**Assignment 2**

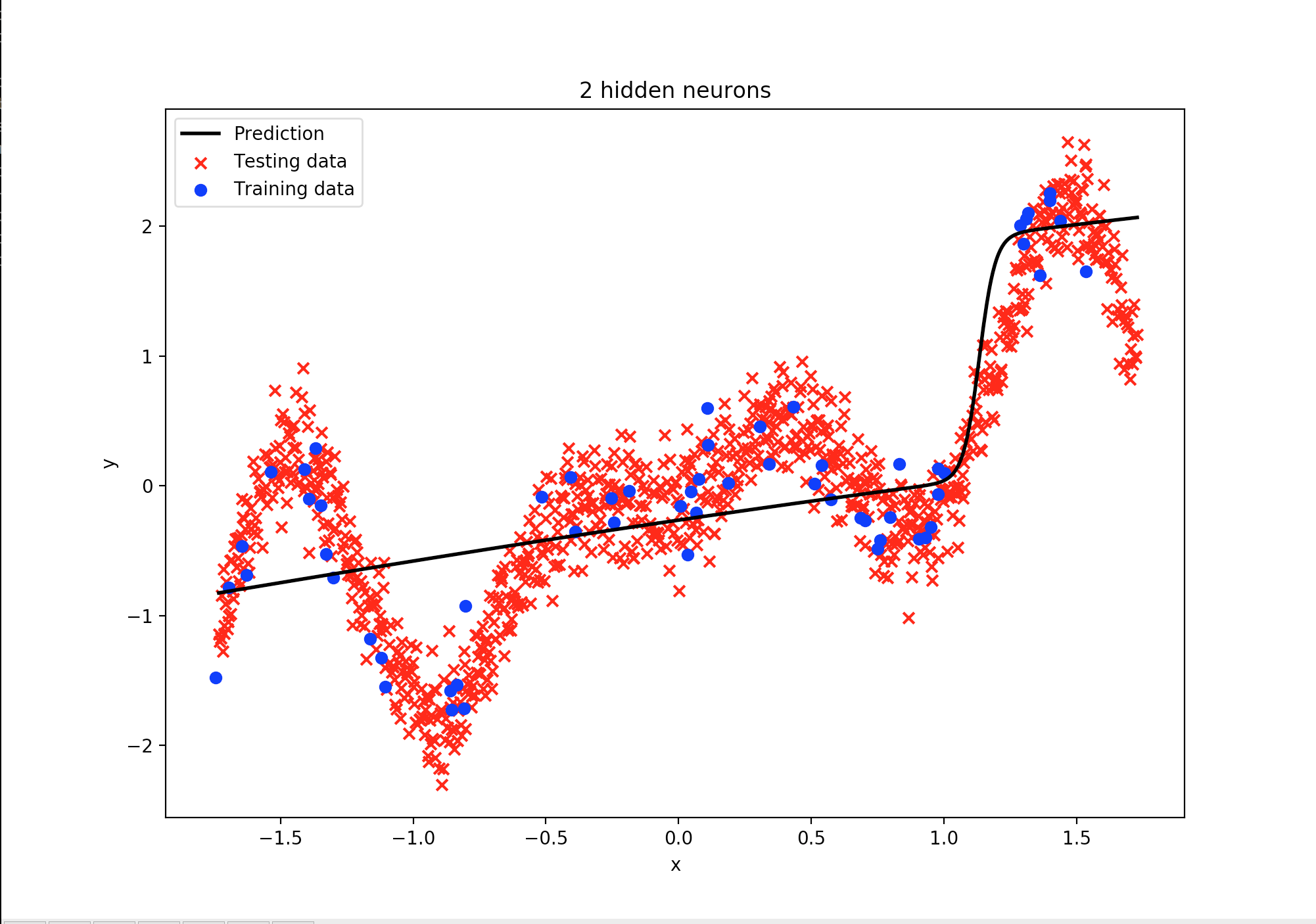
# Computational Intelligence SEW, SS2017

