Optimization Methods for Machine Learning - Fall 2019

Project # 1 - MLP and Generalized RBF Network

Laura Palagi

Posted on October 20, 2019 - due date November 23, 2019

Instructions

The project will be done in groups formed by 1 to 3 people: each group must hand in their own answers. We will be assuming that, as participants in a Master's Degree course, every single student will be taking the responsibility to make sure his/her personal understanding of the solution to any work arising from such collaboration. Every student is expected to be able to explain to the teacher all the parts of the code submitted by him/her.

Project must be sent by email both to the teaching assistant Marco Boresta (marco.boresta@uniroma1.it) and to laura.palagi@uniroma1.it with subject **[OMML-2019] Project 1**. After your submission, you will receive an acknowledgement email that your project has been received. If you have not received an acknowledgement email within 2 days after your submission, contact the instructors.

The mail must contain as an attachment a .zip or .tar.gz file with both a typed report in English and the source code following instructions in the text of the project. **The report must be of at most 4 pages excluded figures that must be put at the end.**

Evaluation criteria Project is due at latest at midnight on the due date. For late submissions, the score will be decreased. It is worth 85% for the next 48 hours. It is worth 70% from 48 to 120 hours after the due date. It is worth 50% credit after 120 hours delay.
You have three questions Q1, Q2 - both with two points (Q1.1; Q1.2; Q2.1, Q2.2) - and Q3.
You will be given the picture of the function and the data set on October 20, 2019.
Please note that also a test set defined by the teachers will be used to check the overall quality of your results.
The grade is "Italian style" namely in the range [0,30], being 18 the minimum degree to pass the exam. Answering only the first questions allows to get up to 24. Answering questions 1 and 2 allows to get up to 28. Answering Q1, Q2 and Q3 allows to get up to 31 (30 cum laude). Please note that you need to write a last summary report of the project (mandatory) as explained at the end of this document.
The first project accounts for 35% of the total vote of the exam.
For the evaluation of the first project the following criteria will be used:

1. 60% check of the implementation
2. 40% quality of the overall project as explained in the report.

In this project you will implement neural networks to solve a regression problem. Your goal is that of reconstructing in the region $[-2,2]\times[-1,1]$ a two dimensional function $F:\mathbb{R}^2 \to R$ whose picture is sent in a separate file. You do not have the analytic expression of the function, only a data set obtained by randomly sampling 300 points. The data set is

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

You have to randomly split the data set into a training set, a validation set and a test set. Suggested percentages for this split are 70 %, 15% and 15%, respectively, but feel free to experiment and change these numbers. Use one of your matricola numbers as a seed for the random split. (This way, everytime you run your code the dataset is always the same).

Please note that in the last part of the project a test set defined by the teachers is used to check the overall quality of your results.

In the exercises where you need to define a starting point for the optimization procedure, we ask you to fix it randomly using one of your matricula numbers.

Question 1. (Full minimization)
You must construct a shallow Feedforward Neural Network (FNN) (one only hidden layer), either a MLP or a RBF network, that provides the model $f(x)$. 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

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

where the hyper parameter ρ stays in the range $[10^{-5} \div 10^{-3}]$.

1. (max score up to "22") Construct a shallow MLP with a linear output unit, namely

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

where $\omega = (v,w,b)$. The activation function $g(\cdot)$ is the *hyperbolic tangent*

$$g(t) := \tanh(t) = \frac{e^{2\sigma t}-1}{e^{2\sigma t}+1} \tag{2}$$

(g is available in Python with $\sigma = 1$: Numpy.tanh)

The hyper-parameters are

- the number of neurons N of the hidden layer
- the spread σ in the activation function $g(t)$
- the regularization parameter ρ

It must be possible to specify N as an input parameter.

Write a program which implements the regularized training error function $E(v,w,b)$ (as obtained by (1) using $f(x)$ of the MLP network) and uses a Python routine of the optimization toolbox (scipy.optimize) to determine the parameters v_j, w_{ji}, b_j which minimize it. The code must produce a plot of the approximating function found.

