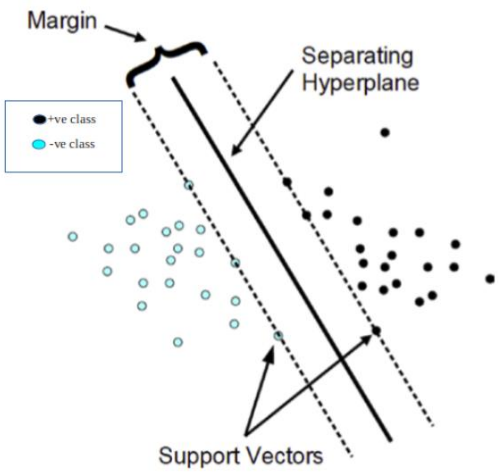
$$D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)} \; if \; j = 0$$

The minimised cost function is now:

$$\frac{\partial}{\partial \Theta_{ij}} J(\Theta) = D_{ij}^{(l)}$$

Source: (Nguyen, Coursera, 2019)

Support Vector Machines (SVM)



Terminologies used:

1. Hyperplane: the decision boundary that differentiates between features
2. Support vectors: the points closest to the hyperplane
3. Margins: the distance of the vectors from the hyperplane

Regions of interests (ROIs) are determined by taking the clusters of information presented in the training set. They are also known as hypothesis generation. A regularisation parameter (C) will determine how much the SVM should avoid misclassifying the training examples. The hypothesis function is given by:

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

The C is chosen according to the amount of bias and the variance we need.

- A large C implies lower bias but higher variance
- A smaller C implies higher bias but lower variance

With every new data, it is trained better to fit into the kernel. The kernel is decided based on the similarity function. In this project, a medium Gaussian kernel has been used.

Gaussian functions are used to represent a probability density function.

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}((x-\mu)/\sigma)^2}.$$

The σ^2 is chosen such that:

- A larger value has higher bias but lower variance. The features vary more smoothly.
- A smaller value has lower bias and higher variance. The features vary less smoothly.

For the data set:

- Number of training examples (m)=281
- Number of features (n)=6

When $n < m$, the Gaussian kernel is used. (Nguyen, Coursera, 2019)

A kernel is decided based on the parameters C and σ^2 and it solves for the parameters θ

Note:

- A feature scaling needs to be done before applying Gaussian kernel
- The similarity function needs to satisfy the Mercer's Theorem which states that the function needs to be a symmetric positive-definite. Some examples of this are polynomial kernel, string kernel, chi-square kernel, histogram and intersection kernel.

Three Methods and their results

Manual implementation of NN

Method

This Project was originally intended to provide more information for the Ecovat Project. As for energy storage a prediction of heat demand would be beneficial, the goal of our work became the prediction of heat demand based on weather data.
For this project, 2 datasets were used. One of these datasets was delivered by a member of HAN. It contains the weekly heat production of a house with a heat pump and a gas heating. This dataset

would provide the information about the heat demand, the output of the model. The other dataset used was the weather data of the closest weather station (Instituut, 2019). This dataset would provide the inputs to the model, consisting of the average weekly temperature, the average weekly windspeed, the average weekly sunshine duration the overall sum of sunshine duration, the average hourly precipitation and the sum of precipitation per week. This data was gained from hourly values provided by the Dutch weather service.

The heat production data was manually cleaned to get rid of outliers, measurement errors and other unwanted values. Dimensions of each column were checked, gaps filled and intervals between data checked for uniformity. As the heat production was given as an accumulative sum, the heat production per week needed to be extracted.

A neural network with one hidden layer was then trained with this data. The Neural net was based on the exam given in BDSD class. As the NN delivered good results for that application, it can be safely assumed, that this NN was well implemented.

To feed the data into the NN, the data preparation was adjusted to read the data properly. The size of input and output neurons was now automatically bound to the number of inputs and outputs.

For the NN to function properly, the data was normalized with the z-score function.

The data was split to 70% for training and 30% for testing.

As the NN was doing classification in the assignment, the evaluation had to be changed as well. New error metrics for this prediction model had to be established. Mean absolute Error, mean absolute relative error and mean absolute percentage error are a few to name.

Now the network could be run with different numbers of hidden neurons and different learning rates.

Algorithm

1. Randomly initialise the weights
2. Implement forward propagation to get $h_\theta(x_i)$ for any x_i
3. Implement the cost function
4. Implement back propagation to compute partial derivatives
5. Use gradient checking to confirm that your backpropagation works.
6. Use gradient descent to minimise cost function with the weights in theta

Results

For the training, a learning rate of 1 with a maximum number of 500000 epochs was chosen. Other learning rates were tried with no change in results.

The NN needs ca. 45 seconds to train on this data with 50 hidden neurons. The output of the NN can be compared with the expected values in the following graphs:

