**Assignment 2**

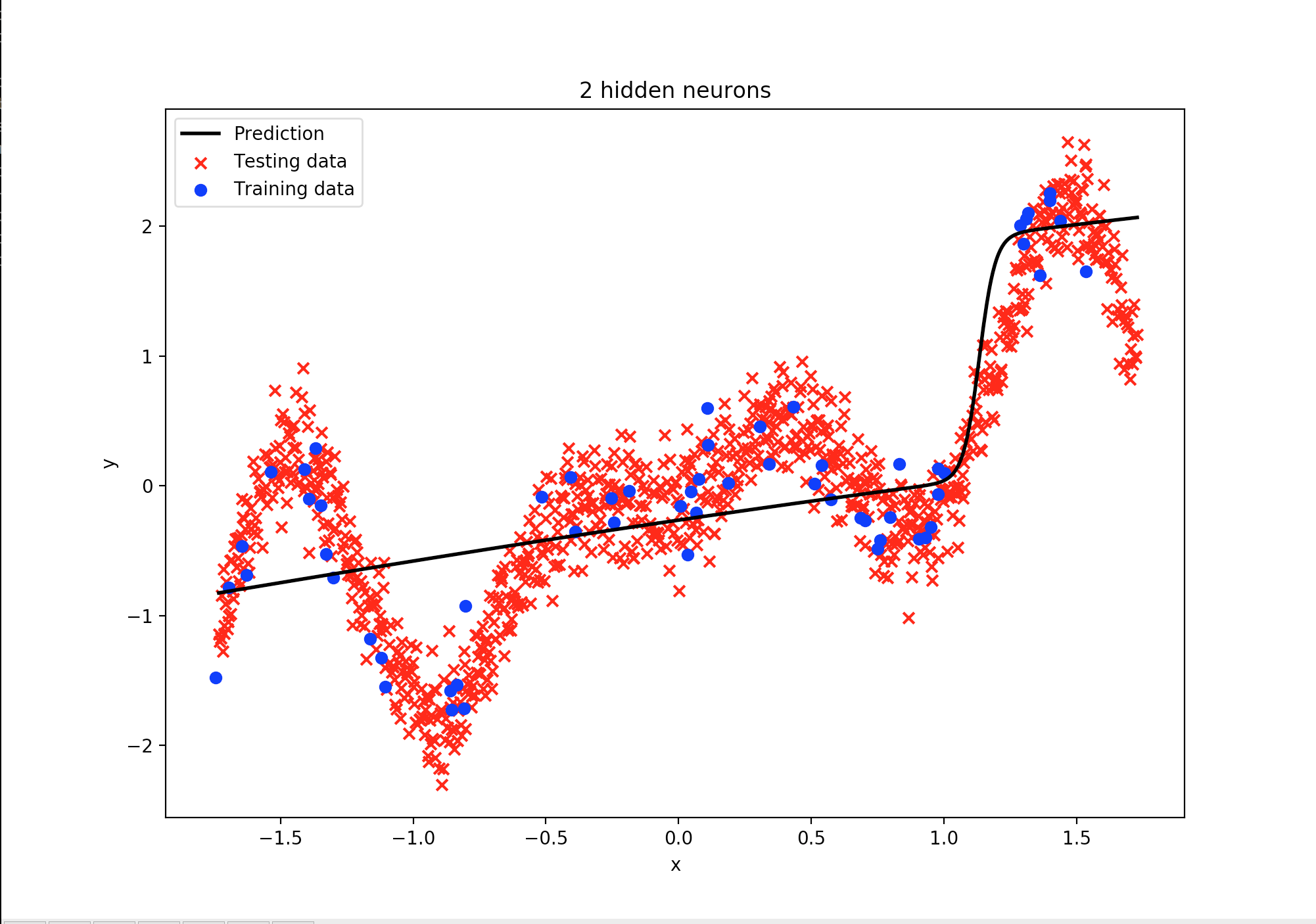
# Computational Intelligence SEW, SS2017

