SAPIENZA
UNIVERSITÀ DI ROMA

Report project1 - Group MDB - OMML

Marco Casalbore, Daryna Hnidets, Bianca Ghiurca

Fall 2023

		N	ρ	σ	$\textbf{Error}_{\textbf{Train}_i}$	$\textbf{Error}_{\textbf{Train}_f}$	$\textbf{Error}_{\textbf{Test}_f}$	Opt. time
$Q_{1.1}$	**Full MLP**	50	$1 * 10^{-5}$	1	25.59795	0.00020	0.00053	1.5 *sec*
$Q_{1.2}$	**Full RBF**	50	$1 * 10^{-5}$	0.9	1.50177	0.00035	0.00134	0.41 *sec*
$Q_{2.1}$	**Extreme MLP**	50	$1 * 10^{-5}$	1	7.80984	0.00192	0.00622	0.1 *msec*
$Q_{2.2}$	**Uns. c RBF**	50	$1 * 10^{-5}$	0.9	3.25541	0.00116	0.00537	0.1 *msec*
Q_{bonus}	**Block Decomp**	50	$1 * 10^{-5}$	0.9	3.75608	0.00110	*.*****	4 *sec*

1 Full Minimization

1.1 Q.1.1 - MLP Network

The structure of the neural network, and therefore the choice of the hyperparameters, was determined by implementing a grid-search, particularly for selecting the number of neurons (N), the spread of the hyperbolic tangent (σ), and the regularization term (ρ); on top of the optimization process, the K-Fold Cross-Validation methodology was also employed, with the choice of folds set to k=5 (tests with k=10 were conducted but did not yield any significant improvement). For the Grid-Search the following combinations were considered:

$$Neurons \in [10, 20, 30, 40, 50]; \quad \rho \in [10^{-3}, 10^{-4}, 10^{-5}]; \quad \sigma \in [0.9, 1, 1.2]$$

The final choice of hyperparameters, following the optimal ones suggested by the Grid-Search, settled on N=50, σ=1, and ρ=10^{-5} because these values accurately approximated the given objective function without making computational calculations overly expensive; in figure 2 it is possible to confront the original given function with the approximated one.
In exploring the basic configuration of the network, various seeds were adopted as test benchmarks: the results showed that, with identical hyperparameters, changing the seed (even by the order of 10^4) did not lead to a significant variation in the objective function value at the "optimal" point; the network's performance remained almost unchanged, with the only noticeable change being a slight variation in computational time, ranging from 1 to 3 seconds (due to the fact that the starting variables were initialized differently); the final seed was set to 200 based on those considerations.

Attention was also given to the "tolerance" value within the minimization process: an exponential increase in computational time was observed when this value was completely eliminated (computation time exceeding 10 seconds). Through a trial&error process, it was concluded that tolerance values between 10^{-4} and 10^{-2} drastically reduced computational time, but also resulted in a deterioration of training and test errors; for these reasons, the tolerance was set to 10^{-7}.
Regarding the optimization process, the "L-BFGS-B" minimization algorithm (implemented by the scipy library) was chosen, proving to be the most suitable for the dimensions of the given problem. The algorithm returned the message "success: True CONVERGENCE: REL REDUCTION OF F $\leq$ FACTR*EPSMCH", with 1186 iterations, 1273 function and gradient evaluations and a computational time of 1,5 seconds. The initial value of the objective function turned out to be $Initial_{Obj.f} = 25.59867$, while the value after the optimization process is $Final_{Obj.f} = 0.00136$. As for the values of the norm of the gradient at the starting and final points, they are $Initial_{Grad} = 33.71731$ and $Final_{Grad} = 0.00466$, respectively.
From the analysis of the created model, observable in figure 1, it was noted that deviating from the optimal value suggested by the Grid-Search (N=50) with fixed ρ and σ led to a general deterioration in network performance, such as a worsening of the training error; for this reason, N=50 turned out to be the best configuration in terms of minimizing underfitting (note that training error and test error values, as requested, are subtracted of the regularization term).
During the analysis, the phenomenon of over-fitting could not be detected based on the given data-set. From the perspective of the hyperbolic tangent spread, it was noticeable that deviating from the optimal value suggested by the grid search resulted in worsened test and training errors.
On the other hand, studying the model's performance as the regularization term (ρ) varies, it was observed that for different ρ values (in the order of 10^{-3}), the network's performance deteriorates in terms of error and also the shape of the function graph deterioreted, even though the number of iterations required by the optimization algorithm is halved.