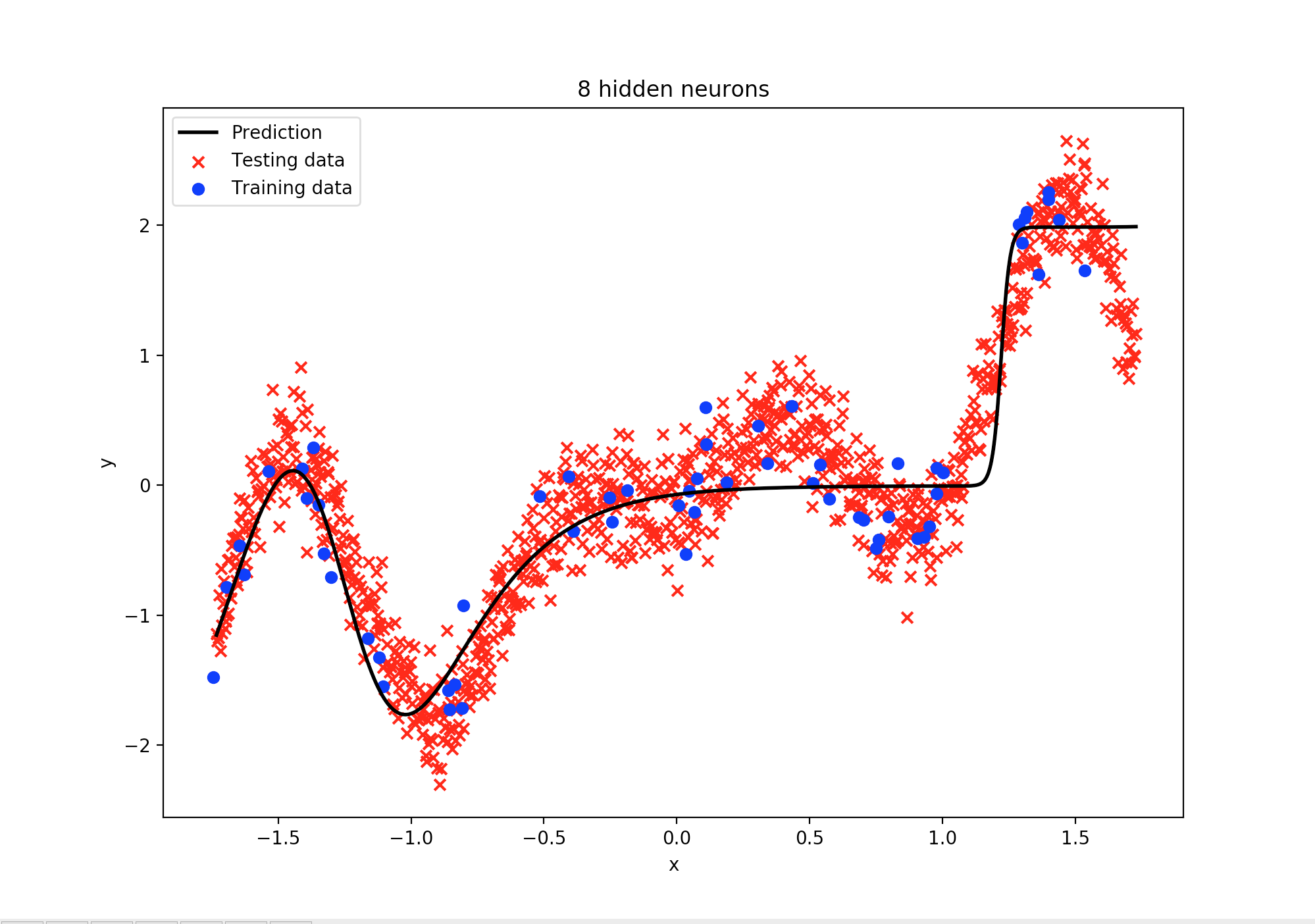
|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| Last name | First name | Matriculation Number |
| Papst | Stefan | 1430868 |
| Guggi | Simon | 1430534 |
| Perkonigg | Michelle | 1430153 |