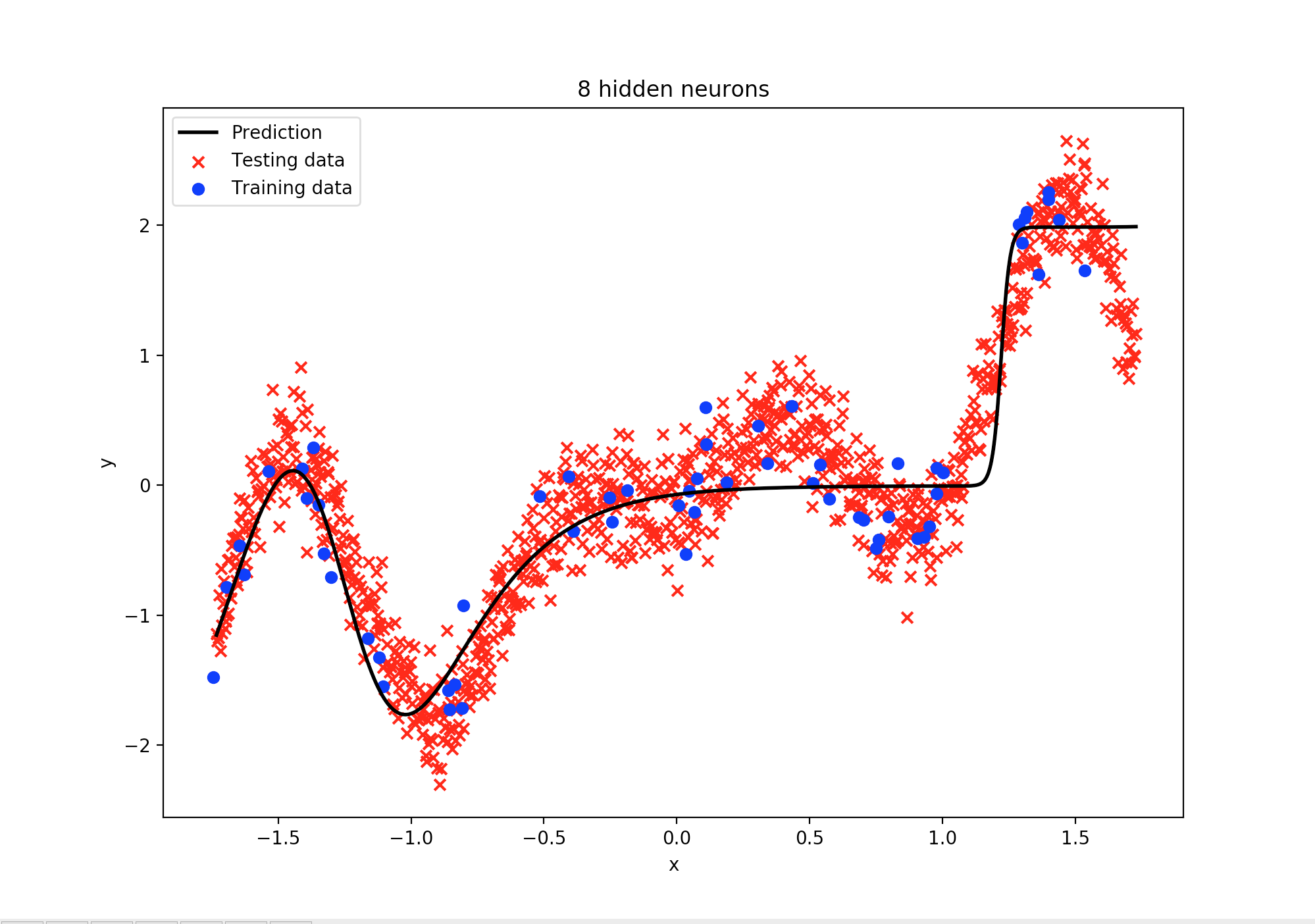
|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| Last name | First name | Matriculation Number |
| Papst | Stefan | 1430868 |
| Guggi | Simon | 1430534 |
| Perkonigg | Michelle | 1430153 |