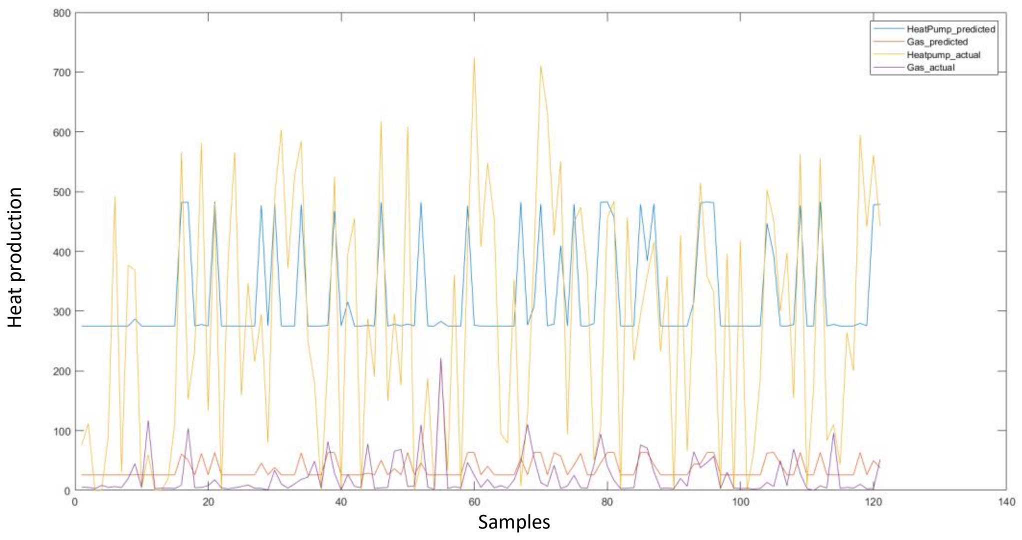


Figure 1: NN output vs expected values – 50 hidden neurons

It can be seen, that the NN is trying to adjust to the values. Nevertheless, it does not manage to meet the expected values. Similar outputs can be observed over several training attempts.

Increasing the number of hidden neurons to 150 blows up the training time to 123s. the results can be seen in the following graph:

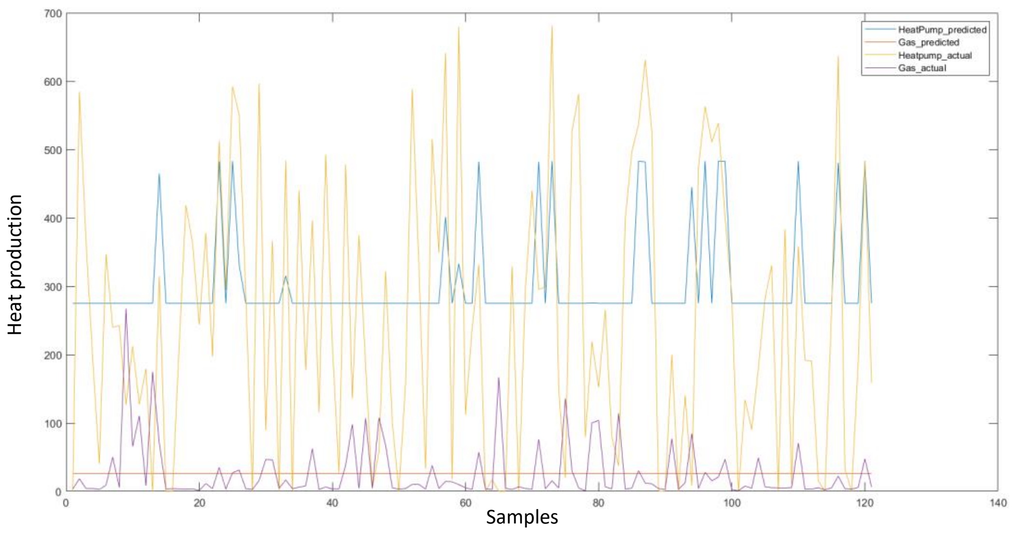


Figure 2: NN output vs expected values - 150 hidden neurons

As the results do not get any better, it can be safely assumed, that a further increase in neurons will not improve the results.

As the output of the network clearly don’t match with the expected values, a further evaluation of the error metrics will be spared.

NN MATLAB toolbox

Method

As the results with the manual implementation of the NN were not satisfying, the NN MATLAB toolbox for neural fitting was applied to the data. This toolbox applies a 2-layer feed-forward network with sigmoid hidden neurons and linear output neurons.

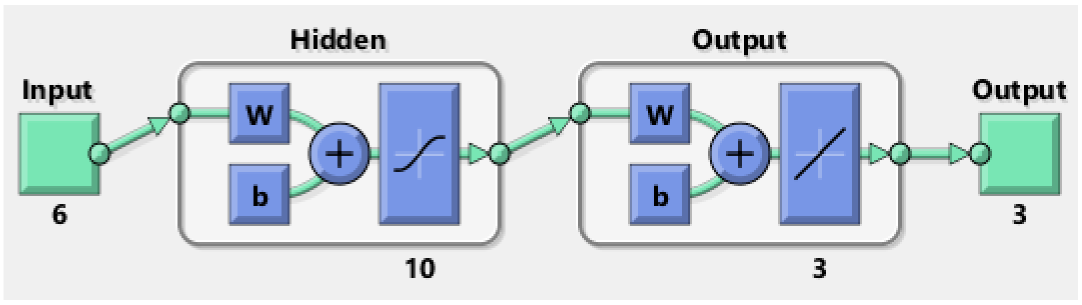
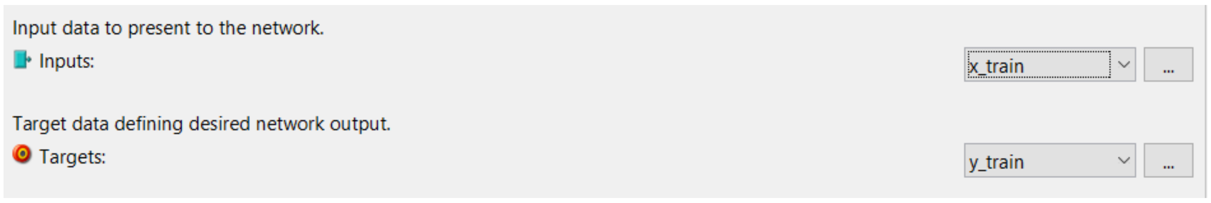


Figure 3: NN as used by the neural fitting toolbox

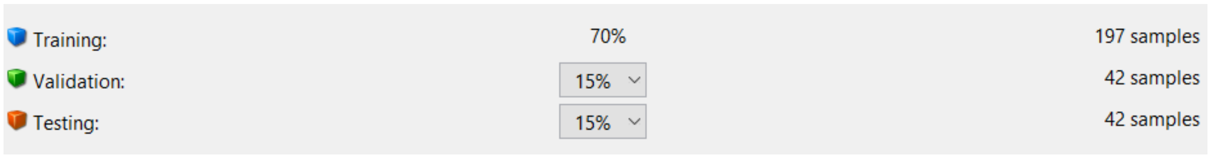
This network is then fed with the x_train and y_train previously generated for the manual implementation of the NN.