Analyse the occurrence of overfitting/underfitting varying the number of neurons N and the parameters ρ and σ (you can also decide to fix $\sigma = 1$ if using the Phyton implementation of Tanh).

Please note that it is not required to evaluate the gradient so derivative-free optimization routine may be applied. Nevertheless using gradient-based method will improve performance and boost up your code.

2. (max score up to "24") Construct a RBF network

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

where $\omega = (v,c)$ $v \in \mathbb{R}^N$ and $c_j \in \mathbb{R}^2$ $j = 1,\dots,N$.

You must choose as RBF function $\phi(\cdot)$ the Gaussian function

$$\phi(\|x - c_j\|) = e^{-(\|x-c_j\|/\sigma)^2} \quad \sigma > 0 \tag{3}$$

The hyper-parameters are

- the number of neurons N of the hidden layer
- the spread $\sigma > 0$ in the RBF function ϕ
- the regularization parameter ρ

It must be possible to specify N and σ as parameters that can be specified as input.

Write a program which implements the regularized training error function of the RBF network $E(v,c)$ (as obtained by (1) using $f(x)$ of the RBF network) and uses a Python routine of the optimization toolbox for its minimization with respect to both (v,c). The code must produce a plot of the approximating function found.

Analyse the occurrence of overfitting/underfitting varying the number of units N, the spread parameters σ and ρ.

In the report, for both Q_{11} and Q_{12} you must state:

- the final setting for N, ρ and σ; how did you choose them and if you can put in evidence over/underfitting and if you can explain why;

- which optimization routine did you use for solving the minimization problem, the setting of its parameters (optimality accuracy, max number of iterations etc) and the returned message in output (successful optimization or others, number of iterations, number of function/gradient evaluations, starting/final value of the objective function, starting/final accuracy etc) if any;

- the initial and final values of the error on the training set;

- the error on the validation and test set.

- the plot of the function representing the approximating function obtained by the MLP and by the RBF networks;

- a comparison of performance between MLP and RBF networks both in terms of quality of approximation (training and test error) and in efficiency of the optimization (number of function/gradient evaluation and computational time needed to get the solution). Please put these values in a table as explained at the end.

Question 2. (Two blocks methods)

1. (max score up to ”26”) Consider again the shallow MLP with linear output unit as defined at Ex. 1 of Question 1. Use the values of N, ρ, σ you fixed at Ex. 1.

   Write a program which implements an *Extreme Learning* procedure, namely one that fix randomly the values of w_{ij}, b_j for all i, j and uses the ”best” Python routine or any other optimization library to minimize the quadratic convex training error $E(v)$ of the only variables $v \in \mathbb{R}^N$.

   As for the random generation of w, b you have no restrictions on the way of proceeding. We suggest to identify a range and repeat the random choice more than once.

   In the report you must state:

   - which optimization routine you use for solving the quadratic minimization problem and the setting of its parameters. Compare the performance of the optimization process w.r.t Full optimization of Question 1 (Ex. 1)

   - the values of the error on the training and test set; Compare these values with the ones obtained by the Full optimization of Question 1 (Ex. 1). Note that, since the hyperparameters are the same of Ex 1., there’s no need of the validation set.

   - the plot of the function representing the approximating function obtained by the Extreme Learning MLP in comparison with the one obtained by Full minimization.

2. (max score up to ”28”)

   Consider again an RBF network as in Ex. 2 of Question 1.

   The number of RBF units N of the hidden layer and the positive parameter σ in the RBF function are those chosen at Ex 2. Question 1.

   Write a program which implements a method with unsupervised selection of the centers. You can select the centers by randomly picking N of the P points of the training set. We suggest to repeat the random choice more than once.

Find the optimal weights by minimizing the regularized error $E(v)$ using a suitable Python routine or any other optimization library to minimize the quadratic convex function.

In the report you must state:

- which optimization routines of the Optimization toolbox you have used to solve the weights subproblems and the setting of its parameters.

- the value of the error on the training and test sets;

- the plot of the function representing the approximating function obtained by the RBF with unsupervised selection of the centers in comparison with the true one.

