Stepic (1861402), Nurasilova (1848425), Nastasic (1852026)

Project # 1 - MLP and Generalized RBF Network

November 25, 2019

SAPIENZA
UNIVERSITÀ DI ROMA

Contents

Introduction

Goal of the project is to reconstruct a two dimensional function F: $\mathbf{R}^2 \rightarrow \mathbf{R}$ in the region [-2, 2] x [-1, 1], implementing neural networks.

Given the data set:

$$(x^p, y^p) : x^p \in \mathbf{R}^2, y^p \in \mathbf{R}, p = 1, ..., 300$$

Regularized estimation error is given as:

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

We denote by π the hyper-parameters of the network to be settled by means of an heuristic procedure and ω the parameters to be settled by minimizing the regularized training error.

Q1: Full minimization - Feedforward Neural Network

Q1.1: MLP

Firstly, the data was split into training and validation dataset. They were used in order to perform hyper-parameter optimization using a Grid-Search 5-Fold cross-validation technique with regularized error as an objective function to prevent overfitting. Every fold was randomly shuffled before the training of the data. Then, as a final setting for the hyper-parameters, we wanted to find the configuration(a combination of hyper-parameters) that gives us the smallest average validation error calculated. This error is used to estimate how the data will perform on the unseen data. And as we see in optimization results below, there is not much difference between error on the validation and the test error, which shows that the model generalizes well. The final setting of hyper-parameters:

Number of neurons N	**Spread σ**	**Regularization term ρ**
28	**1**	**0.00001**

Moreover, **Figure 6** shows how the training and validation error change with the number of neurons when rho and sigma are fixed. A small number of neurons caused underfitting and had the biggest validation error. Cross-validation showed us that the choice for rho gave the best results when it had the smallest value. Furthermore, for the number of neurons, we chose 28, because the validation error reached its minimum. It was also one moment before it started to increase, causing overfitting on the unseen data.

Minimization in respect to ω is done using optimization procedure SciPy optimize with **BFGS** method with default function parameters.

Return message:	Optimization terminated successfully.
Number of iterations:	1569
Number of function evaluations:	185136
Number of gradient evaluations:	1624
Starting objective function value:	3.06501
Ending objective function value:	0.00305
Final norm of the gradient:	3.32912e-05
Initial training error:	3.06406
Final training error:	0.00027
Validation error:	0.000362
Test error:	0.00032

Plot of reconstructed function is shown in **Figure 1**.

Q1.2: RBF

We decided on hyper-parameter values performing Grid Search 5-Fold Cross-Validation in order to take into account model behaviour on unseen data, to avoid overfitting.

Number of neurons N alternated among values [2, 9, 25 ,33 ,55], σ among [1.11, 1.455, 1.99] and ρ between [0.0001, 0.00001]. 5-Fold Cross-Validation is applied on all possible combination of the values of hyper-parameters.

Combination of the hyper-parameters which generated the smallest validation error is chosen. Results of Cross-Validation, in terms of training and validation error values for different choices of hyper-parameters are shown in the **Table 3**.

The final setting of hyper-parameters is:

Number of neurons N	**Spread σ**	**Regularization term ρ**
25	**1.11**	**0.00001**

Finally, we performed minimization of the error function in respect to ω, with hyper-parameters fixed as above using optimization procedure SciPy optimize with **Conjugate Gradient** method with default function parameters (no tuning on tolerance, max number of iterations etc).

Return message:	Optimization terminated successfully.
Number of iterations:	5374
Number of function evaluations:	640486
Number of gradient evaluations:	8318
Starting objective function value:	0.70135
Ending objective function value:	0.02385
Final norm of the gradient:	0.00003
Initial training error:	0.70077
Final training error:	0.01237
Validation error:	0.01697
Test error:	0.01238

Plot of reconstructed function is shown in **Figure 2**.