Input data to present to the network.
Inputs: x_train

Target data defining desired network output.
Targets: y_train

Figure 4: Input to NN

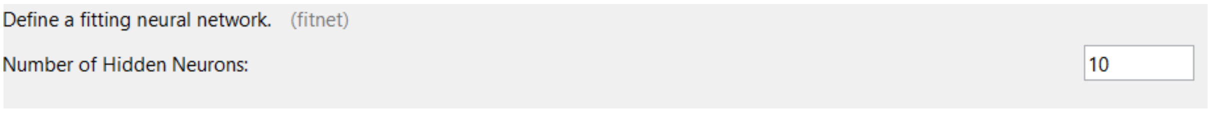
The data is then split up into training- test- and validation- set.



Training:	70%	197 samples
Validation:	15%	42 samples
Testing:	15%	42 samples

Figure 5: Training data vs Validation and Testing data ratio

For the hidden layer 10 neurons are chosen.

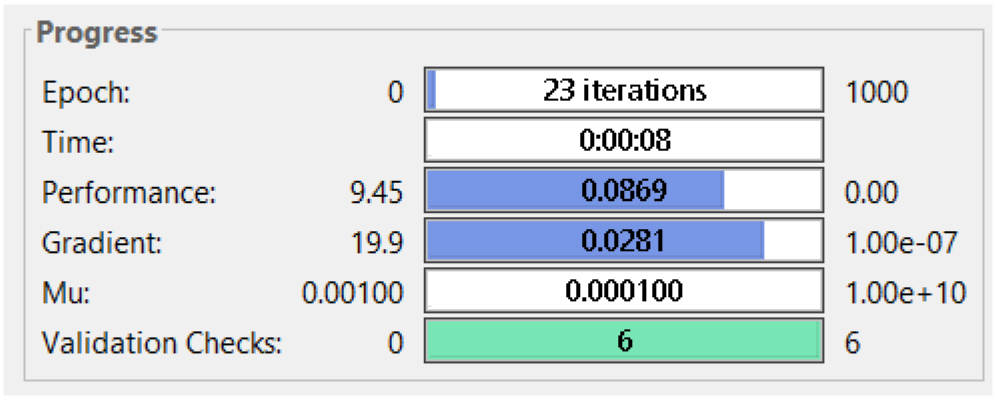

Define a fitting neural network. (fitnet)

Number of Hidden Neurons: 10

Figure 6: Hidden layer size

Results

The nntraintool automatically determines the right amount of iterations.



Progress			
Epoch:	0	23 iterations	1000
Time:		0:00:08	
Performance:	9.45	0.0869	0.00
Gradient:	19.9	0.0281	1.00e-07
Mu:	0.00100	0.000100	1.00e+10
Validation Checks:	0	6	6

Figure 7: Progress report after training

From the validation performance, it can be clearly seen, that the NN is being trained quickly. By the validation performance it is determined, that at epoch 17 the NN delivers best performance so training can be stopped.

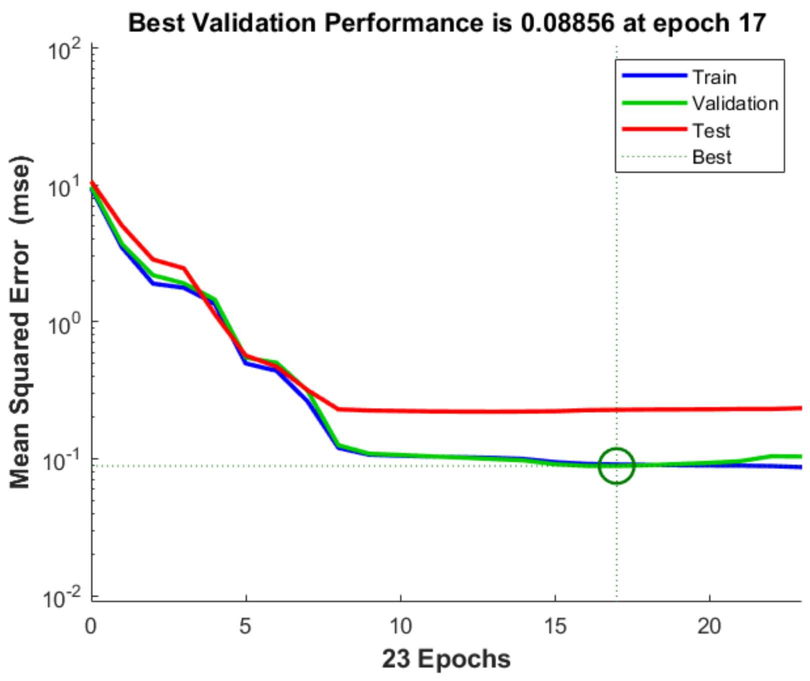


Figure 8: Validation Performance

After 17 epochs overfitting starts.