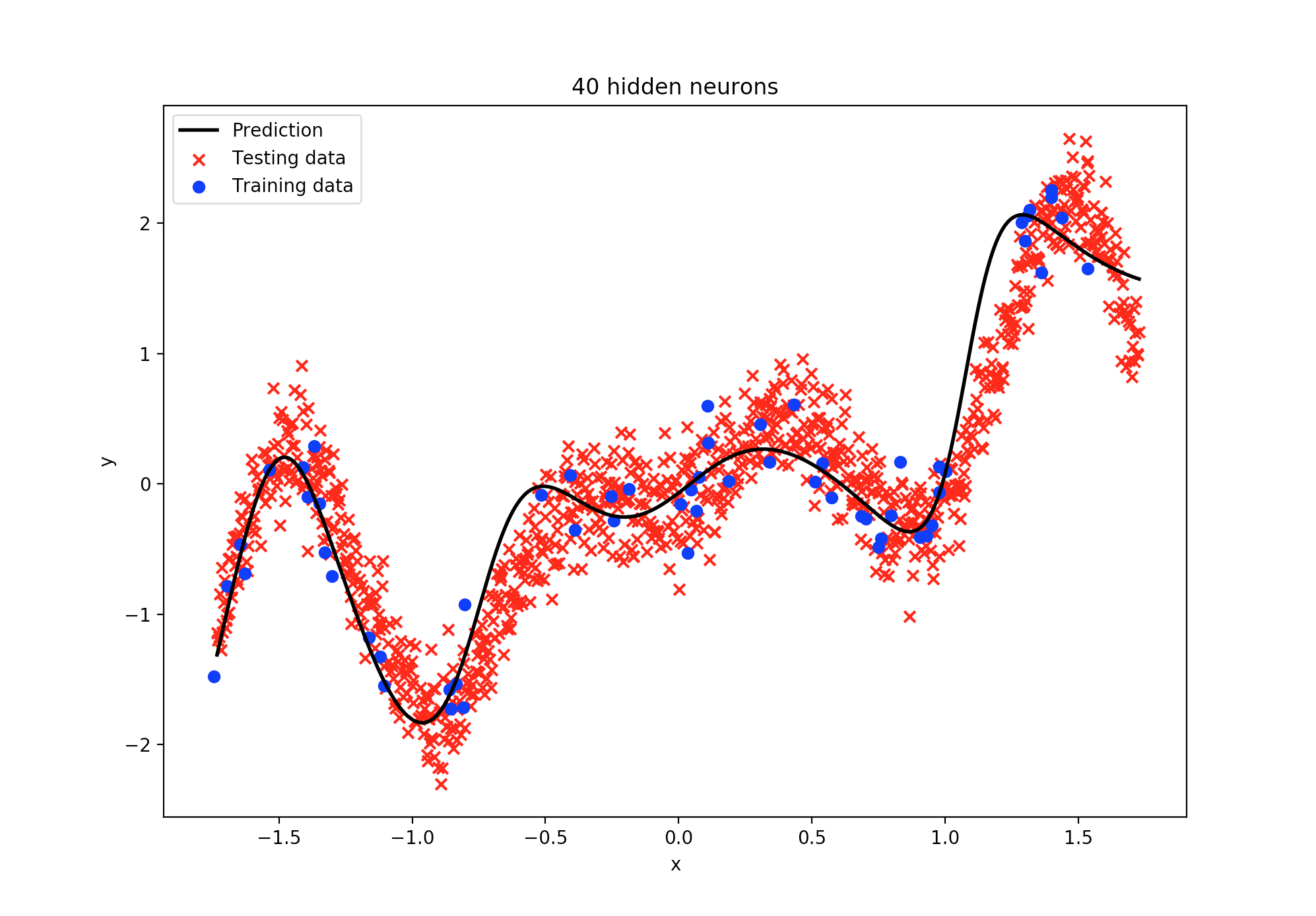
1.1 Simple Regression with Neural Networks

* 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.

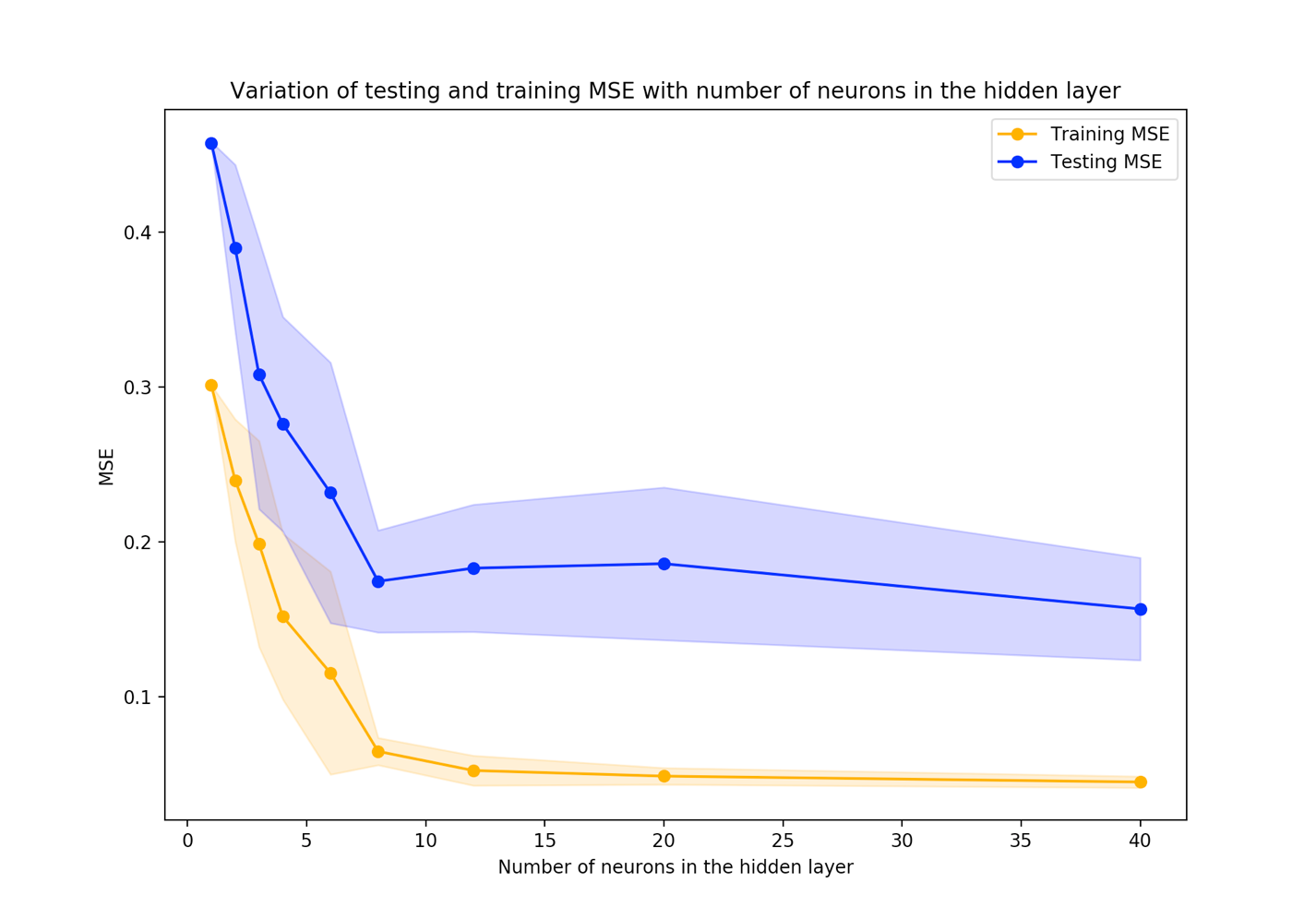
**What source of randomness will persist if SGD is replaced by standard Gradient Descent ?**