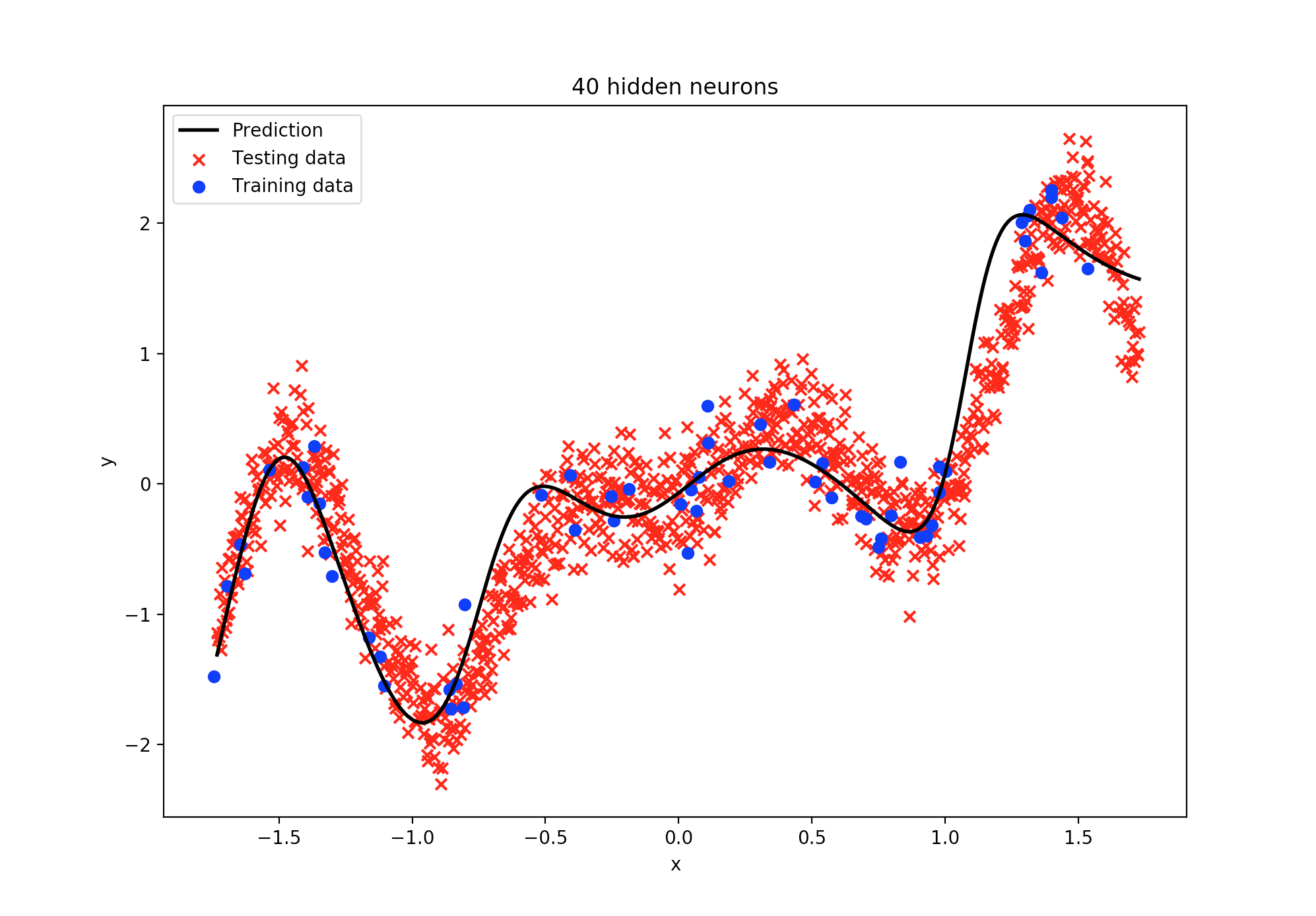
1.1 Simple Regression with Neural Networks

* a)  Learned function
* nh = 2, nh = 8 and nh = 40

As you can see in the plot with 2 hidden neurons that under fitting is an existing problem in neural networks. The training set is fitted with a too “uncomplex” function, which results in high errors in the training and testing data. 8 hidden neurons are a better choice, but 40 one fitting the training data as well as the testing data. Overfitting could also happen, for example there are too much hidden neurons, which fit the training set very well, but the testing one not.







* b)  Variability of the performance of deep neural networks

**Here are the minimum, maximum, mean and standard deviation of the mean square error we obtained on the training set for a sample run:**

min\_train: 0.0517864946589

max\_train: 0.102915924071

mean\_train: 0.0714639291011

std\_train: 0.0183697649393

**Is the min MSE obtained for the same seed on the training and on the testing set ?**

No, it is a different one.

**Explain why you would need a validation set to choose the best seed ?**

With the validation data you usually try to find the best performing approach after training the neural network with the training data. In our case this would be the best seed.