1.2 Q.1.2 - RBF Network

Once the model of the MLP was constructed, building the network based on a radial basis function was almost immediate.
As requested, the Gaussian function is employed as the Radial Basis Function (RBF) for the creation of the neural network: $\Phi(||x - c_j||) = e^{-(||x-c_j||/\sigma)^2}$:
Following the approach adopted previously, the determination of the hyperparameters was carried out by implementing a Grid-Search (for number of centers N, ρ, σ) supported by a K-Fold Cross-Validation procedure (with k fixed, as before, at 5).
The optimal combination, as suggested by the Grid-Search and various conducted tests, turned out to be: N=50, $\rho = 1 * 10^{-5}$ and $\sigma = 0.9$. For the Grid-Search the following combinations were considered:

$$Centers \in [10, 20, 30, 40, 50], \quad \rho \in [10^{-3}, 10^{-4}, 10^{-5}], \quad \sigma \in [0.9, 1, 1.2]$$

Similar to the MLP, the model's behavior was analyzed concerning the variation of tolerance (within the optimization process) and seed (parameter initialization): decreasing tolerance has brought a better performance; on the other hand, varying the seed revealed that the precision of the function approximated in the plot was more noticeably affected. Based on those considerations and for modeling simplicity, these parameters were set to the same values as those of the MLP: seed=200, tol=10^{-7}.
Considering the behavior of the network as the number of centers (N) increases (from 10 to 50), figure 3, it was possible to observe, by changing the N value, a decrease in training error and test error until reaching optimal values for N=50 (the best condition for underfitting, also supported by the Grid-Search); as expected, it was immediately evident that the number of centers and computational time are linearly dependent on each other.
Also in this case, by changing the ρ value was observed a deterioration in the network's performances in both errors and graph approximation.
Again, overfitting is not appreciable in this model: from a certain value onward, the training error remains consistently low.

As for the optimization process of the MLP, also in this case the "L-BFGS-B" minimization algorithm (implemented by the scipy library) was chosen. The algorithm returned the message "success: True CONVERGENCE: REL REDUCTION OF F $\leq$ FACTR*EPSMCH", with 296 iterations, 306 function and gradient evaluations and a computational time of 0.41 seconds. The initial value of the objective function turned out to be $Initial_{Obj.f} = 1.50249$, while the value after the optimization process is $Final_{Obj.f} = 0.00118$. As for the values of the norm of the gradient at the starting and final points, they are $Initial_{Grad} = 1.03075$ and $Final_{Grad} = 0.00084$, respectively.
In Figure 4, it is possible to compare the plot of the original function with that of the approximation using RBF.

1.3 A brief comparison

Even though the two networks, MLP and RBF network, closely resemble each other both in terms of Python code and theoretically, the RBF network proves to be faster for the chosen configurations: it requires, on average, half the computational time and half the gradient and objective function evaluations compared to the MLP; on the other hand, the MLP model appears to be somewhat more visually robust (with an excellent similarity between the approximation and the original plot). Indeed we observed that in the MLP the training and the test errors are lower than the values of the RBF.
It is interesting to note that with the exponential increase in N, the RBF network remains the safer choice between the two. The MLP model tends to significantly increase computational time with a high number of neurons: as istance, with N=300, the MLP takes about 20 seconds, while the RBF network takes just under 5 seconds. Comparing the two plots in figure 5 (RBF and MLP), it is possible to observe that the plot of the MLP is more precise.

