Optimization Methods for Data Science, 2022-2023

FINAL PROJECT

Veronica Piccialli & Cecilia Salvatore

Submission

Projects must be sent via e-mail to the addresses:

`veronica.piccialli@uniroma1.it`
`cecilia.salvatore@uniroma2.it`

with the object:

OMDS Project Submission

Both members of the team must be included in CC in the email. After the submission, you will receive a confirmation email from the teachers: if you don't receive this email, don't hesitate to get in touch with the teachers to be sure of the reception.

The team must attach the Python files (organized as requested in the instructions at the end of the project) and a typed report (in pdf format) in a compressed folder named "Surname1_Surname2".zip, where Surname1 and Surname2 are the surnames of the members of the team.
The report must be of at most 3 pages per each of the two parts (a total of 6 pages), excluding figures that must be put at the end of each part. Do not include part of the Python coding.

IN EVIDENCE: Evaluation criteria Each part is graded up to 30 points (plus one for the bonus question); the final grade is computed by averaging the two parts. In case you answer fully only one of the two Parts (excluding the bonus), you can reach sufficiency (18,not more).
For the evaluation of the Final 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 training methods for classification problems.

Dataset
You are asked to build a classifier that distinguishes between scan images of capital letters of the English alphabet. It is a good database for trying learning techniques on real-world data while spending minimal effort on preprocessing and formatting. The database is made of a large number of black-and-white rectangular pixels referring to the 26 capital letters in the English alphabet. The character images were based on 20 different fonts, and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts), which were then scaled to fit into a range of integer values from 0 through 15. The columns of the dataset are referred to the following attributes:

Y	capital letter. This is the column of the output y.
A1	horizontal position of box (integer)
A2	vertical position of box (integer)
A3	width of box (integer)
A4	height of box (integer)
A5	total # on pixels (integer)
A6	mean x of on pixels in box (integer)
A7	mean y of on pixels in box (integer)
A8	mean x variance (integer)
A9	mean y variance (integer)
A10	mean x y correlation (integer)
A11	mean of x * x * y (integer)
A12	mean of x * y * y (integer)
A13	mean edge count left to right (integer)
A14	correlation of x-ege with y (integer)
A15	mean edge count bottom to top (integer)
A16	correlation of y-ege with x (integer)

It is necessary to scale the data before the optimization.

You will be given the full dataset that you will need to process to obtain a target set made up of images belonging to two or three classes. The two/three classification tasks have the objective of discriminating between two/three characters in the Y column of the training set. Characters must be identified as follows

- (two-class classification - questions 1, 2, 3 and 4): First letter of the surnames of the members of the team. If they coincide, replace one with the first letter of the name of one member of the team.
- (three-class classification - question 5): Add the first letter of the name of one member of the team.

The characters used must be specified in the report.

You are not given a test set. You can obtain it by randomly splitting the target set into a training set and a test set with a percentage of 80 %, 20%. Use a fixed number as a seed for the random split. **A k-fold cross-validation should be used to set the hyperparameters.**

1 PART 1 - MLP

Write a program implementing a MLP network trained by minimizing the regularized binary cross-entropy error function

$$E(\omega;\pi) = -\frac{1}{P}\sum_{i=1}^{P}\left[y_i \ln(p_i) + (1-y_i)\ln(1-p_i)\right] + \rho\|\omega\|^2 \tag{1}$$

where the hyper parameter ρ stays fixed to 10^{-4} and where y_i is the target value defined in $\{0,1\}$ and $p_i \in [0,1]$ is the Softmax probability for the i-th class. Note that, in this exercise, you only have two classes.

The Softmax is an operator $S : \mathbf{R}^n \to \mathbf{R}^n$ such that, given a vector $v = (v_1,\dots,v_n)$:

$$S(v)_j = \frac{e^{v_j}}{\sum_{h=1}^{n} e^{v_h}} \quad j = 1,\dots,n$$

The activation function $g(\cdot)$ is the *hyperbolic tangent*

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

The hyperparameters are:

- the number H of hidden layers (max. 4) (only for question 1)
- the number of neurons N of the hidden layers

- the spread $\sigma > 0$ in the activation function g (g is available in Python with $\sigma = 1$: Numpy.tanh)

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.

For the identification of the parameters, you must develop:

Question 1. (grade up to 20) an optimization algorithm for the minimization with respect to (w, v, b) that uses the gradient (please note that **it is forbidden to use an automatic differentiation tool**).

Question 2. (grade up to 10) an RBF neural network trained by implementing the **decomposition method** studied in class.

Remarks for the report - Part 1

In the report you must state:

1. the final setting for H, N and σ; how did you choose them and if you can put in evidence over/under fitting and if you can explain why;
2. which optimization routine you used 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;
3. the initial and final values of the regularized error on the training set; the average value of the error on the validation set; the final value of the test error.
4. a comparison of the performances of the two optimization methods implemented (number of function/gradient evaluations, computational time needed to get the solution, errors on the training and test set). Please put these values in a table at the end of the report.