	Full minimization MLP	**Full minimization RBF**
Test error	0.00032	0.01238
Number of function evaluations	185136	640486
Number of gradient evaluations	1624	8318
Computational time	84.37 seconds	106.75 seconds

Q2: Two blocks methods

Q2.1: Extreme Learning

With the setup of hyper-parameters from the task Q1.1 (N = 28, $\sigma = 1.0$, $\rho = 0.00001$), we implemented Extreme Learning procedure that fix randomly the values of w_{ij} and b_j for all i and j and optimizes regularized quadratic convex training error as a function of the only variables v, E(v). The random choice of vectors w i b is repeated 10 times.

Minimization of the error function in respect to v is done using optimization procedure SciPy optimize with **Conjugate Gradient** method with default function parameters (no tuning on tolerance, max number of iterations etc). In the following table is shown comparison between Full minimization MLP and Extreme Learning procedure.

	Full minimization MLP	**Extreme Learning**
Test error	0.00032	0.19241
Number of function evaluations	185136	46800
Number of gradient evaluations	1624	1560
Computational time	84.37 seconds	85.95 seconds

Plot of reconstructed function is shown in **Figure 3**.

Q2.2: Unsupervised center selection RBF

Method with unsupervised selection of the centers was written and procedure of minimizing of weights was performed. Additionally gradient function was implemented also in order to speed up the process of minimizing. We repeated a minimizing if weights 25 times with different values of initial weights and centers. We used optimization routines of the Optimization toolbox SciPy optimize with **Conjugate Gradient method** with default function parameters (no tuning on tolerance, max number of iterations etc), except jac parameter where we put our gradient function. We obtained following results with hyper parameters (N = 25, $\sigma = 1.11$, $\rho = 0.00001$):

Train error	0.07339
Test error	0.05419
Number of function evaluations	2220
Number of gradient evaluations	2220
Computational time	21 seconds for 25 repetitions

Plot of reconstructed function is shown in **Figure 4**.

Q3: Decomposition method

In this section we decide to consider RBF with the hyper-parameters chosen in exercise 2 question 1 (N = 25, $\sigma = 1.11$, $\rho = 0.00001$). We implemented algorithm which is performing two block decomposition method. In the first block the convex minimization with respect to the output weights was done. In order to speed up the process we computed gradient according to weights as for the exercise Q 2.2. In the second block the non convex minimization with respect to the centers was performed. For this we also implemented a computation of the gradient respect to centers. We used optimization routines of the Optimization toolbox SciPy optimize with **Conjugate Gradient method** with default function parameters (no tuning on tolerance, max number of iterations etc), except jac parameter where we put our gradient functions respect to the weights and centers in first and second blocks respectively.

Comparison of results from Q1.2, Q2.2 and Q3 are represented in the **Table 1**. Plot of reconstructed function is shown in **Figure 5**.

Summary of all the results is represented in the **Table 2**.

The stopping criteria Q3:	Values of weights and centers did not change in one iteration or if error of the second block is not changing significantly
Number of outer iterations (subproblems):	3
Number of function evaluations:	4618
Number of gradient evaluations:	4595
Execution time:	2.66 seconds

Table 1 Comparison of RBF from Q1.2, Q2.2 and Q3

	Q1.1. Full RBF	Q2.2. EL RBF	Q3. TBD RBF
test error	0.01238	0.05419	0.04676
Number of function evaluations	640486	2220	4618
Number of gradient evaluations	8318	2220	4595
Computational time	106.75 seconds	21 seconds for 25 rep.	2.66 seconds