**Unlike with linear-regression and logistic regression, even if the algorithm converged the variability of the MSE across seeds is expected. Why ?**

The seed sets the initial weight values of the neural network. Although the algorithm converges and there is almost the same overall result, the individual weights may differ from each other, because the weights are updated according to their values. This gives us a variability of the MSE for different seeds.

**What is the source of randomness introduced by Stochastic Gradient Descent (SGD) ?**

It is a property which was introduced by SDG to escape local minima to find better minima.

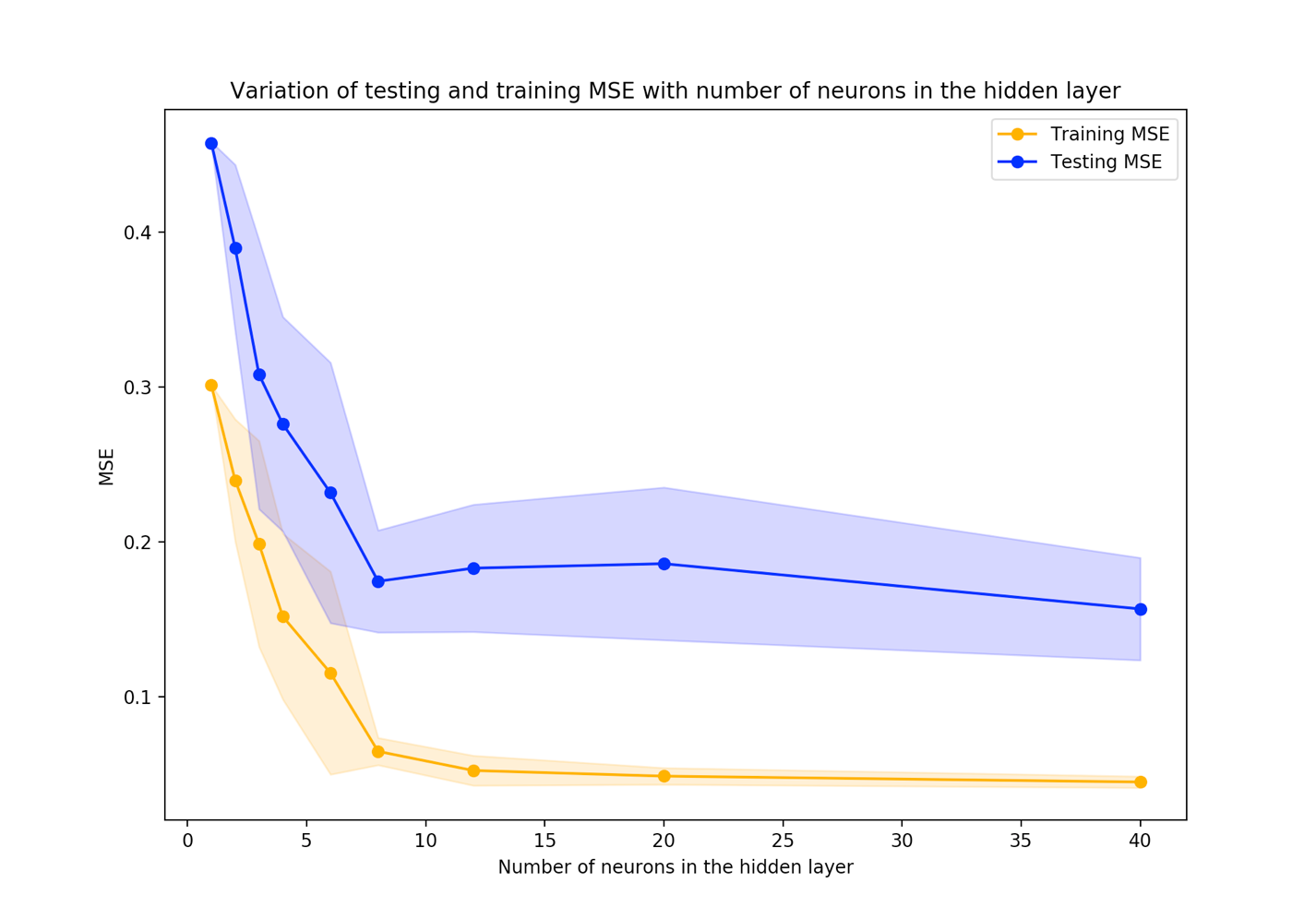
**What source of randomness will persist if SGD is replaced by standard Gradient Descent ?**

The way how the weights are initialized.

c) Varying the number of hidden neurons:

**What is the best value of n\_h independently of the choice of the random seed ?**

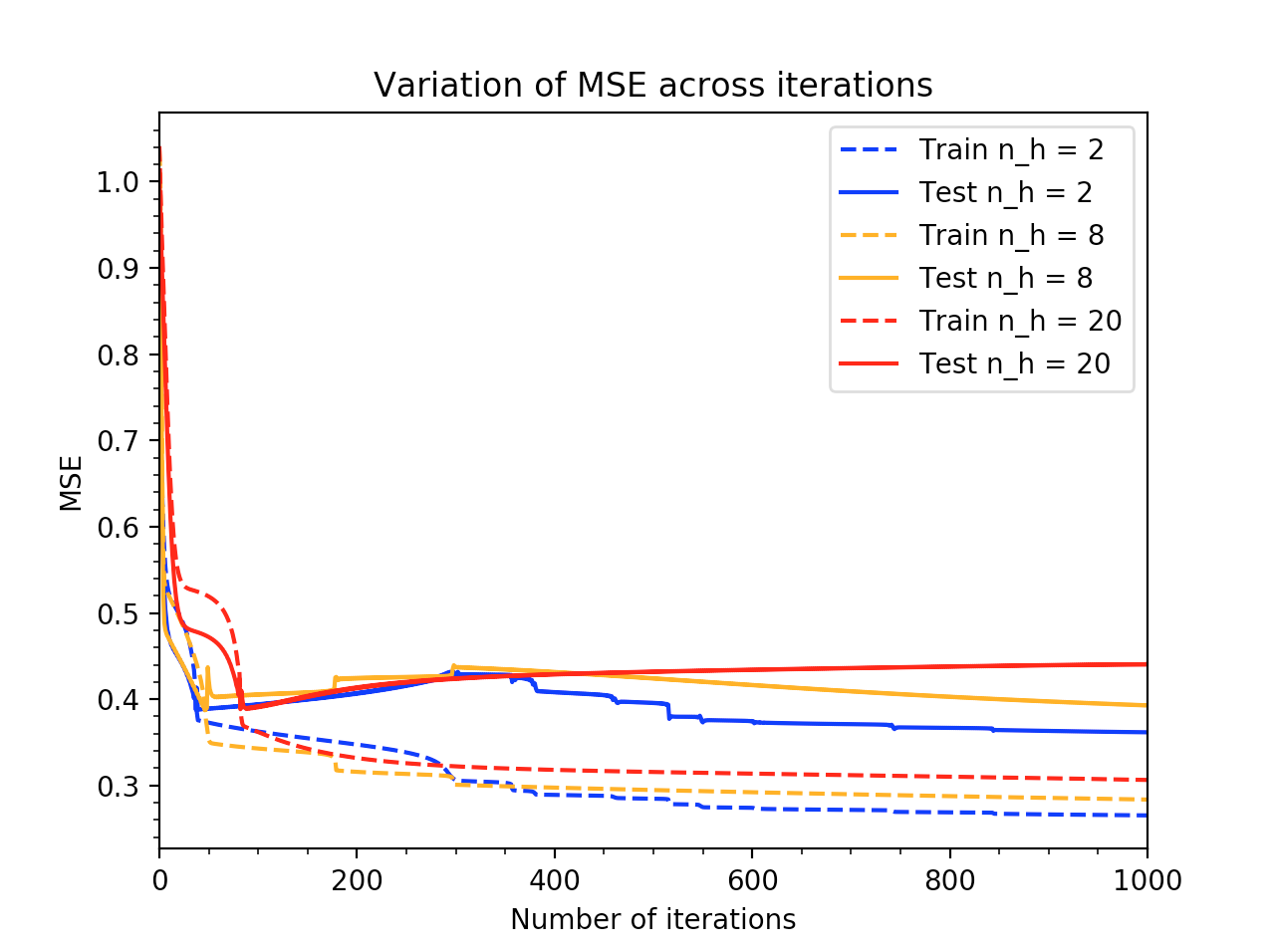
8



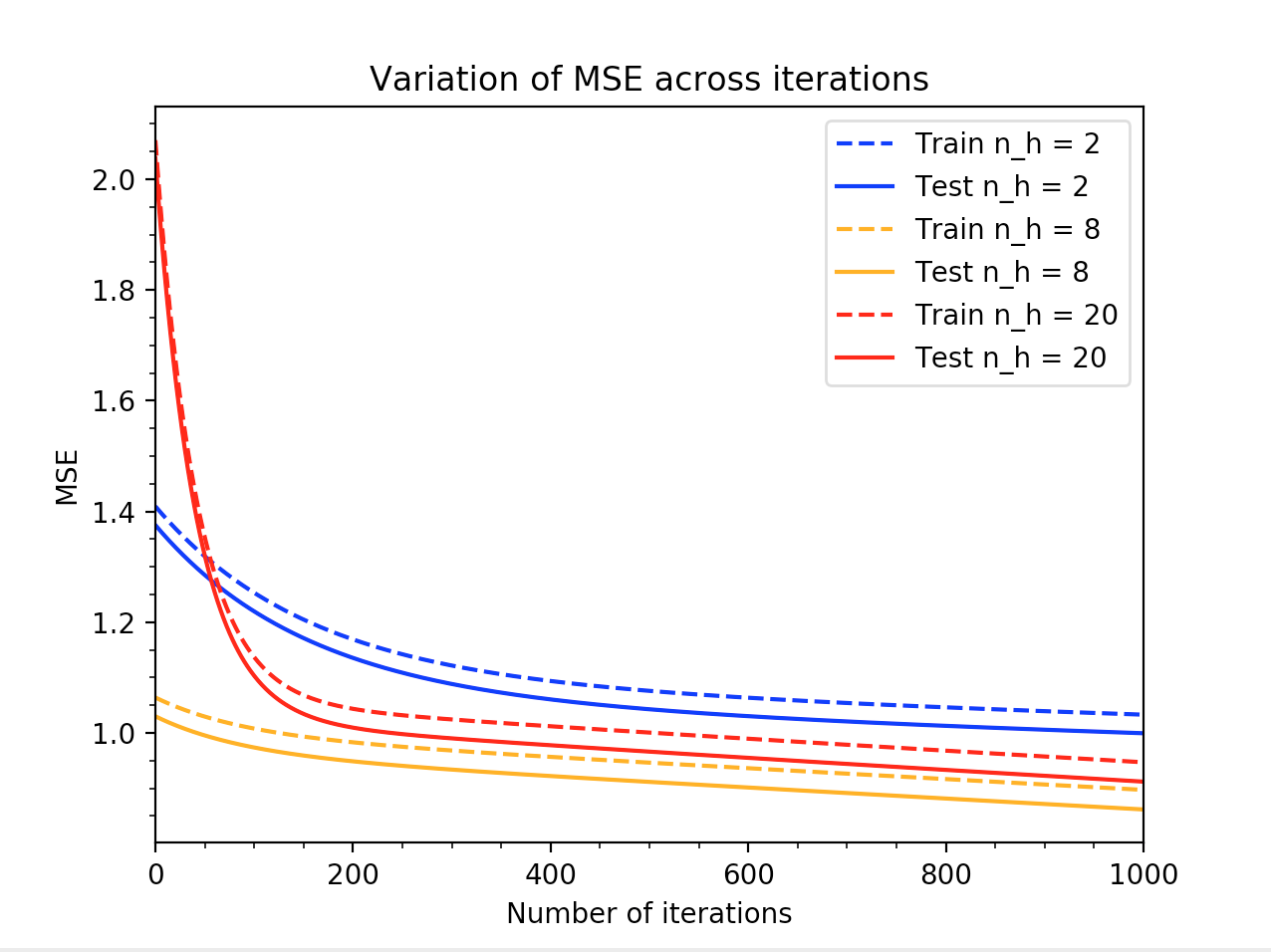
Obviously 8 is the best value for neurons in the hidden layer, so everything above is called over fitting. The training MSE is getting lower, but the training MSE doesn’t. With 8+ neurons the neural network is over fitting the training set, which is a drawback for further using with testing sets. In the other hand values under 8 would cause in under fitting. This does mean that the neural network is not trained complex enough to fit either the training set nor the testing set very well.

d) Variations of MSE during training:

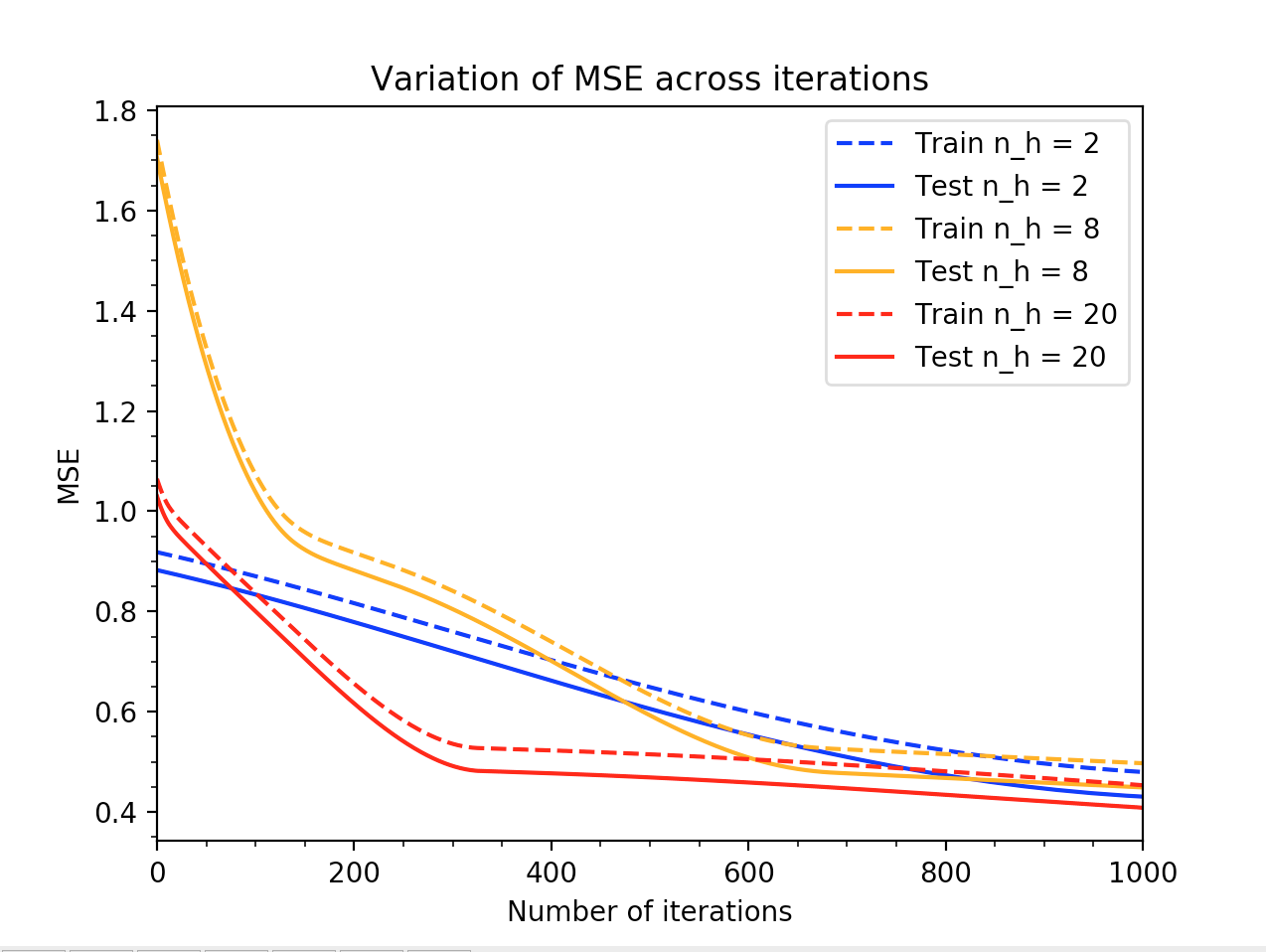
solver: “lbfgs”



solver: “sgd”



solver: “adam”



**Is the risk of overfitting increasing or decreasing with the number of hidden neurons ?**

Increasing.

**adam’ is a variant of ‘sgd’ and both are first order methods (the parameter updates are based on the gradient only), whereas ‘lbfgs’ is a second order method (the updates are also based on the Hessian). Which methods seem to perform best in this problem ?**

The first order methods seem to be more stable, but the second order method getting faster to a lower MSE. For this problem, we would say that the “adam” solver fits the requirements the best.

**What feature of stochastic gradient descent helps to overcome over fitting ?**

Early-stopping.

**The neural network is rather small as compared to what is used is real-life problems, according to your analysis which solver will be more appropriate when the number of neurons increases ?**

“Adam” would be the best choice, because it is a first order method and do not to calculate the hessian, which lasts very long for big networks.