The Lords Of The Neurons: OMML-Homework 1

Eleonora Grassucci, Daniele Mocavini, Dario Stagnitto

Given the data set:

$$(x^p, y^p) : x^p \in R^2, y^p \in R, p = 1, ..., 200$$

Consider a shallow MLP of the type

$$f(x) = \sum_{j=1}^{N} v_j \cdot g\Big(\sum_{i=1}^{n} w_{ji} \cdot x_i - b_j\Big)$$

where $g(.)$ is a **tanh** activation function with $\sigma = 1$.

And consider an RBF network of the type

$$f(x) = \sum_{j=1}^{N} v_j \cdot \phi(||x^i - c_j||)$$

where

$$\phi(||x^i - c_j||) = e^{(||x - cj||/\sigma)^2}$$

with $\sigma > 0$.

We need to find the parameters v_j, w_{ji} and b_j for MLP and the parameters v_j and c_j for RBF which minimize the error function:

$$E(\omega; \pi) = \frac{1}{2P} \sum_{p=1}^{P} ||f(x^p) - y^p||^2 + \rho \cdot ||\omega||^2$$

Where $\omega = (v_j, w_{ji}, b_j)$ for MLP and $\omega = (v_j, c_j)$ for RBF and P is the dimension of the training set.

The optimization method we used to solve the minimization problem is the Broyden-Fletcher-Goldfarb-Shanno (BFGS). This method requires only the gradient of the objective to be computed at each iteration. It means that the Hessian matrix is not directly computed but is approximated. By successive measurements of the gradient, it builds a quadratic model of the objective function.

Ex.1.1: Full MLP

(*Click for the plot of function recostruction for Full MLP*)

First of all, we set $\rho = 0.005$, that is, the range medium value and $\sigma = 1$. Given that we had to provide some initial values for the optimization, we decided to choose them in a random way (using uniform values between 0 and 1). With these parameters we applied a Grid Search with a neuron range from 1 to 25 then we repeated this operation twice.

Moving along this way, we should have been able to avoid the "bad or unlucky started point"; eventually we decided to use $N = 21$.

Besides Grid Search we also tried other N values, it seems that an overfitting problem does not occurs because test error, even if worse, it's not so relevant. (Fig.2 and Table 1)

Table 1

Neurons	Training Error	Test Error	Time (sec)
30	0.0035	0.0068	25.49
40	0.0030	0.0116	31.76
50	0.0029	0.009	62.03

Once optimized the number of neurons, we focused on the ρ parameter using a grid of values composed by three fixed values (**0.00001**,**0.0005**,**0.001**) and two random values.

This way of splitting the Grids Searchs made the code faster. Moreover, doing the optimization twice, the test error is better. In fact, we got a value of 0.00382 against a value of 0.00647 obtained by optimizing directly with the "optimal" hyperparameters found.

As shown by the table below , the minimum value of ρ gives back a small error but it takes more function evaluations; we decided to use the ρ value with the lowest value in terms of error.

Table 2

Neurons	Rho	Func.Eval.	Time (sec)	Initial Train Err.	Train Err.	Test Err.
21	0.00001	147.662	15.61	13.3947	0.003570	0.006475
21	0.0005	67.424	4.32	13.3947	0.068946	0.084954

Furthermore, in the Fig.3 the ρ behaviour is shown.

Ex.1.2: Full RBF

(*Click for the plot of function recostruction for Full RBF*)

We fixed the value of ρ at the minumum values (0.00001) and $\sigma = 0.5$. We choose initial values in a random way using uniform values between 0 and 1 for v and uniform values between -2 and 2 for c.

As done before, we applied the Grid Search for Neuron range from 1 to 100. As always, we run this twice to avoid the "bad start".

At this point, we implemented the **Gradient** which allows us to reduce the **number of functions evaluation**. The result of Grid Search is 50 neurons.