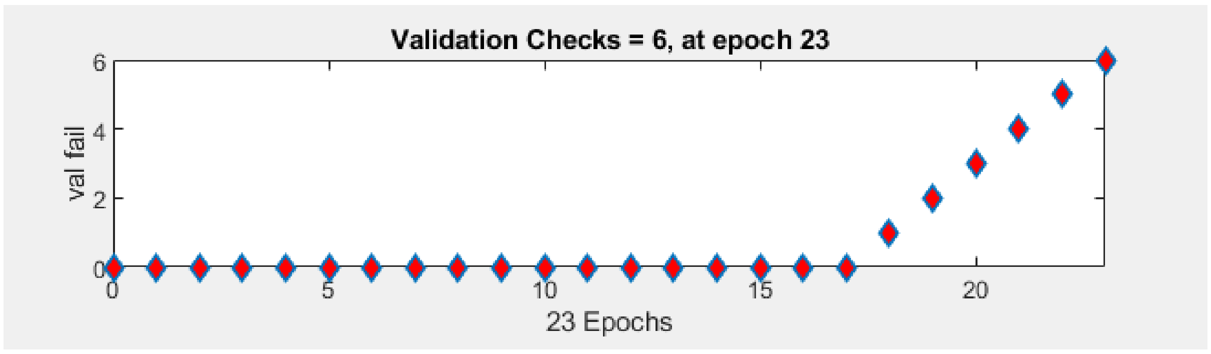


Figure 9: Validation Checks

When the output of the NN lies close to the target value, they lie on a diagonal line. The closer the datapoints are to this line, the better the fit.

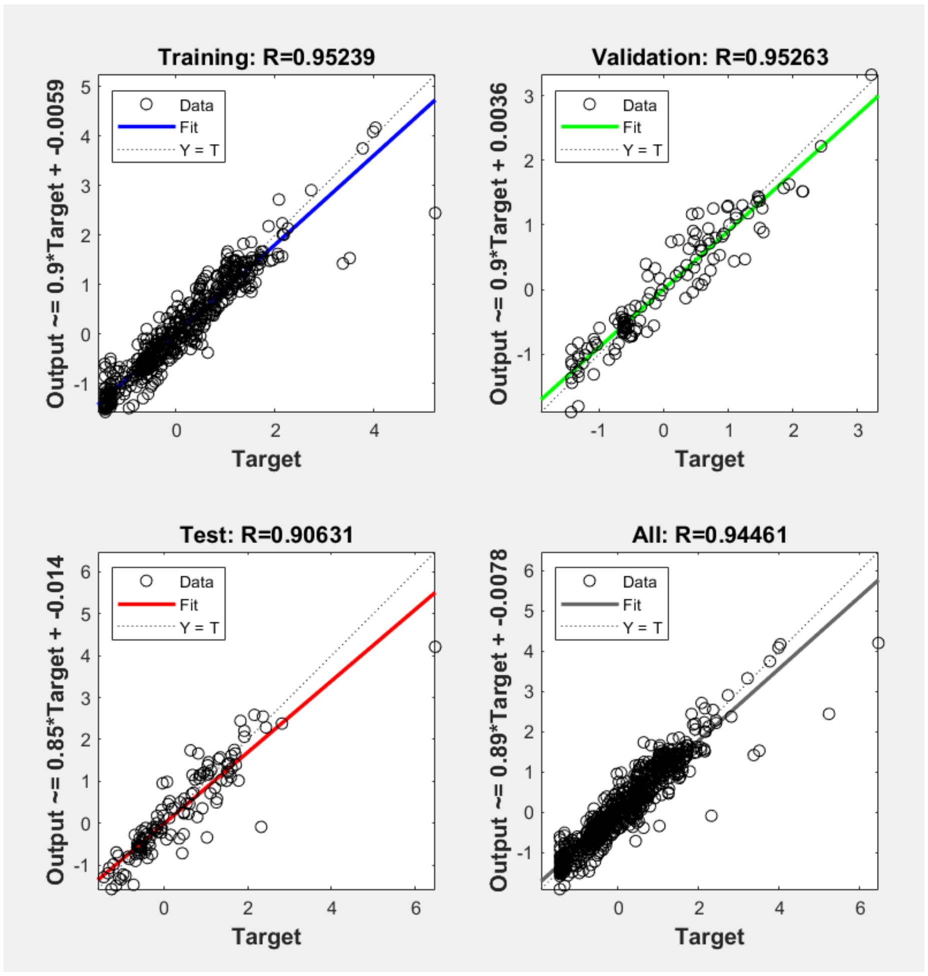
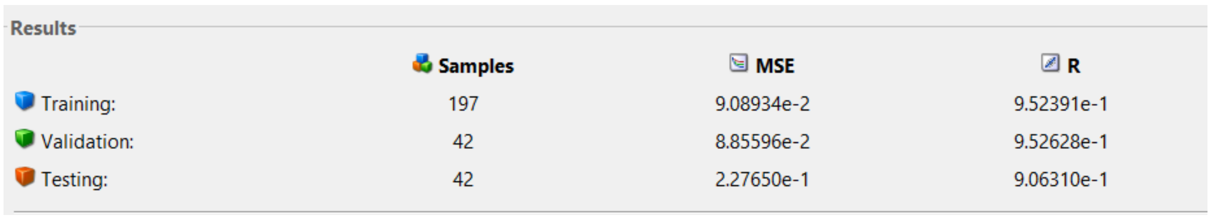


Figure 10: Regression plot

MSE (Mean Squared Error) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and what is estimated. The closer the MSE is to zero, the better the fit. As the MSE is very low here, there seems to be a good fit with the data.