2 Two Blocks Methods

2.1 Q.2.1 - MLP Network

As requested, consider the MLP neural network sized at point Q.1.1 where N=50, $\rho = 1 * 10^{-5}$ and $\sigma = 1$.
For this configuration, it was decided to explore various possible seed initializations through a multi-start process which returned the optimal value of seed=87057.
Once the weights w and the biases b are fixed through random procedure, the error minimization problem becomes quadratic, where the loss function depends only on the output weights v: for this reason, the decision was made to implement the LSTSQ function directly from the numpy library, called np.linalg.lstsq, as it proved to be the most efficient and straightforward method; indeed, by doing so, we are sizing the neural network through an extreme learning process.
The results of the optimization routine in terms of final training and test errors are $E_{train.f}$= 0.00192 and $E_{test.f} = 0.00622$. From an initial analysis of the network, it appears to be extremely fast (computational time of milliseconds) and efficient compared to the fully minimized MLP, this is due to the fact that in this case the code is performing a simple matrix inversion, whereas in the case of total minimization, an optimization algorithm needs to be applied, as in our case with L-BFGS-B; however, this computational speed comes at the cost of accuracy, both in terms of graphical representation and errors: the plot of the function approximated by the full minimized MLP turns out to be the one that best approximates the given objective function, and the error values are significantly lower than those obtained with extreme learning.
The comparison between the given function and the approximated one with MLP extreme learning is shown in figure 6.

2.2 Q.2.2 - RBF Network

As requested, consider the RBF neural network sized at point Q.1.2 where N=50, $\rho = 1 * 10^{-5}$ and $\sigma = 0.9$.
The situation is similar to that of point Q.2.1, with the difference that in this case (in addition to being an RBF network and not an MLP), the minimization of the error function occurs with respect to the

parameters v, given that the centers (N) are randomly selected from the samples before performing the optimization procedure.
In this case as well, a multi-start was implemented to find the best seed in terms of performance, and it turned out to be seed=87226.
Here too, the chosen minimization routine was np.linalg.lstsq since the only difference from the previous problem was the activation function g(t).
This model, implemented with an unsupervised selection of the centers, yielded the following results: $E_{train.f}$=0.00116 and $E_{test.f}$=0.00537.
As expected from the performance analysis conducted for the previous point, the final considerations regarding the comparison between RBF full minimized and RBF with unsupervised selection of centers are mostly the same as those made for MLP full minimized and MLP extreme: computational time is certainly the most evident improvement brought about by extreme learning, although, also in this case, such computational speed is traded off for a deterioration in the accuracy of the plot and the goodness of errors (although the variation in terms of error is less severe in this case).
The comparison between the given function and the approximated one with RBF unsupervised selection of centers is shown in figure 7.
From a general perspective, the result obtained through RBF with unsupervised selection of centers is the one that, graphically, exhibits a greater number of irregularities, several lateral spikes and less smoothness, and for these reasons, it is the one that approximates the original function in a less favorable manner (a fact confirmed by the higher error values as well). For this reason, it appears to be the least accurate configuration among the four, although it is the fastest overall.

3 Q.Bonus

For the two block decomposition method the work was conducted on the RBF network with parameters N=50, σ=0.9, and ρ=10^{-5}. In this instance, the entire dataset was employed as our training set, and the seed was set to 200. The LSTSQ solver addressed the convex problem defined by fixing the centers; subsequently, we worked on the non-convex minimization for the centers using the L-BFGS-B solver, with tol set to 10^{-7}. Early stopping rules were implemented, setting a maximum number of iterations to 100 and patience fixed at 10.
The number of function and gradient evaluations was 2148, resulting in a training error of 0.00110 and a computational time of 4 seconds. Comparing these results with those obtained from the RBF with unsupervised selection of centers (Q.2.2), it was observed that precision improved in this case, with the training error being lower, however, the unsupervised selection of centers proved significantly faster, taking only about 0.1 milliseconds compared to the 4 seconds required for the two-block decomposition method.
Upon comparing the network built using the two block decomposition with the fully minimized RBF, it was noted that the second one exhibited a lower training error.
In figure 10 is reported a comparison between the original plot of the given function and the one approximated with the two block methods RBF network.
In figure 11 is observable a comparison between the RBF full minimized and the two block methods RBF network.

4 Images and Graphs

Figure 1: Performance measures varying the number of neurons, Q.1.1.