With this configuration, we applied the Grid Search for σ with the following values: **0.25**, **0.75**, **1**, **1.5** and for **v** and **c** the parameters already optimized.

As we can see from Fig.6 the minimum error is with $\sigma = 1$.

Overfitting problem does not occurs even with a high number of neurons (Fig.5)

Table 3

Neurons	Rho	Func.Eval.	Time (sec)	Initial Train Err.	Train Err.	Test Err.
50	0.00001	1150	2.26	9.9023	0.0072	0.0059

Ex.2.1: Extreme Learning

(*Click for the plot of function recostruction for Extreme Learning MLP*)

In this section we used the same hyperparameters of the 1.1. With this algorithm we've seen that time and number of functions evaluation are lower, but error which has been resulted to be much higher in spite of 15 different starting points were used.

So, we have done some try increasing the number of neurons. With 100 neurons the test error is equal to 0.225 while using 500 neurons the test error is 0.09 but with a number of functions evaluations as same as in the 1.1.

Table 4

Type	Initial Train Err.	Train Err.	Test Err.	Nfev	Time (sec.)
FULL	13.3947	0.00357	0.0064	147662	15.26
EXTREME	13.00	0.14642	0.3154	3611	0.21

Ex.2.2: Unsupervised Center Selection

(*Click for the plot of function recostruction for Unsupervised RBF*)

In this point for the unsupervised selection of the centers we implemented the $K - Means++$ algorithm, setting "matricola" as initial random state and the hyperparameters from 1.2

The results turned out to be worse but faster. We used 5 different starting points. We tried with different configuration of the number of neurons: with 100 neurons the test error is equal to 0.034 while using 150 neurons (max number allowed) the test error is 0.033.

Table 5

Type	Initial Train Err.	Train Err.	Test Err.	Nfev	Time (sec.)
FULL	9.9023	0.0072	0.0059	1150	2.23
UNSUPERVISED	8.20	0.0186	0.0642	262	0.34

Ex.3: Two Blocks Decomposition

(*Click for the plot of function recostruction for Two Blocks RBF*)

In this section we used the same hyperparameters of the 1.2 but using also $K - Means++$ algorithm and we added the **early stopping** parameter. The network used has been the **RBF**. Two different functions have been built to calculate the Gradient in the **Two Blocks** method.

To obtain a number of function evaluations similar to the one of the Full RBF, we have to set the value of early stopping equal to 2. In this way we got the same test error despite a faster optimizaion.

However, we can increase this early stopping value to obtain a lower value of the test error. (Table 5) As we can see from Fig.10, after a rapid initial decrease, the function converges to the value of 0.00529. We have decided to take the lowest early stopping value which allows us to minimize the test error.

Table 6

Type	Early Stopping	Initial Train Err.	Train Err.	Test Err.	Nfev	Time (sec.)
FULL	-	9.9023	0.0072	0.0059	1150	2.23
TWO BLOCKS	2	12.28	0.0030	0.0059	1452	1.84
TWO BLOCKS	115	12.28	0.0062	0.0052	11063	13.14

Summary Table

Ex	FFN	Settings	Train Err	Test Err	Optimization Time
Q1.1	Full MLP	N=21 σ= 1 ρ=0.00001	0.0035	0.0064	15.26
Q1.2	Full RBF	N=50 σ=1 ρ =0.00001	0.0072	0.0059	2.26
Q2.1	Extreme MLP	N=21 σ=1 ρ =0.00001	0.1464	0.3154	0.21
Q2.2	Unsupervised c RBF	N=50 σ=1 ρ =0.00001	0.0186	0.0642	0.34
Q3.0	The ONE you choose (RBF)	N=50 σ=1 ρ =0.00001	0.0062	0.0052	13.14