Tables and Figures

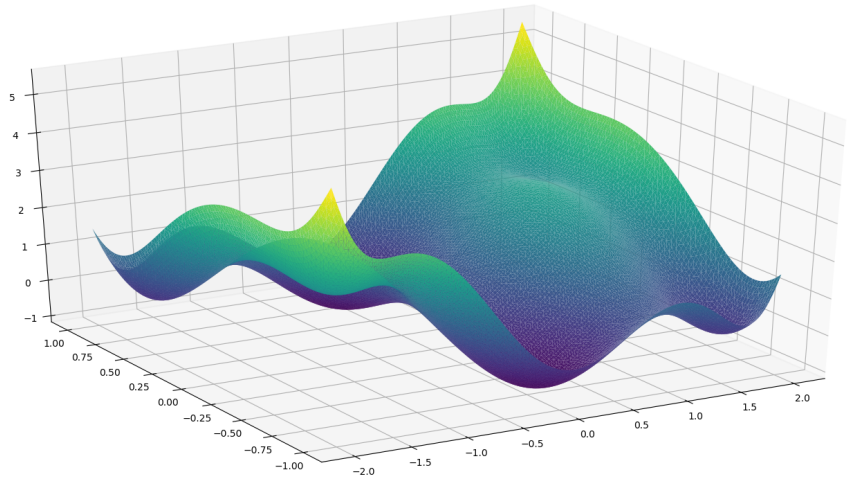

Figure 1: Function reconstruction MLP full minimization

Table 2 Final table of results

Q	Name	N	ρ	σ	tr. err.	val. err.	test err.	opt. time
1.1	Full MLP	28	0.00001	1	0.00027	0.000362	0.00032	84.37
1.2	Full RBF	25	0.00001	1.11	0.01237	0.01697	0.01238	106.75
2.1	EL MLP	28	0.00001	1	0.20659	0.19813	0.19241	85.95
2.2	EL RBF	25	0.00001	1.11	0.07820	0.07164	0.05419	21
3	TBD RBF	25	0.00001	1.11	0.0534	0.0475	0.04676	3.73