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 terms of accuracy in learning and computational effort in training.

Ex		settings				Final	Final	optimization
		H	N	σ	ρ	train error	test error	time
Q1.1	Full MLP							
Q2.1	RBF							

2 PART 2 - SVM

Consider a nonlinear SVM with kernel $k(\cdot,\cdot)$ and the corresponding nonlinear decision function

$$y(x) = sign\left(\sum_{i=1}^{L} \lambda_i y^i k(x^i, x) + b\right)$$

where λ, b are obtained as the optimal solution of the dual nonlinear SVM problem.

As kernel function you can use a Gaussian Kernel $K(x,y) = e^{-\gamma\|x-y\|^2}$ or a Polynomial Kernel $K(x,y) = (x^T y + 1)^p$ where p is a hyper-parameter, which should be set using k-fold cross-validation.

You are requested to train the SVM, namely to both set the values of the hyperparameters C and p with a heuristic procedure and to find the values of the parameters λ, b with an optimization procedure. For this last task, you need to follow the different optimization procedures described in the following questions.

Question 3. (grade up to 15) Write a program to find the solution to the SVM dual quadratic problem. Apply any procedure for identifying the values of the hyperparameters C and γ. To find the solution of the SVM dual quadratic problem, **you should use a specific method for convex optimization, such as CVXOPT**.

Question 4. (grade up to 15) Fix the dimension of the subproblem to $q = 2$ and implement the most violating pair (MVP) decomposition method, which uses the analytic solution of the subproblems.

BONUS Question 5. (+ 1 point) Consider a three classes problem with the classes of letters chosen. Implement a SVM strategy for multiclass classification (either one against one or one against all).

Remarks for the report - Part 2

- Only for Question 3: the final setting for the hyperparameter C (upper bound of the constraints of the dual problem) and of the hyperparameter of the kernel chosen; how you have chosen them and if you could identify values that highlight over/underfitting;
- Only for Question 3: which optimization routine you use for solving the quadratic minimization problem and the setting of its parameters, if any;
- For each Question:
    - machine learning performances: write the value of the accuracy on the training and test set; (for question 3, write also the value of the validation accuracy, if computed).

- optimization performance: report the initial and final value of the objective function of the dual problem, the number of iterations, the number of function evaluations, and the violation of the KKT conditions, either as it is returned by the optimization routine used or evaluated by yourselves.

The comparison among all the implemented methods - in terms of accuracy in learning and computational effort in training - must be gathered into a final table (an example below)

	hyperparameters		ML performance		optimization performance			
question	C	γ	Training accuracy	Test accuracy	number its	number fun evals	KKT viol	cpu time
Q3								
Q4								
Q5								

3 Instructions for Python code

You are allowed to organize the code as you prefer **BUT** for each question, you have to create a different folder where **you must provide the files according to the following instructions.**

3.1 Part 1 - MLP

- For each question, you have to provide a file called **Functions_ij_surname1_surname2**. This file should only include all the classes, functions and libraries you used for solving the specific question ij.
- A file called **run_1i_surname1_surname2.ipynb**. This file should include the import of all the functions you need, 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 (if any)
  5. Optimization solver chosen (e.g. L-BFGS, 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 in question (1), without regularization term.
  10. Test Error (defined as above but on the test set)

3.2 Part 2 - SVM

- For each question, you have to provide a file called **Functions_ij_surname1_surname2**. This file should only include all the classes, functions and libraries you used for solving the specific question ij.
- A file called **run_2i_surname1_surname2.ipynb**. As in part 1, it has to print ONLY:
  - Setting values of the hyperparameters
  - classification rate on the training set (% instances correctly classified)
  - classification rate on the test set (% instances correctly classified)
  - The confusion matrix
  - Time necessary for the optimization procedure
  - number of optimization iterations
  - final difference between $m(\lambda)$ and M(λ) (only for question 5)
  - final value of the objective function of the dual SVM problem

3.3 General Remarks concerning Python code

Note that:

1. Each question will be graded by averaging the Performance score (25%), the Efficiency score (25%) and the Mathematical Correctness score (50%).
2. There is no competition with other students. Your grade does not depend on other students' performances.
3. You are warmly recommended to write clean, commented and easily readable code. Not respecting general assignments (for example, printing different values or not printing requested outputs) will result in penalization.
4. Your code will not be debugged and will not be modified **in ANY case** after the submission. **If the code does not run, the resulting score will be zero**. The code will be executed in Google Colab, so you are warmly recommended to check that your code can successfully execute on this platform.
5. It is welcomed to take inspiration from other students, as well as from open-source contents online, but **"copy and paste" projects will be not considered valid**. Be aware that **your code will be cross-checked with online materials, as well as with other students' code**.
6. The use of any library automatically building neural models (e.g, Pytorch, TensorFlow, sklearn.neuralnetwork etc.) is **FORBIDDEN** and automatically invalidates the project.