Results

	Samples	MSE	R
Training:	197	9.08934e-2	9.52391e-1
Validation:	42	8.85596e-2	9.52628e-1
Testing:	42	2.27650e-1	9.06310e-1

Figure 11: Error metrics

The trained network can then be fed with the test data. A good fit can be seen here.

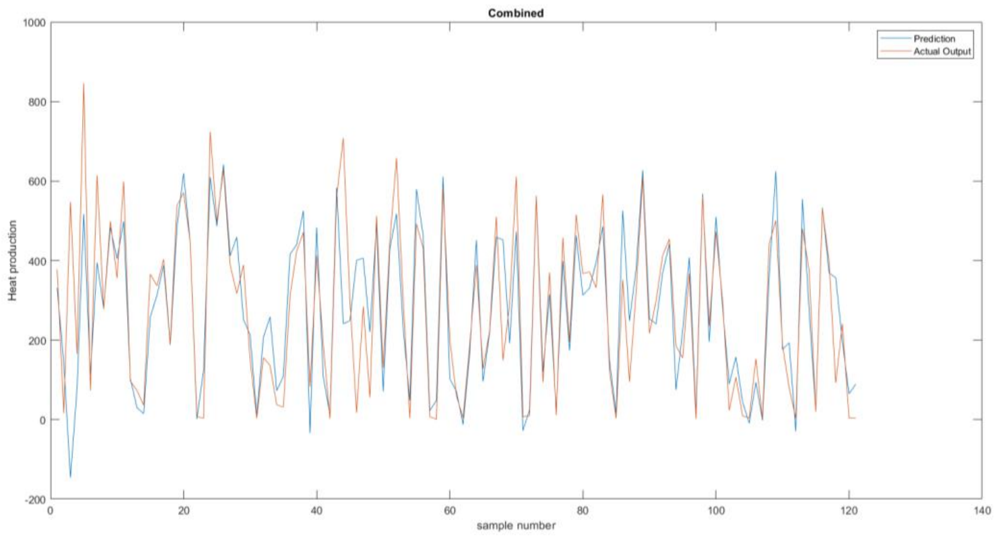


Figure 12: Combined heat production predicted vs actual

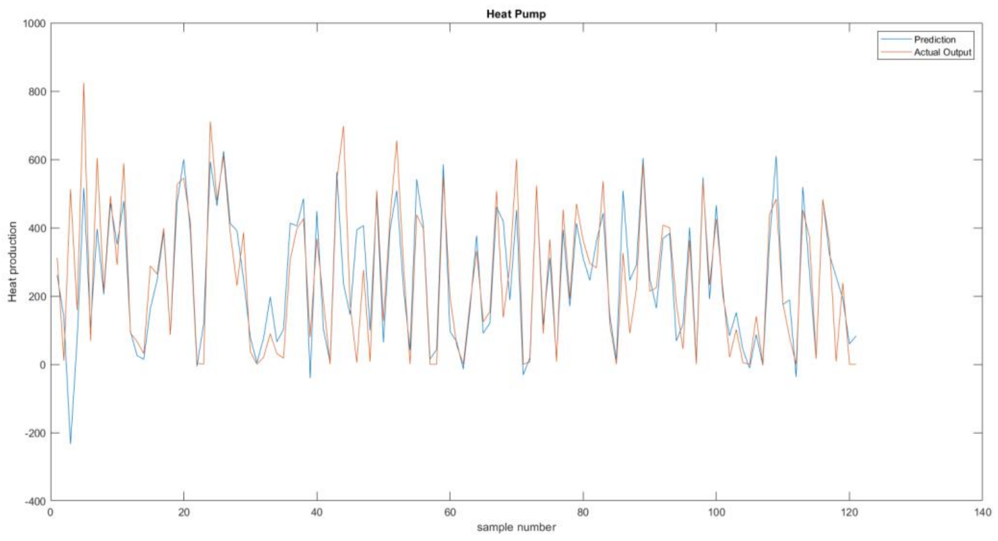


Figure 13: Heat Pump heat production predicted vs actual

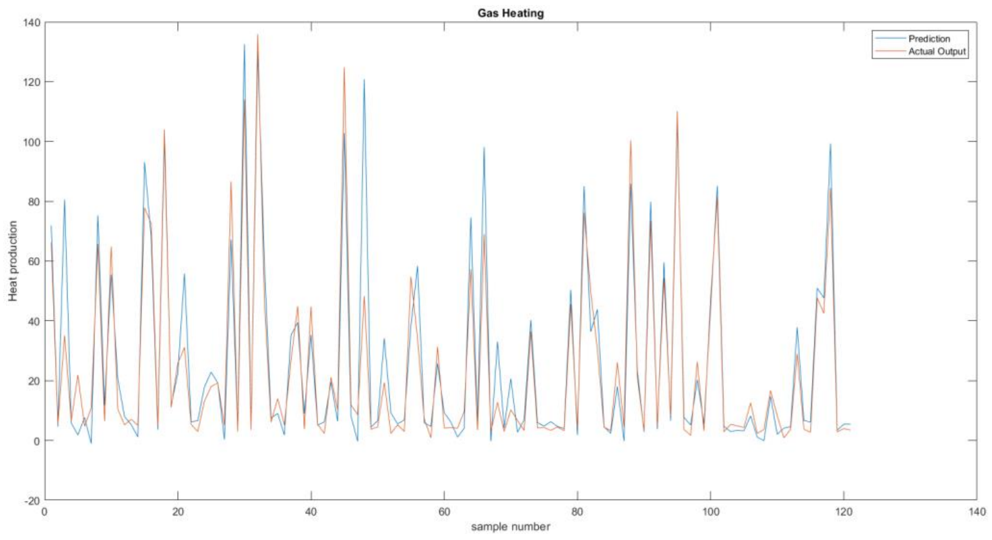


Figure 14: Gas Heating heat production predicted vs actual

Regression MATLAB toolbox

Method

The regression SVM toolbox in MATLAB was employed. Using the regression model through SVM (Gaussian kernel) function, the root mean square error (RMSE) is very small and so are the other error metrics. A 5-fold cross validation is used to validate the model.