Images

Fig.1: Full MLP

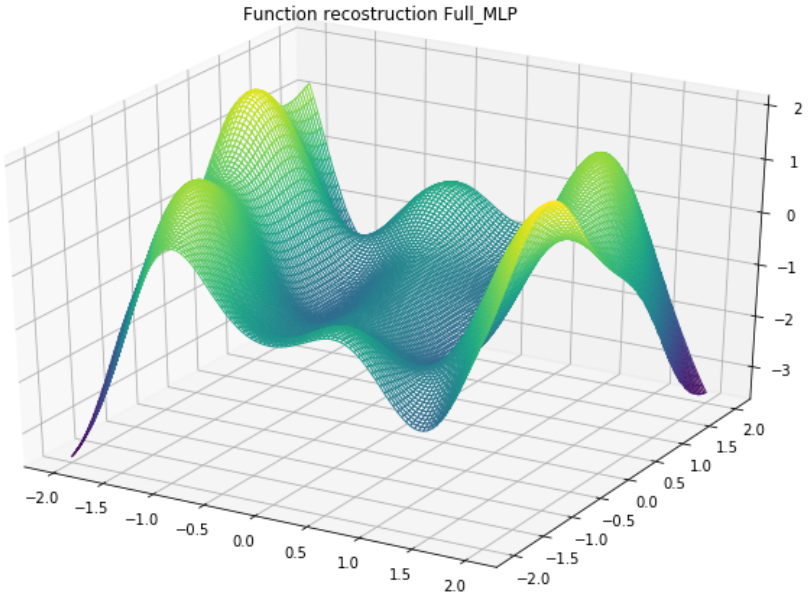

Fig.2: Error behaviour

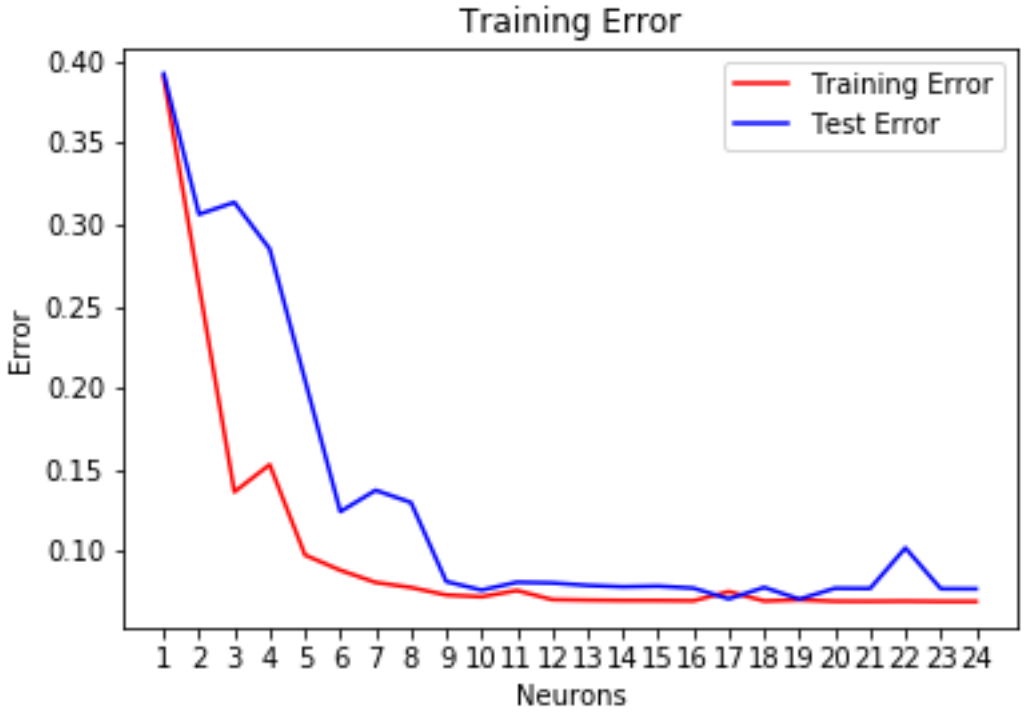

Fig.3: Error vs Rho