Question 3. (Decomposition method)

3. (max score up to ”30”)

Consider either the shallow MLP or the shallow RBF network.

Write a program which implements a two block decomposition method, which alternates the convex minimization with respect to the output weights v and the non convex minimization with respect to the other parameters (either (w,b) for the MLP or the centers c for the RBF network respectively).

Set the regularization parameter ρ, number of neurons N and spread σ at the values you selected in Question 1.

You must use appropriate Python routines of the optimization toolbox for solving the two block minimization problems. In this case, it is required that you calculate the gradient of the nonconvex block.

Compare the results with those obtained at the corresponding exercise of Question 1.

In the report you must state:

- which optimization routines of the Optimization toolbox did you use to solve the two block subproblems and the setting of the parameters. Consider to adjust accuracy in optimization with increasing iterations;

- the stopping criteria of the decomposition procedure; consider the use early stopping rule.

- the number of outer iterations (number of subproblems solved), number of function/gradient evaluation and computational time needed to get it.

- the value of the error on the training and test sets;

- the plot of the function representing the approximating function obtained by the RBF with block decomposition in comparison with the true one.

- Comparison with the corresponding network obtained by random selection at the corresponding Exercise of Question 2. Comparison are both in the quality of the network (test error), and in the efficiency of the optimization procedure (number of function/gradient evaluation and computational time needed to get the solution). Please put these values in a table.

Additional bonus exercise. Score up to ”30 cum laude” (or 1 point bonus)
You can use all the 300 samples in the data set to construct the model and you may use different starting points than the one used in the preceding exercises. Submit your best model (either a MLP or RBF network). The teachers will use a new test set (randomly defined and not included within the data set) to check the quality of the results and the bonus point is assigned on the basis of an accuracy ranking.

Final remarks
In the final report (pdf) it is (MANDATORY) to gather into a final table (an example below) the comparison among all the implemented methods - in term of accuracy in learning and computational effort in training.

Ex	FFN	settings	Training error	Validation error	test error	optimization time
Q1.1	Full MLP	$N=?$, $\sigma=?$, $\rho=?$				
Q1.2	Full RBF	$N=?$, $\sigma=?$, $\rho=?$				
Q2.1	Extreme MLP	$N=?$, $\sigma=?$, $\rho=?$				
Q2.2	Unsupervised c RBF	$N=?$, $\sigma=?$, $\rho=?$				
Q2.3	The ONE you chose					

Instructions for python code

You are allowed to organize the code as you prefer **BUT** for each question $ij = \{11, 12, 21, 22, 3\}$ you have to create a different folder where you must provide two files:

- A file called **run_ij_GroupName.py** (e.g. run_11_Example.py, run_12_Example.py). This file will be the only one executed in phase of verification of the work done. It can include all the classes, functions and libraries you used for solving the specific question ij but it has to print ONLY:
  1. Number of neurons N chosen
  2. Value of σ chosen
  3. Value of ρ chosen
  4. Values of other hyperparameters
  5. Optimization solver chosen (e.g. lbfgs, CG,NTC, Nelder-Mead....)
  6. Number of function evaluations
  7. Number of gradient evaluations
  8. Time for optimizing the network (from when the solver is called, until it stops)
  9. Training Error (defined as $E(w,v) = \frac{1}{P_{Train}} \sum_{i=1}^{P_{train}} (y_i - \tilde{y}_i)^2$
  10. Test Error (defined as above but on the test set)

  Please, in order to maintain the code as clean/clear as possible, do not define functions inside the run_ij_GroupName.py file, but make it call functions defined in other files.

- A file with the complete code you have written that includes all the functions that are called from the **run_ij_GroupName.py file**.

  You then have to proved one last extra file, called **Test_ij_GroupName**. In this file you have to read from a new Excel file, called $dataPointsTest.xlsx$. This file will have the same structure of the excel file you received with the Data Points (3 columns with P points each). From this, you will build an array X_{test} and an array Y_{test}. There should then be a function that takes in input the X_{test} array and returns a vector Y_{pred}, which is the prediction returned by the best weights you found in point 1.1. You then have to print the mean squared error between Y_{pred} and Y_{test}.