Cross validation is also called a rotation estimation. It's very easy to see how accurately the predictive model will perform in practical settings. The data is divided into 5 portions of test data sets and training data sets for each iteration. The same test set is not used for all iterations.

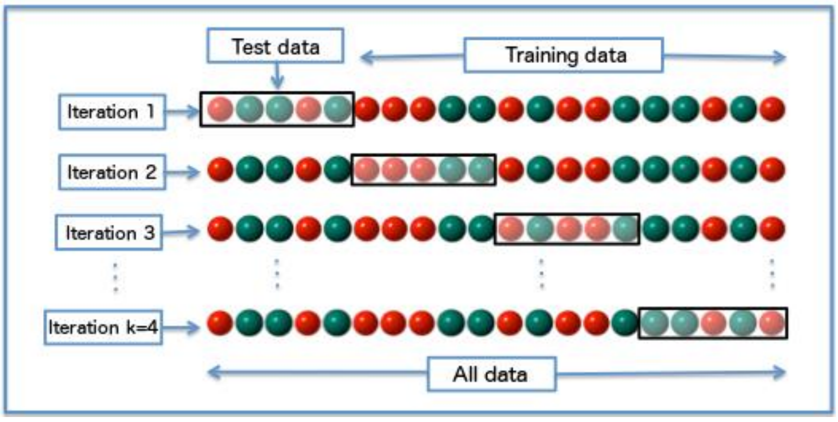
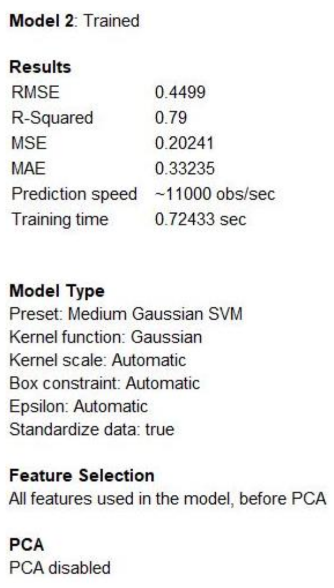


Figure 15: k-fold cross validation

Results

The error metrics RMSE, R-Squared, MSE and MAE are observed and all of them are small. This means there's less errors in the system.



Model 2: Trained

Results

RMSE	0.4499
R-Squared	0.79
MSE	0.20241
MAE	0.33235
Prediction speed	~11000 obs/sec
Training time	0.72433 sec

Model Type
Preset: Medium Gaussian SVM
Kernel function: Gaussian
Kernel scale: Automatic
Box constraint: Automatic
Epsilon: Automatic
Standardize data: true

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Figure 16: Error metrics

In the graph below, the graph corresponds to the **actual** heat production and the **predicted** heat production. It can be seen here that there is a good correlation between the output (heat demand) and the temperature (x-axis, column_1).

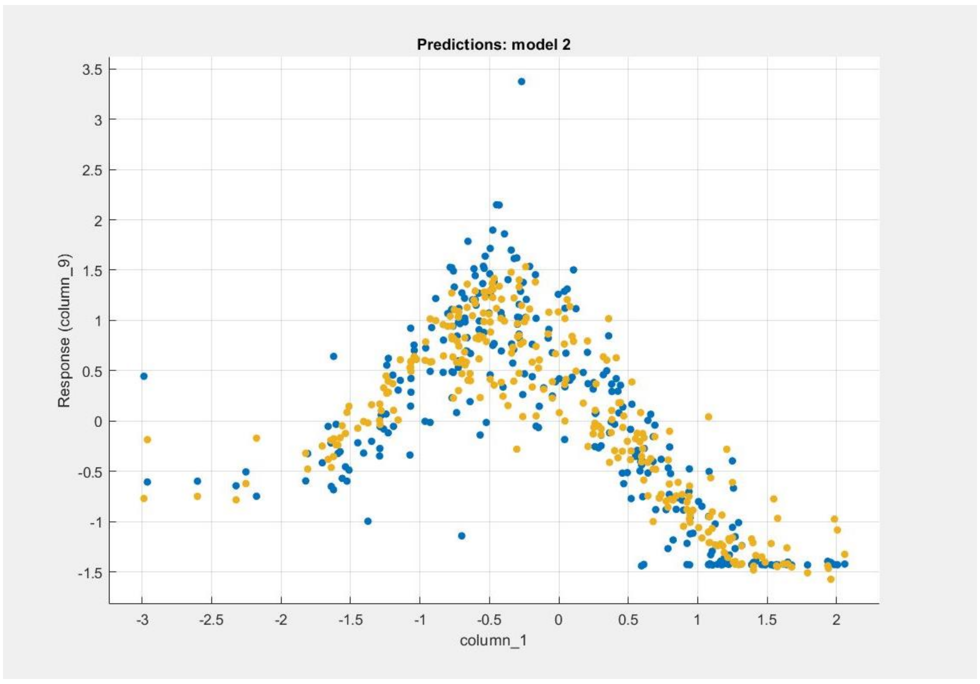


Figure 17: Predicted vs actual heat demand based on temperature

The graph below shows the fit of the actual and predicted responses. The linear graph is the ideal fit. The closer the points are to each other and the ideal line, better the fit. As can be seen, it's a good prediction model.