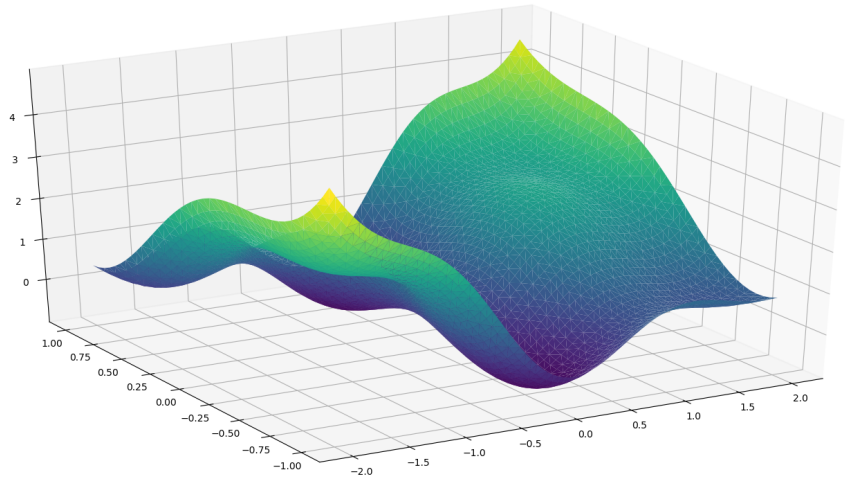

Figure 2: Function reconstruction RBF full minimization

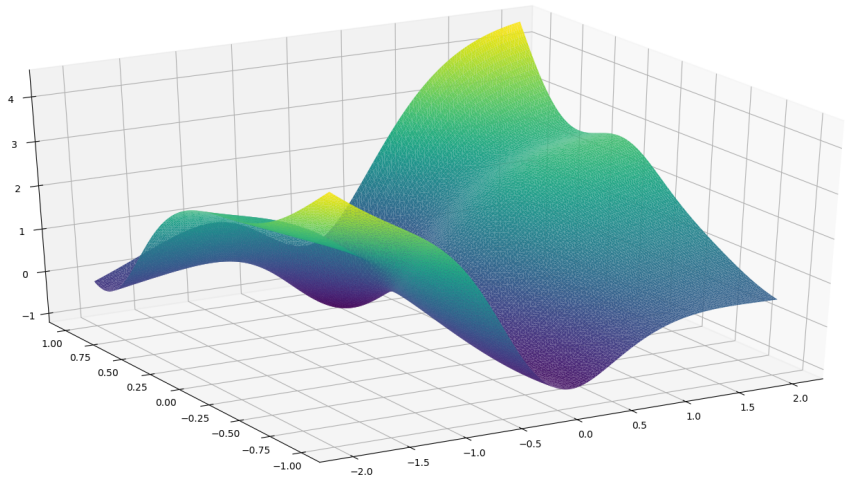

Figure 3: Function reconstruction MLP Extreme Learning

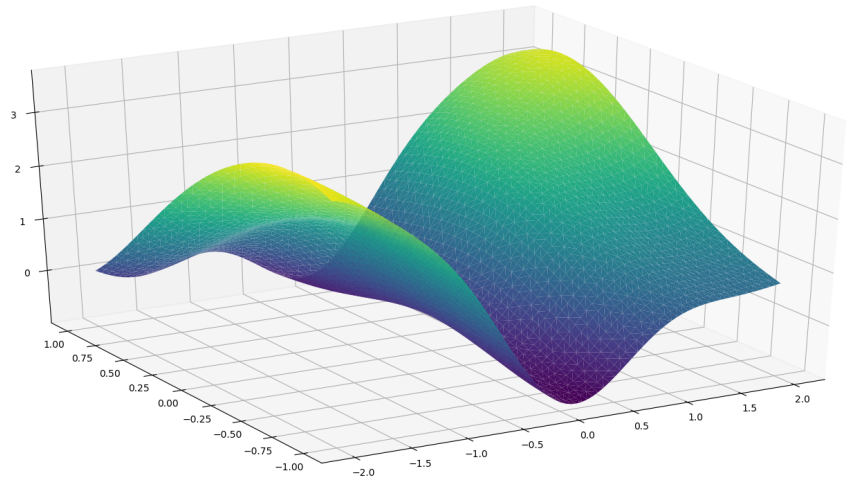

Figure 4: Function reconstruction RBF unsupervised centers

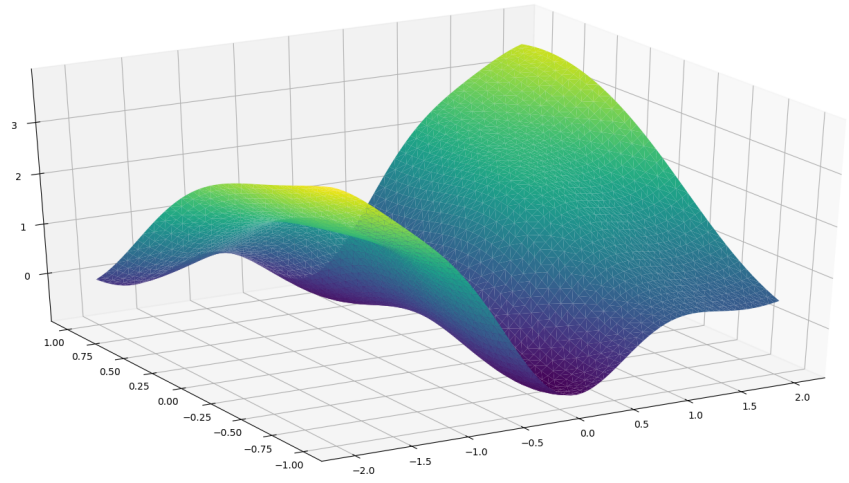

Figure 5: Function reconstruction RBF two block decomposition

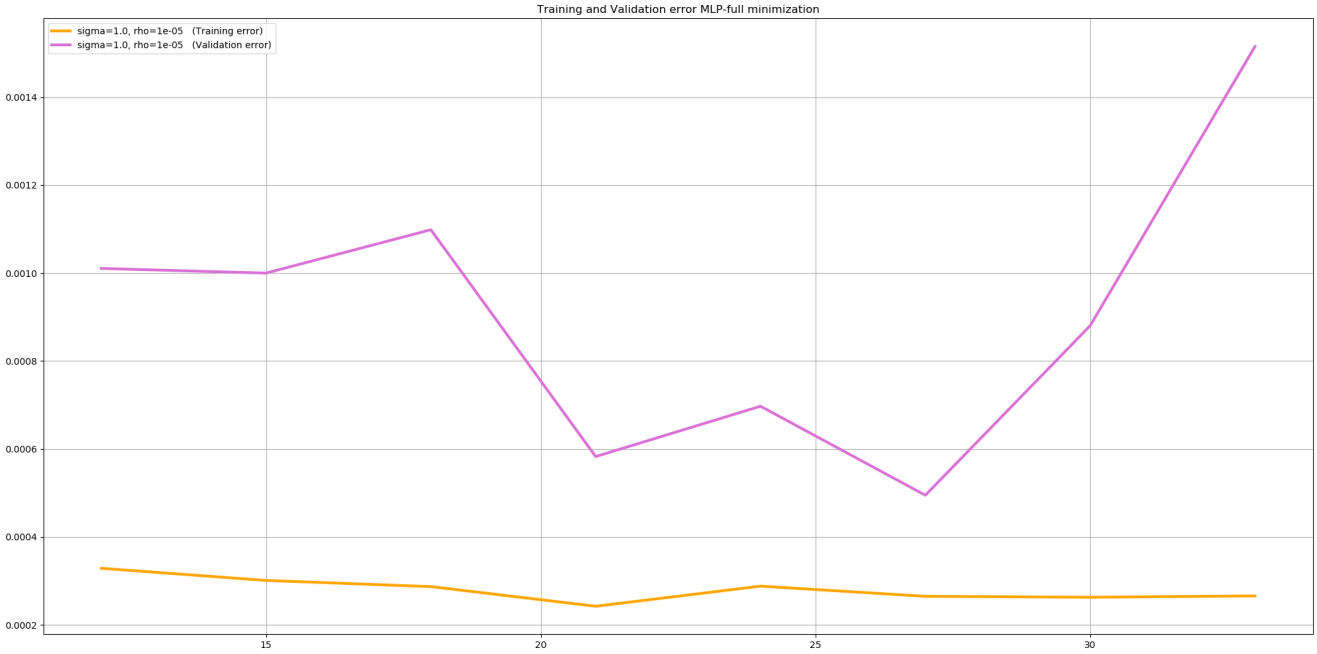

Figure 6: Training and Validation error MLP-full minimization

Table 3 RBF - Grid Search and 5-Fold CV hyperparameter selection

N	Rho	Sigma	Training error	Validation error
2	0.0001	1.11	0.6579	0.7539
2	0.0001	1.455	0.1978	0.2044
2	0.0001	1.99	0.2318	0.2317
2	1e-05	1.11	0.1738	0.1906