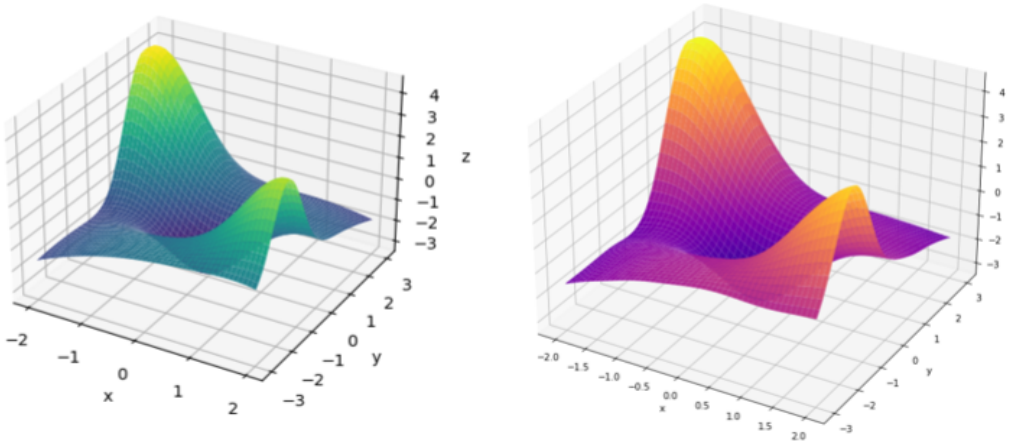

Figure 2: Comparison original plot (left) with MLP approximation (right), Q.1.1

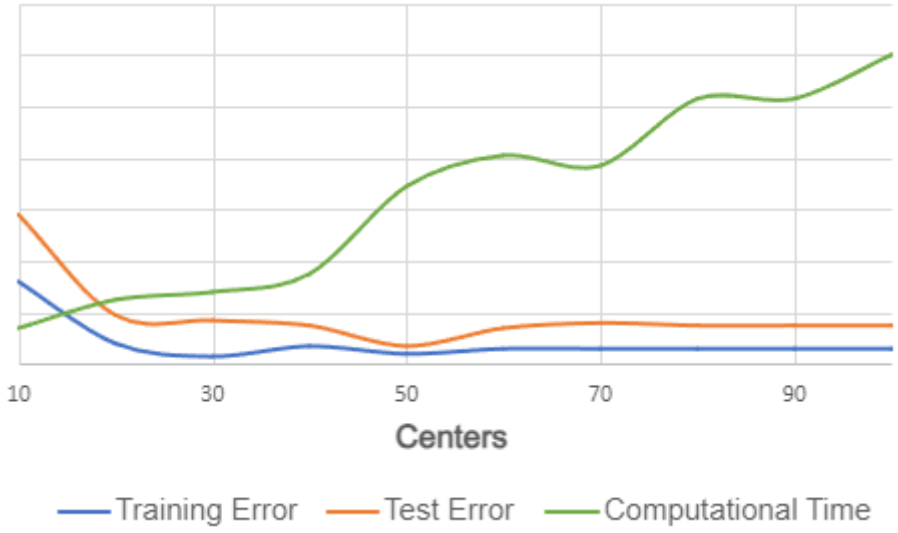

Figure 3: Performance measures varying the number of centers, Q.1.2

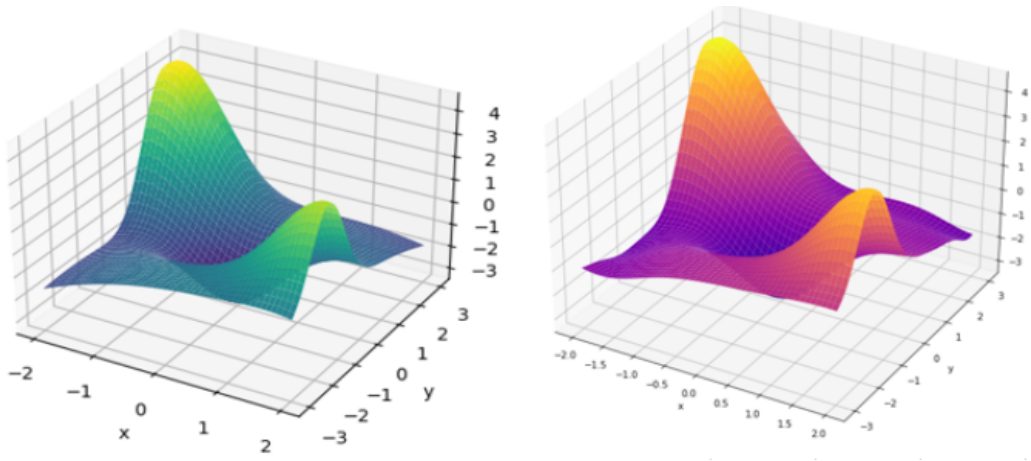