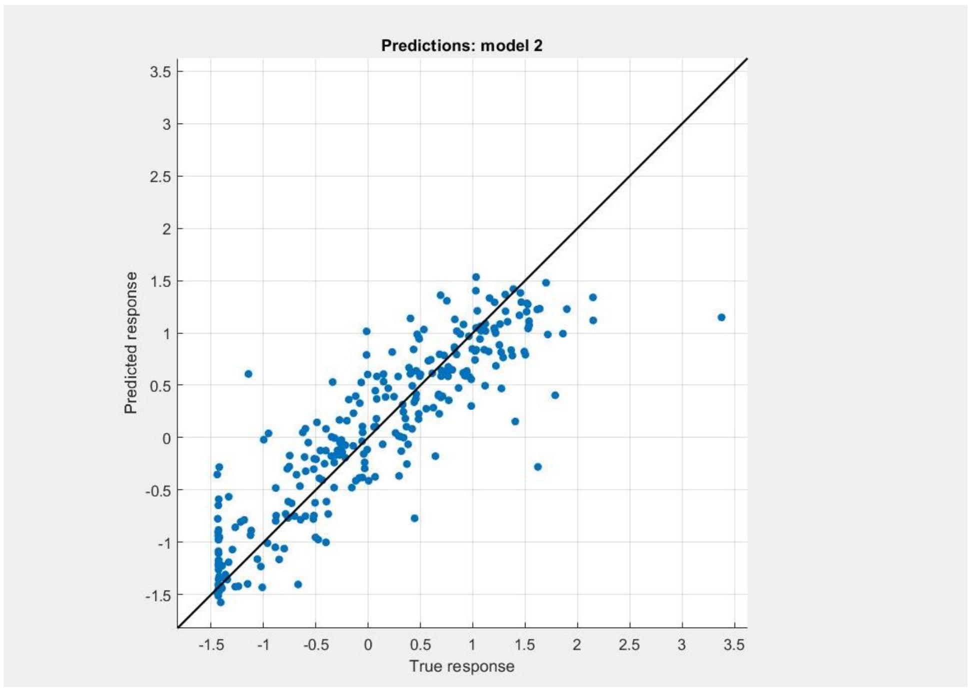


Figure 18: Fit of normalised predicted and actual outputs of heat demand

The graph below shows the residuals with respect to the predicted response. For the normalised values that range between -1.5 and 1.5, the residuals are lower for the lower end and increases with an increase in normalised values.

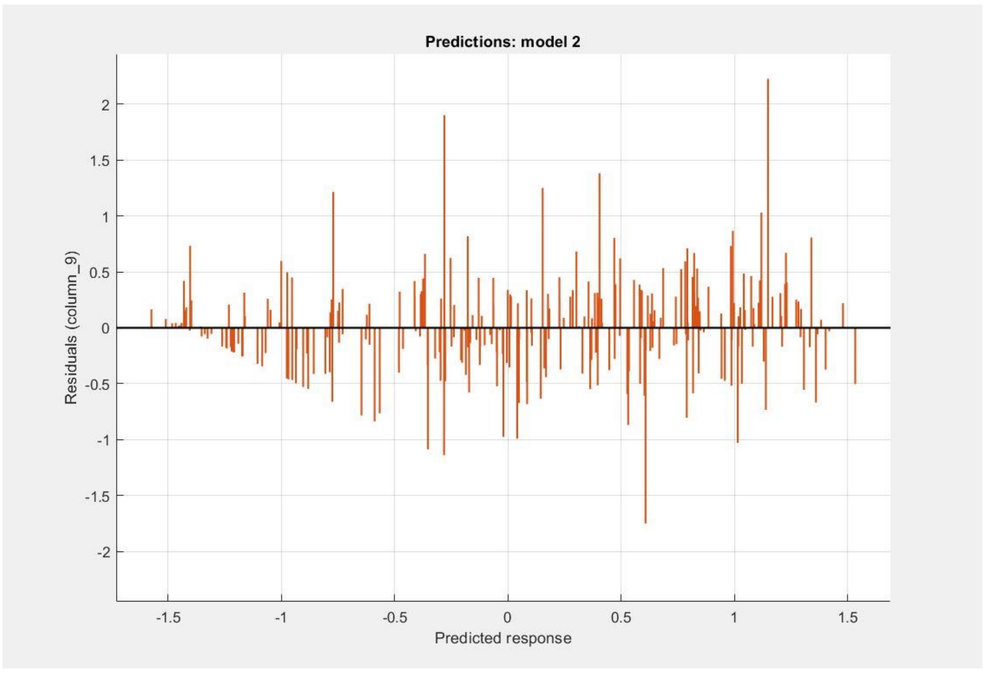


Figure 19: Residuals for predicted heat demand

The model can then be fed with the test data.