2	1e-05	1.455	0.1994	0.1951
2	1e-05	1.99	0.2279	0.2352
9	0.0001	1.11	0.0827	0.1239
9	0.0001	1.455	0.0999	0.1644
9	0.0001	1.99	0.1417	0.1574
9	1e-05	1.11	0.0474	0.0743
9	1e-05	1.455	0.0639	0.0938
9	1e-05	1.99	0.1190	0.1183
25	0.0001	1.11	0.0399	0.0712
25	0.0001	1.455	0.0903	0.1246
25	0.0001	1.99	0.1192	0.1535
25	1e-05	1.11	0.0110	0.0170
25	1e-05	1.455	0.0297	0.0551
25	1e-05	1.99	0.1063	0.1581
33	0.0001	1.11	0.0315	0.0721
33	0.0001	1.455	0.0828	0.1428
33	0.0001	1.99	0.1181	0.1701
33	1e-05	1.11	0.0086	0.0253
33	1e-05	1.455	0.0226	0.0332
33	1e-05	1.99	0.0940	0.1532
55	0.0001	1.11	0.0291	0.0586
55	0.0001	1.455	0.0749	0.1362
55	0.0001	1.99	0.1221	0.1299
55	1e-05	1.11	0.0059	0.0240
55	1e-05	1.455	0.0171	0.0412
55	1e-05	1.99	0.0934	0.1271