Figure 4: Comparison original plot with RBF-Network approximation, Q.1.2

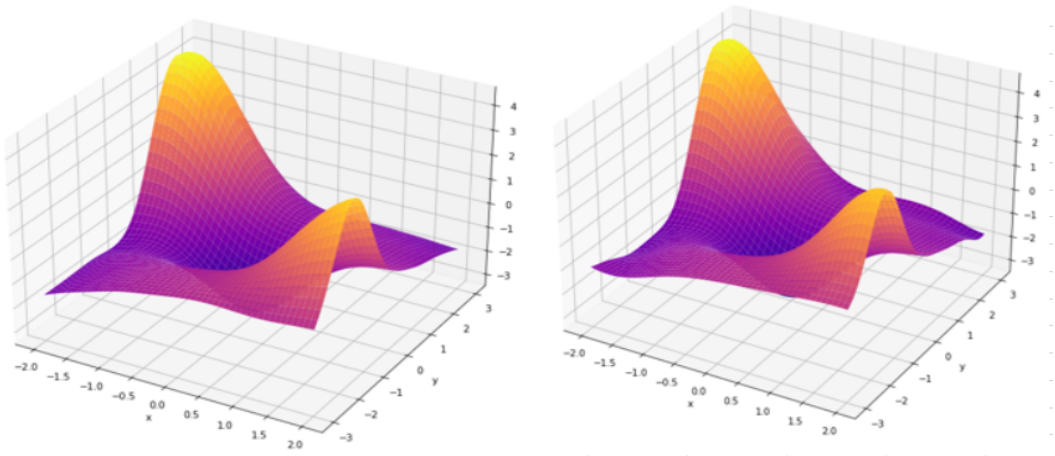

Figure 5: Comparison MLP plot approximation (left) with RBf plot approximation (right),Q.1.2

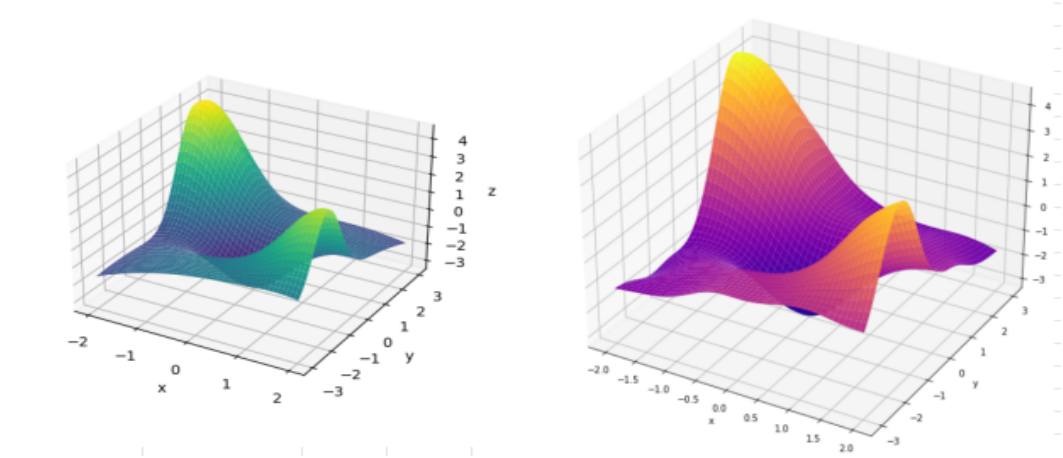

Figure 6: Comparison original plot with MLP extreme learning approximation, Q.2.1