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 ?**

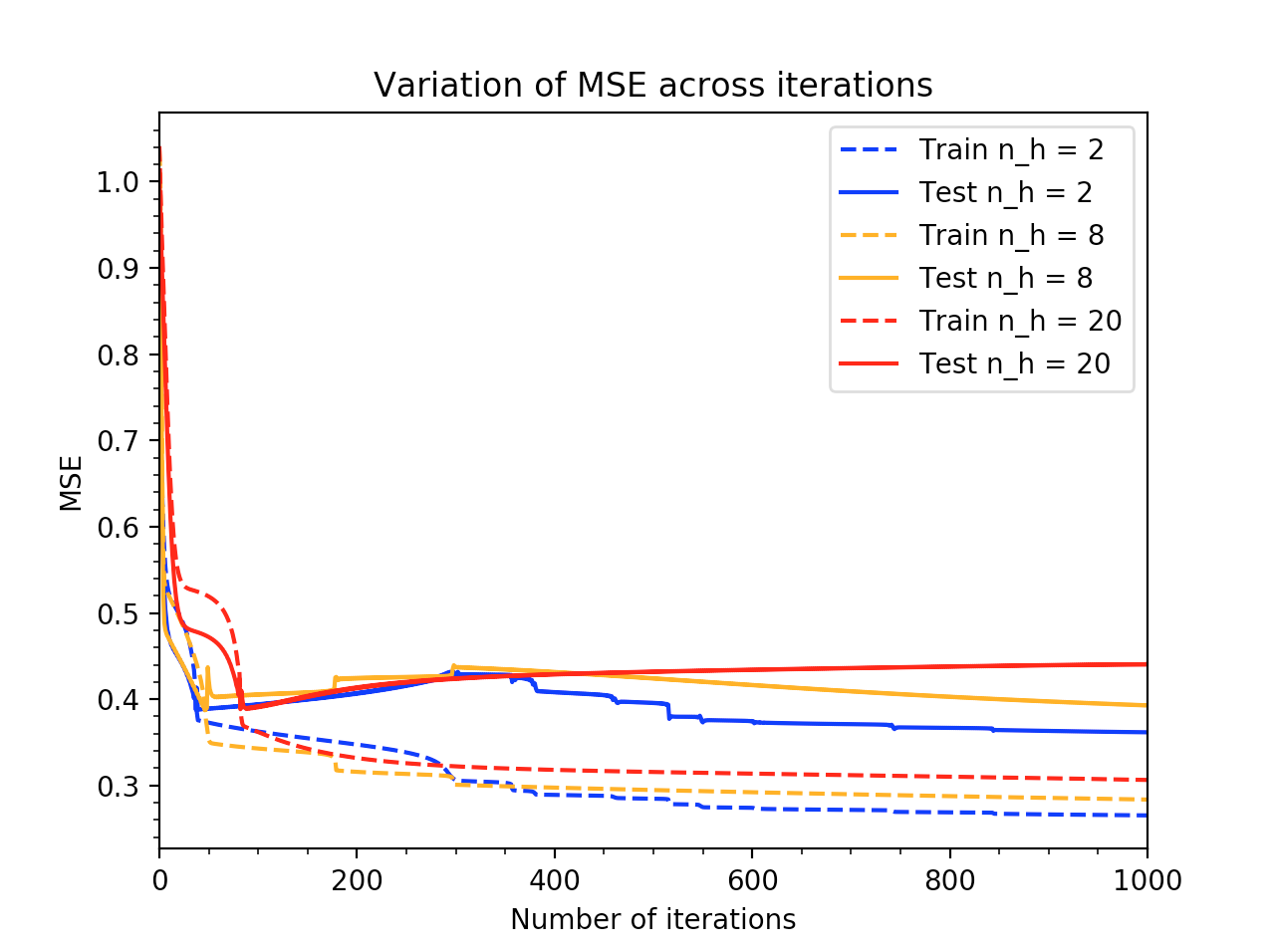
8



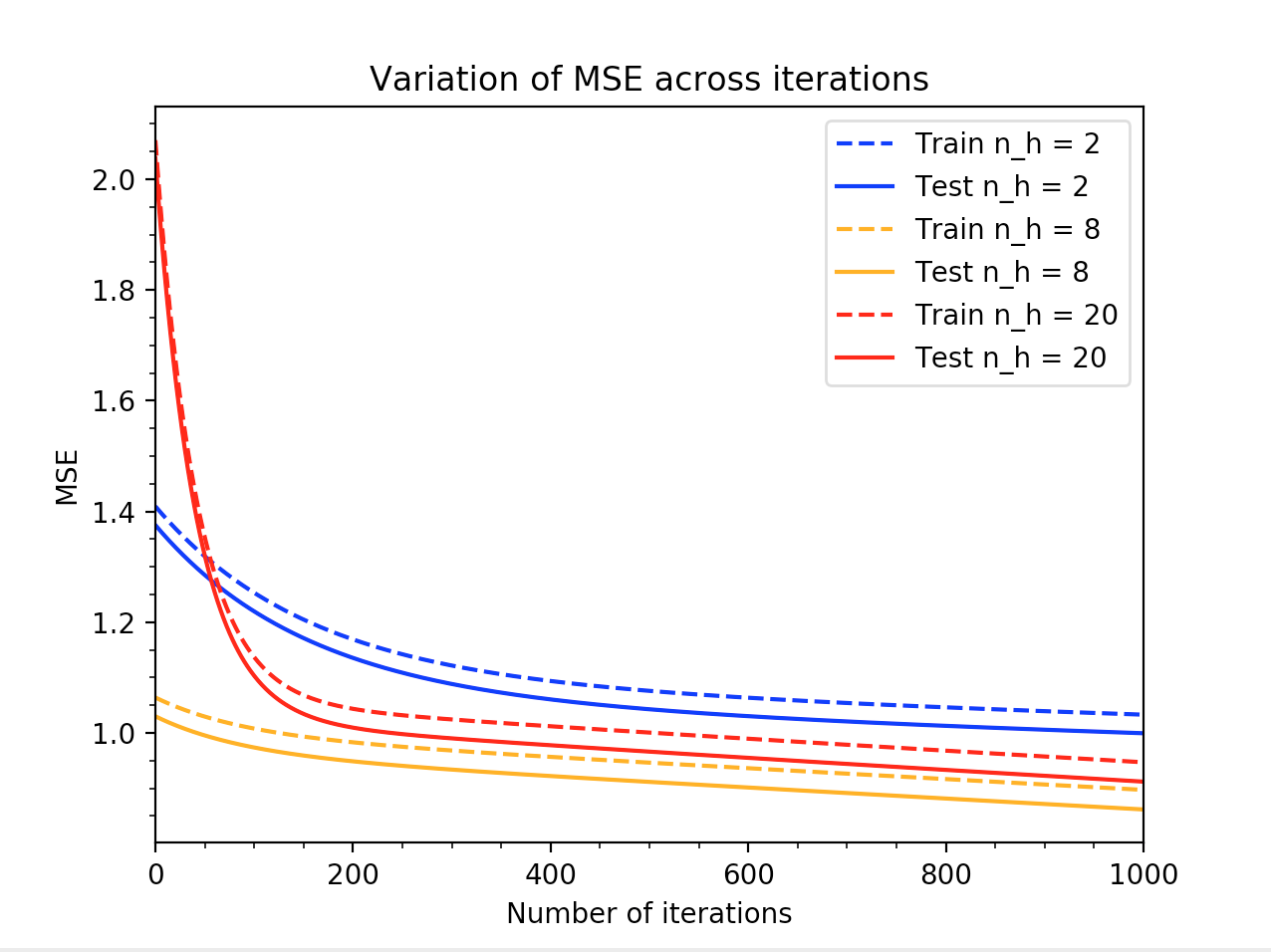
Obviously 8 is the best value for neurons in the hidden layer, so everything above is called over fitting. The training MSE is getting lower, but the training MSE doesn’t. With 8+ neurons the neural network is over fitting the training set, which is a drawback for further using with testing sets. In 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:

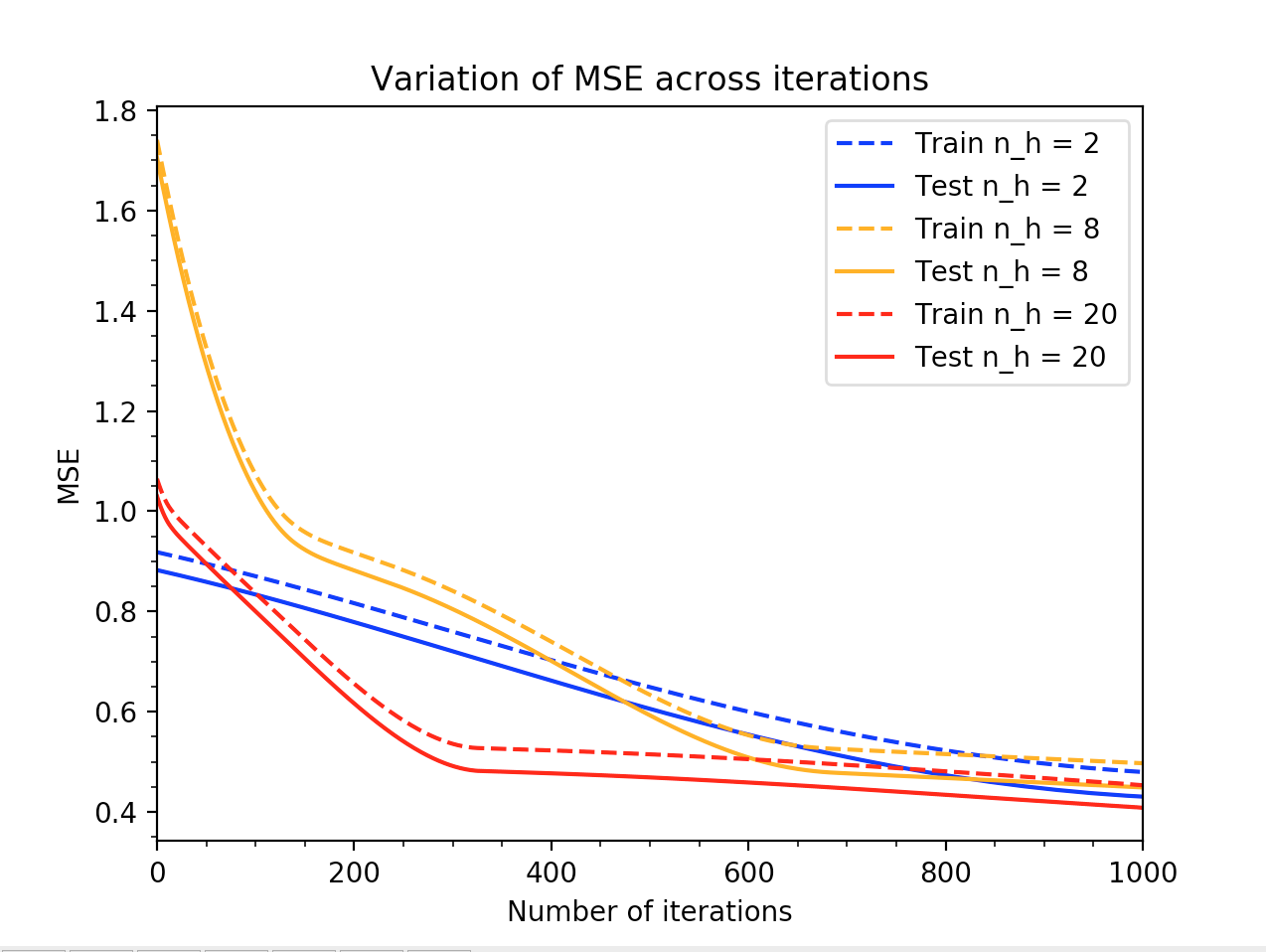
solver: “lbfgs”



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.