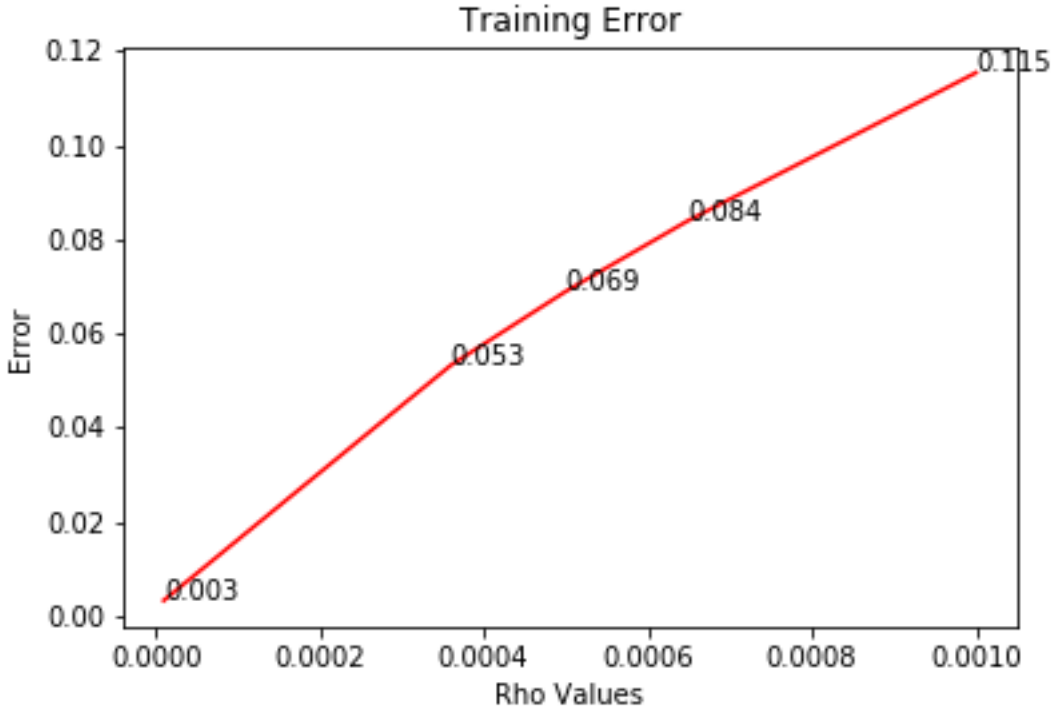

Fig.4: Full RBF

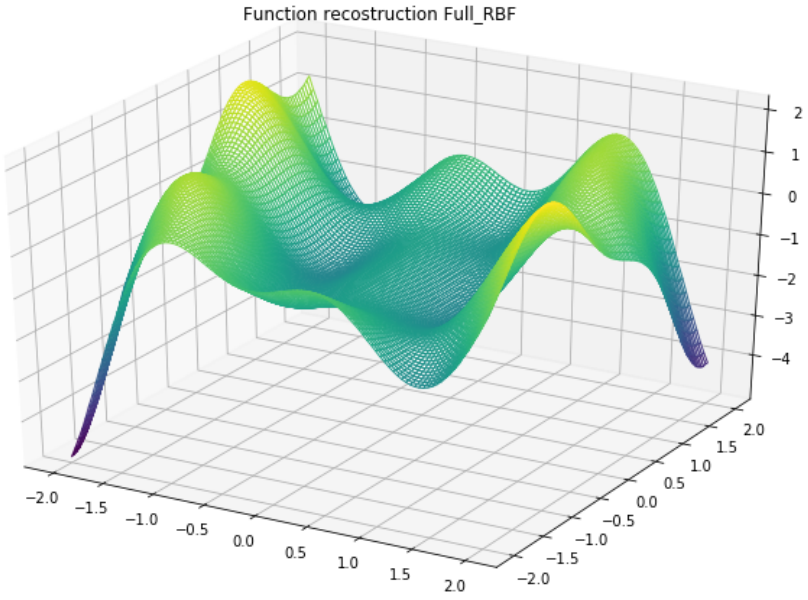

Fig.5: Error behaviour

Fig.6: Error vs Sigma