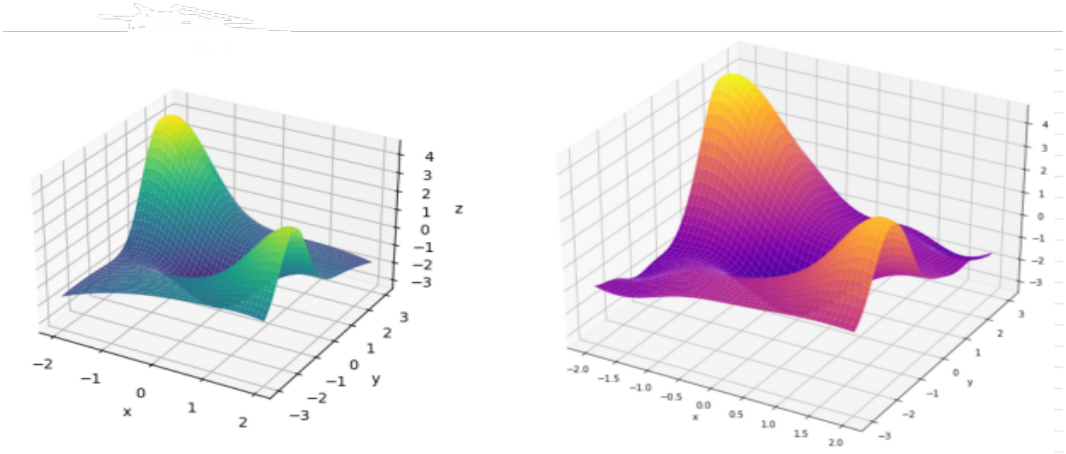

Figure 7: Comparison original plot with RBF unsupervised selection of centers approximation, Q.2.2

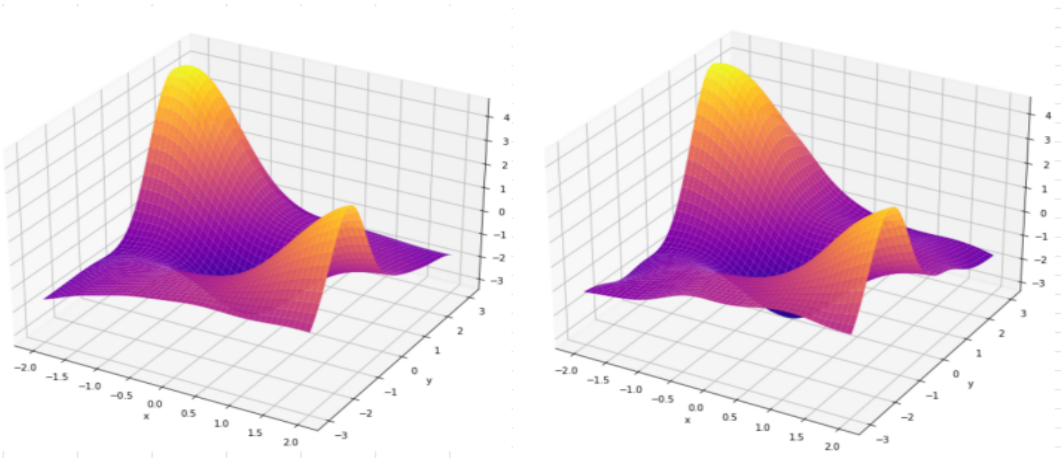

Figure 8: Comparison full MLP plot (left) with MLP extreme learning plot (right), Q.2.1