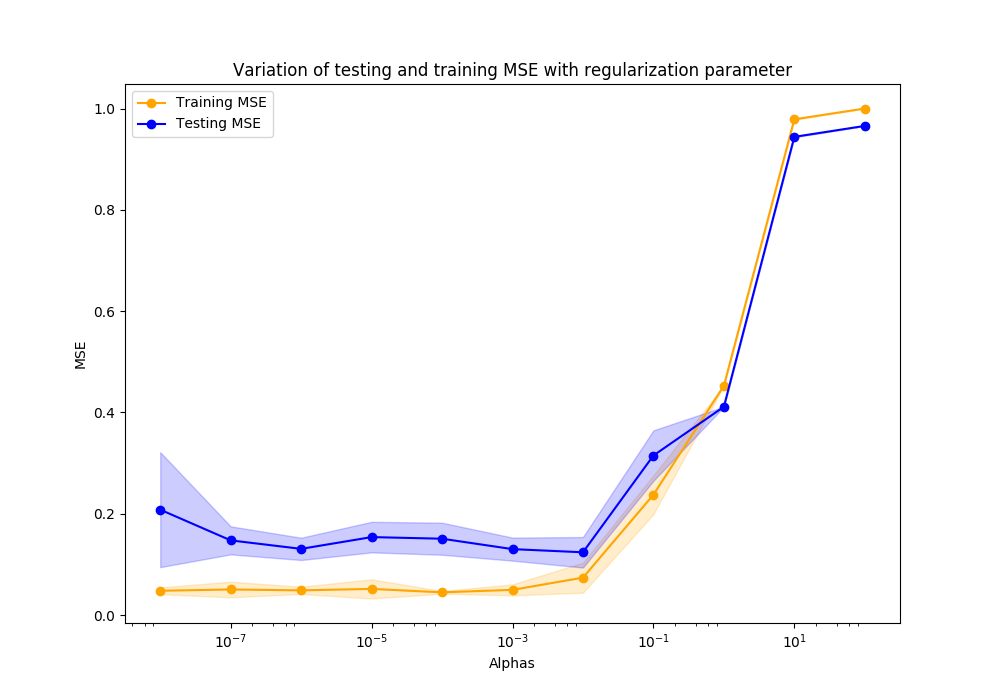
**The neural network is rather small as compared to what is used is real-life problems, according to your analysis which solver will be more appropriate when the number of neurons increases ?**

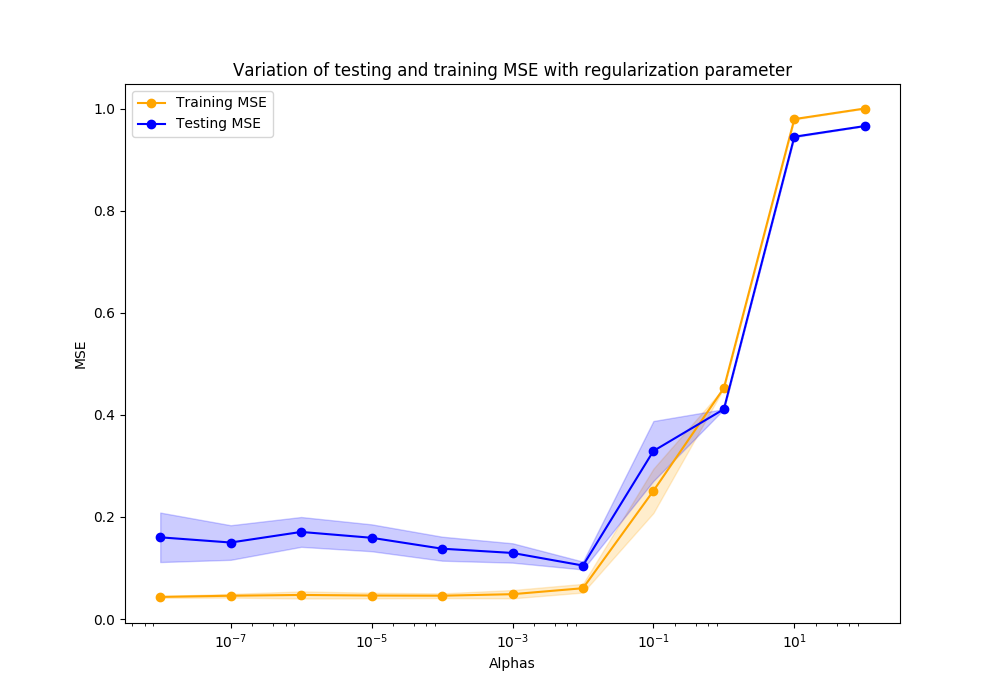
“Adam” would be the best choice, because it is a first order method and do not to calculate the hessian, which lasts very long for big networks.