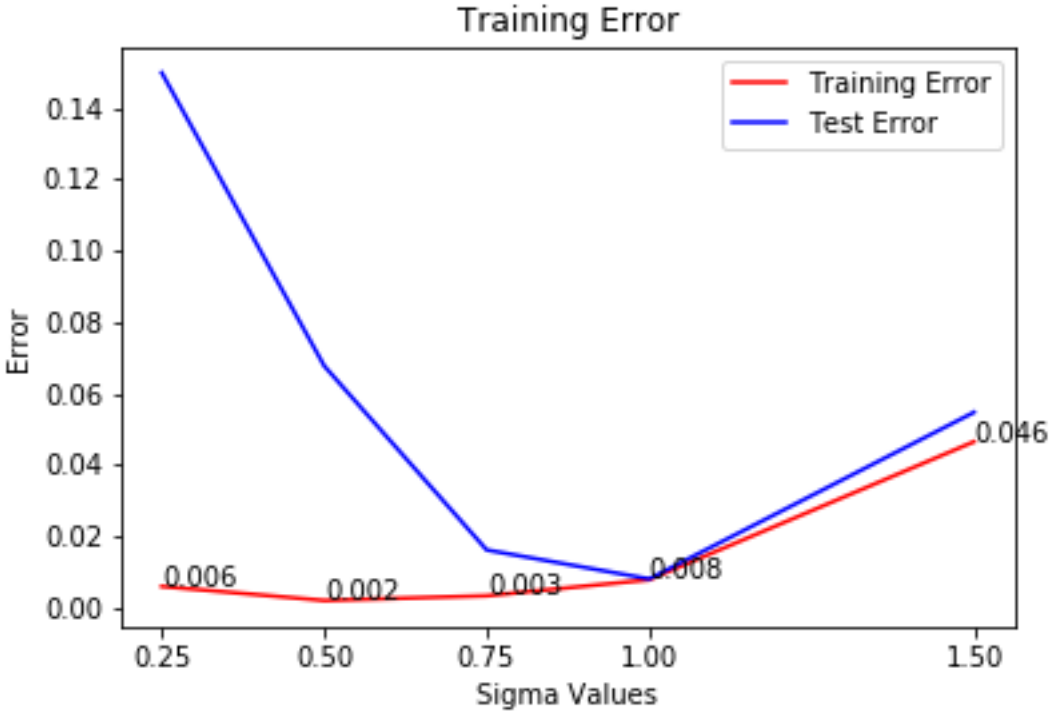

Fig.7: Extreme Learing

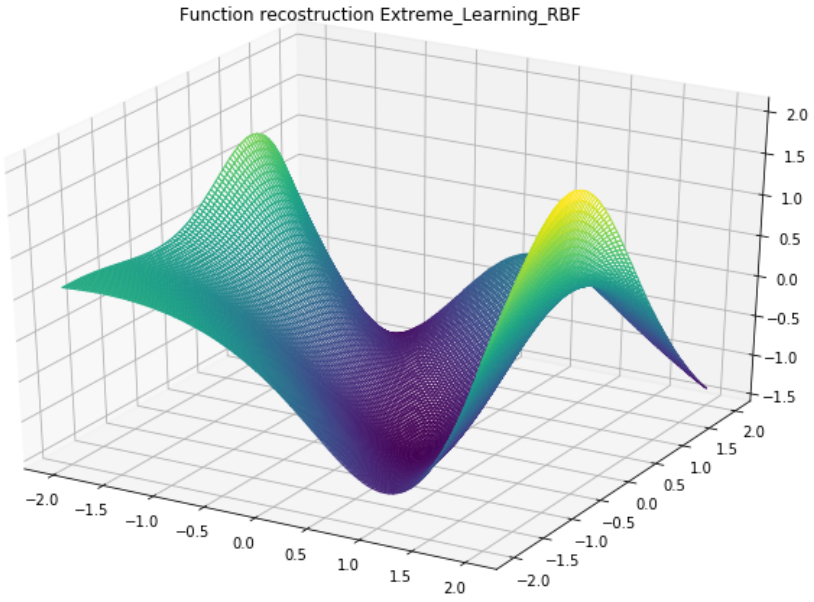

Fig.8: Unsupervised Center Selection RBF