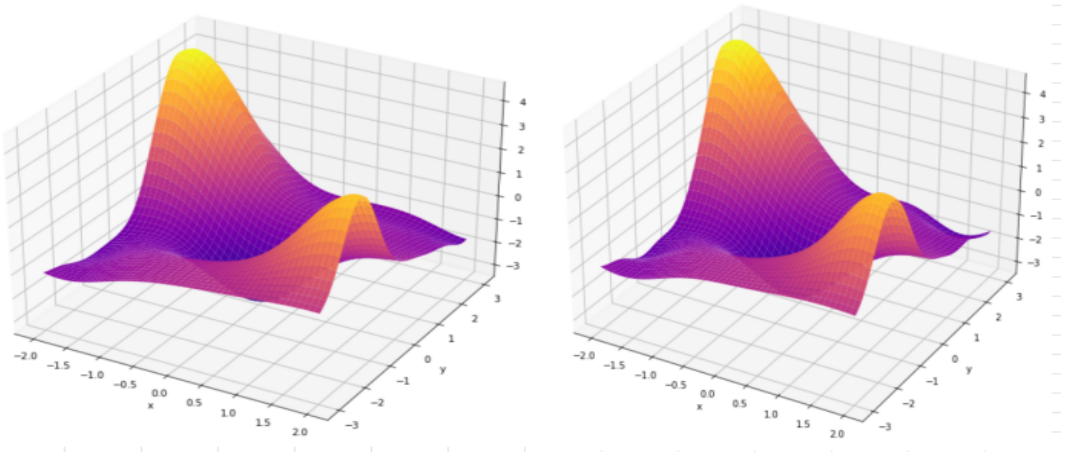

Figure 9: Comparison full RBF plot (left) with RBF unsupervised selection of centers plot(right), Q.2.2

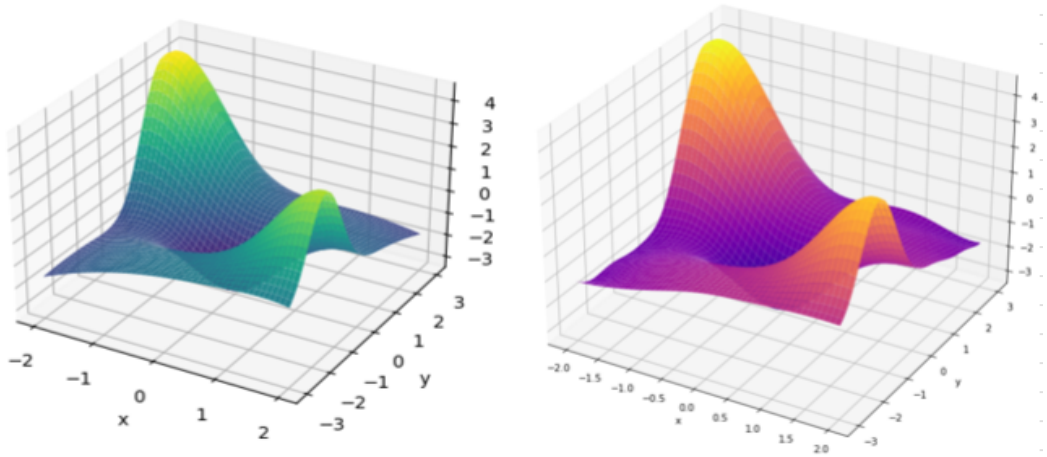

Figure 10: Comparison original plot with RBF two block decomposition method approximation, Q.bonus

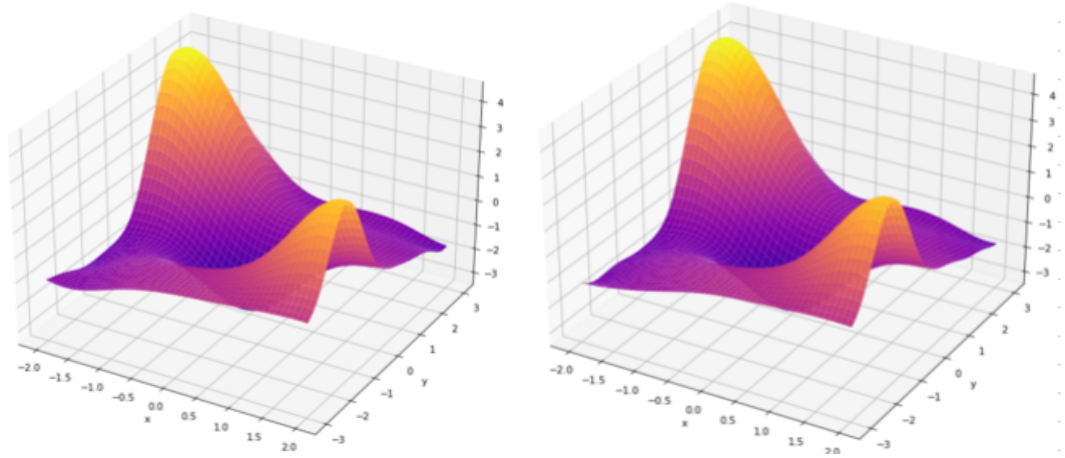

Figure 11: Comparison Full RBF plot (left) with two block decomposition method RBF plot (right), Q.bonus