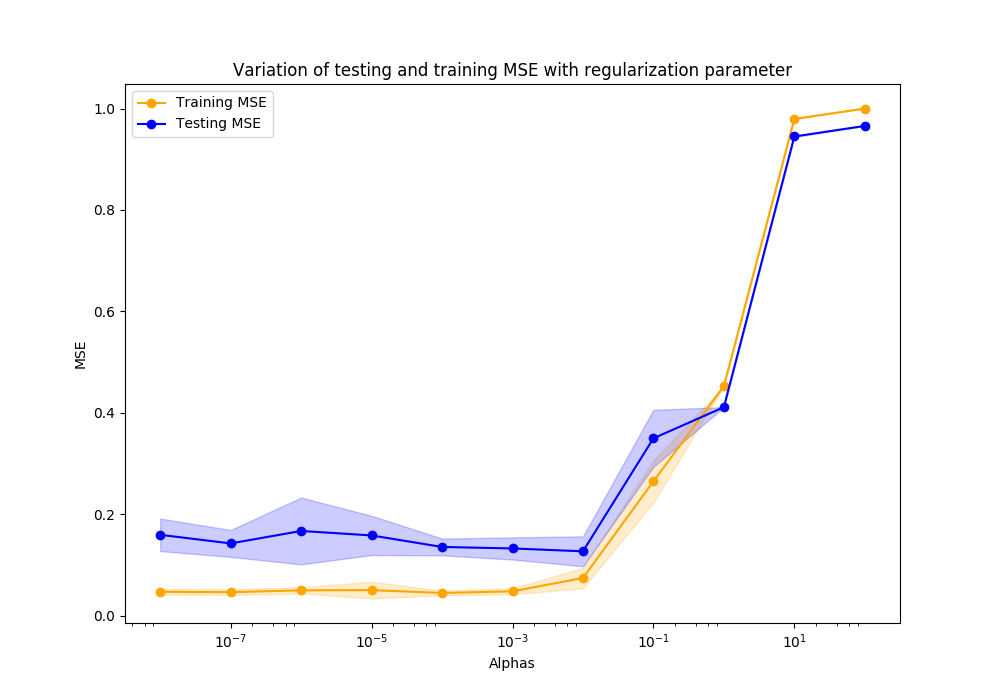
1.2 Regularized Neural Networks

a) Weight Decay:

Include plots of the variation of MSE of the training and test set with the value of α:







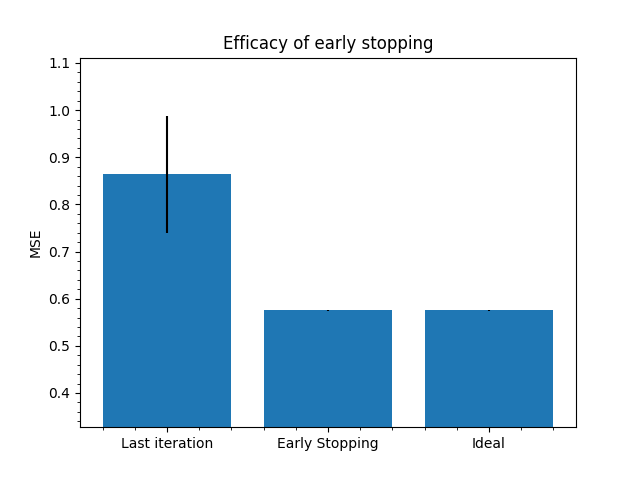
What is the best value of α?

The best value of α as seen in the plots above is 10^-3.

Is regularization used to overcome overfitting or underfitting? Why ?

Regularization is used to reduce overfitting through adding a penalty for complexity to the loss function.

b) Early Stopping:



Include the bar plots to compare the errors on the test sets at the last training iterations, at early stopping and when it is minimal.

In the light of question 1.1.b) is it expected that early stopping happens (validation error is minimized) at the same iteration number for all random seeds ? Is it coherent with your results?

No it should not happen on the same iteration on random seeds, because it compares the MSE, and on differenct seeds the number of iteration will differ. This is also coherent with our results.

Early stopping in its standard form is a little different, instead of stopping when the validation error is minimized, one stops training as soon as the validation error increases. What are the pros and cons of those standard form of early stopping and the one you implemented ?

Stopping when the error is minimized results in a better fitted neuronal network, but the time to get it is longer as with the real early stopping where the result is not as perfect, but faster. The disadvantage is if the validation error increases just for a short bit and decreases later on, we can’t detect it, because we stopped to early.

Explain your choice of number of hidden neurons, regularization parameter and solver. Then describe in a short paragraph but rigorously the protocol followed to identify the optimal random seed (mention all the parameter you chose such as ).

We choose our number of hidden neurons to be as optimal as possible, because to few neurons are a cause for underfitting, to many are a cause for overfitting. We used early stopping to prevent overfitting as means of regulariation and we used the “lbfgs” solver because it gets faster to a lower MSE .

To identify the optimal seed we calculated the minimal validation error for every seed. Then we checked which seed had the smallest minimal validation error and with this information we have the optimal random seed with the corresponding regressor.

Report the mean and standard deviation of your training, validation and testing error. Report the training, validation and testing error of your optimal random seed.