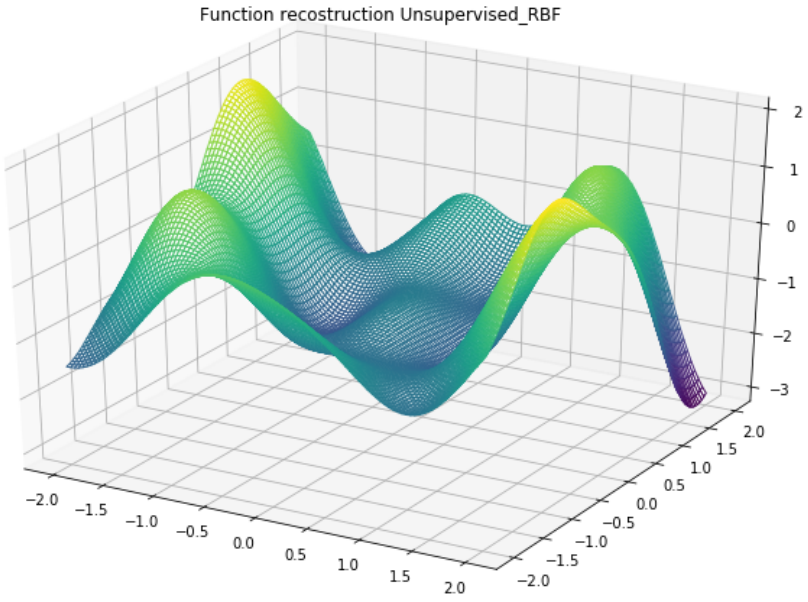

Fig.9: Two Block RBF

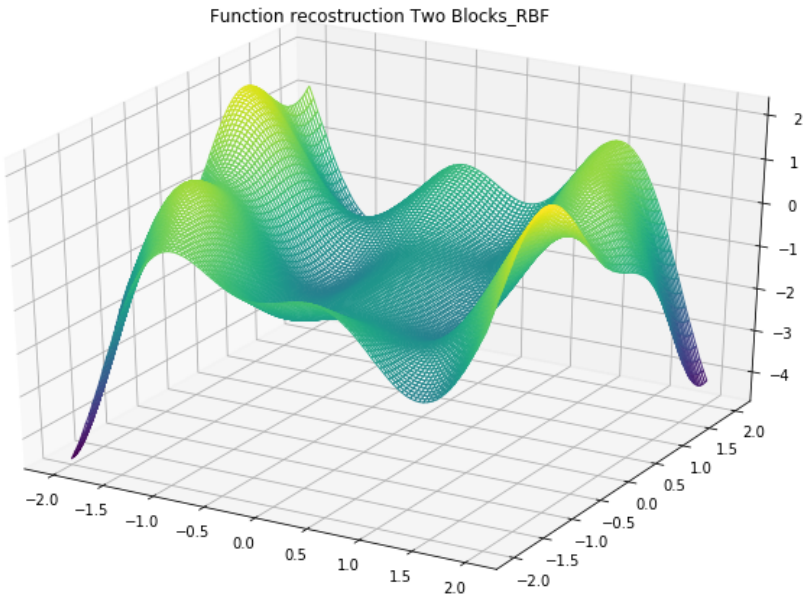

Fig.10: Error vs Outer Iter