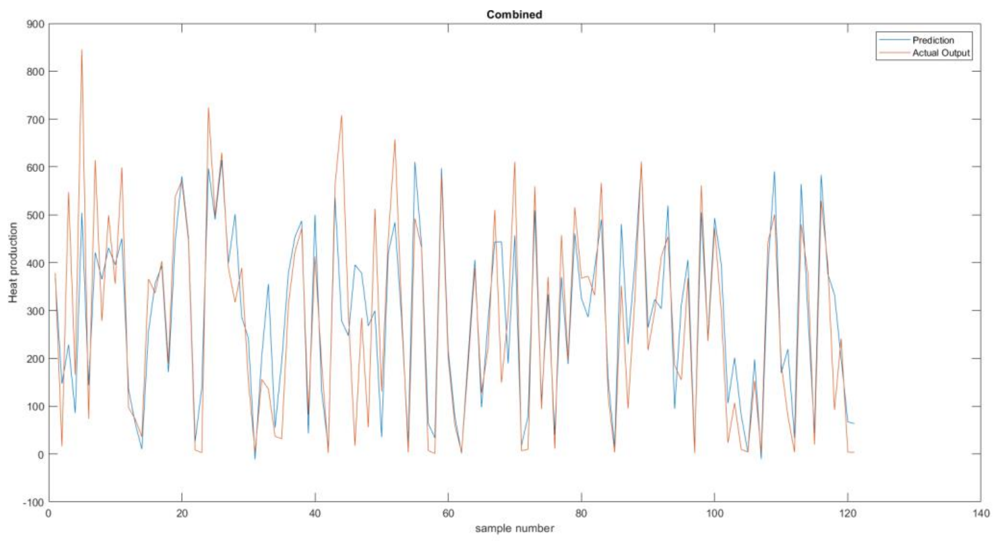


Figure 20: SVM combined heat production predicted vs actual output

Conclusions

Manual implementation of NN

The Manual implementation of the NN did not manage to train on the given dataset. As the NN had already proven to be implemented correctly, it was assumed, that the heating demand cannot be predicted from this weather data with this type of NN.

Next steps would have been using a different algorithm like linear regression or a NN with more hidden layers.

Another way to get usable results that was proposed was the use of a better dataset. This dataset was given for a single household with only one person living there. The heat demand will be influenced a lot by different daily routines of this person. Gathering a dataset from a bigger house with several apartments would give a more averaged view and cancel out the individual heat demands of each apartment.

Also feeding the NN with daily values could improve on the results, as the weather data is available even hourly. Together with that improvement, the day of the year could also be given to the NN, so it gets a clue about different times of the year.

A final way to improve the dataset would be gathering actual heat consumption data and splitting these up between heating and hot water demand. This would allow to correlate the actual heat demand with the weather and minimize the noise introduced by the hot water demand.

NN toolbox

With the NN toolbox a good prediction for the heat production from weather data could be made. This shows, that the manual implementation of the NN was not done correctly. The cause for the discrepancy could not be found though.

A reason for that discrepancy could be the difference in the output activation. While the manual implementation of the NN uses a sigmoid activation function in the output layer, the NN toolbox uses a linear activation function.

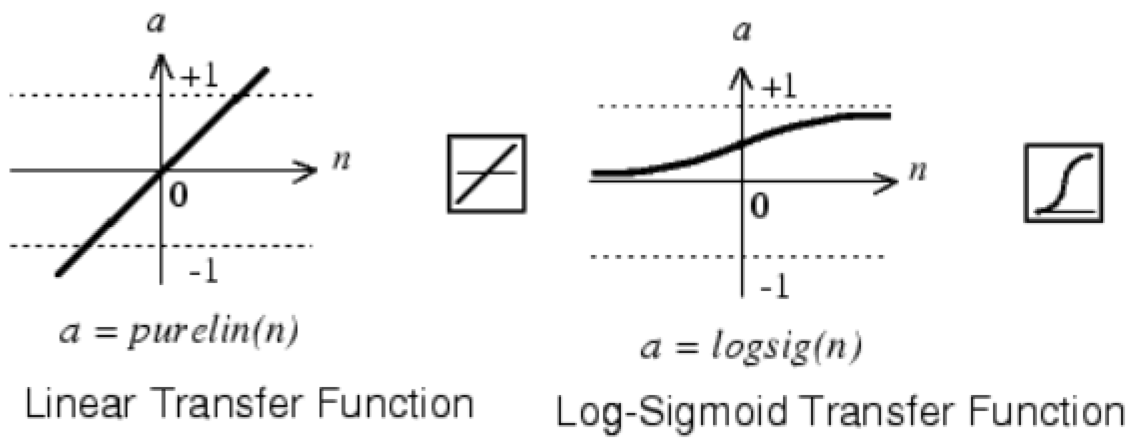


Figure 21: Linear vs Sigmoid activation function

SVM with Regression toolbox

The SVM with the regression toolbox can find a correlation in the data, especially with the temperature input. The MSE for the test sets for both SVM and the NN from the toolbox are very similar (0.20 (SVM) and 0.23 (NN)) , so either one or both techniques could be used for prediction of heat demand.

With this toolbox, for the current data set, with smaller predicted values, the errors are smaller.

References

Instituut, K. N. (2019, 06 15). *Uurgegevens van het weer in Nederland - Download*. Opgehaald van Klimatologie: http://projects.knmi.nl/klimatologie/uurgegevens/selectie.cgi

Nguyen, A. (2019, 06 27). *Coursera*. Opgehaald van Machine Learning: https://www.coursera.org/learn/machine-learning/

Nguyen, A. (2019, 06 17). *Coursera*. Opgehaald van Machine Learning: https://www.coursera.org/learn/machine-learning/

Richárd, N. (2019, 06 17). *Towards Data Science*. Opgehaald van The differences between Artificial and Biological Neural Networks: https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

Wikipedia. (2019, 06 16). *Artifical Neural Networks*. Opgehaald van Wikipedia: https://en.wikipedia.org/wiki/Artificial_neural_network

Wikipedia. (2019, 06 16). *Machine Learning*. Opgehaald van Wikipedia: https://en.wikipedia.org/wiki/Machine_learning

Wikipedia. (2019, 06 16). *Predictive Analytics*. Opgehaald van Wikipedia: https://en.wikipedia.org/wiki/Predictive_analytics