Optimization Methods for Machine Learning - Fall 2018

FINAL PROJECT

Laura Palagi

Posted on December 20, 2019- due date: Exams' date

In this project you will implement training methods for classification problems.

Data sets.
You are asked to build a classifier that distinguishes between images of Zalando's articles that belong to different classes. Each sample is a 28x28 grayscale image, associated with a label. **It is necessary to scale the data before the optimization**.

You will be given a full dataset and you will extract the target set made up of

- (two class classification . question 1 and 2) images of pullovers and coats (identified by the labels 2 and 4, in the target set, respectively)
- (three class classification - question 3) images of pullovers, coats and shirts (identified by the labels 2, 4, and 6 in the target set, respectively).

Instruction on how to import the data in Python will be provided in a separate file.

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 one of your matricola numbers as a seed for the random split. A $k-$fold cross validation can be used to set the hyperparameters. Please note that the teacher has defined a test set that might be used to check the accuracy of your models.

Question 1. (Two class classification)

Write a program (please provide the sources) which implements an RBF network which is trained by minimizing the regularized error function

$$E(w) = \frac{1}{2P}\sum_{i=1}^{P}\left(\sum_{j=1}^{N} w_j\phi(\|x^i - c_j\|) - y^i\right)^2 + \frac{\rho}{2}\|w\|^2 + \frac{\rho}{2}\|c\|^2,$$

where $\rho > 0$ is a regularization parameter fixed to the value $\rho = 10^{-4}$. As RBF function $\phi(\cdot)$ you must use the the Gaussian function

$$\phi(\|x - c_j\|) = e^{-(\|x-c_j\|/\sigma)^2} \quad r > 0 \tag{1}$$

with derivative

$$\nabla_{c_j}\phi_(\|x-c_j\|) = \frac{2}{\sigma^2} e^{-(\|x-c_j\|/\sigma)^2}(x-c_j)$$

The hyper-parameters are

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

Analyse the occurrence of overfitting/underfitting varying the number of neurons N and of the parameters σ and use a k-fold cross validation for setting the best values.

For the identification of the parameters (w, c) you must develop:

1. an optimization algorithm for the minimization with respect to both (w, c) that uses the gradient (please note that if you provide a function that evaluated the gradient, the performance of the optimization method will improve); use appropriate Python routines of the optimization toolbox for the minimization with respect to both (w, c).
2. a two block decomposition method which alternates the minimization with respect to weights and centers. Use appropriate optimization routines for solving the two minimization problems respectively with respect to centers c and to weights w. You need to implement an early stopping criterion.

In the report you must state:

1. the final setting for N, ρ 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 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;
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 evaluation, computational time needed to get the solution, errors on the training and test set). Please put these values in a table at the end.

Question 2. (Two class classification)

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

$$y(x) = sign\left(\sum_{p=1}^{P} \alpha_p y^p k(x^p, x) + b\right)$$

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

As a kernel you must use a Gaussian Kernel

$$K(x,y) = e^{-\gamma\|x-y\|^2} \qquad \gamma > 0$$

The hyper-parameters are

- the spread $\gamma > 0$ in the Gaussian Kernel
- the penalty coefficient $C > 0$

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

1. a standard QP algorithm for the solution of the dual QP problem
2. a decomposition method with $q = 2$ for the dual quadratic problem. You must define the selection rule of the working set, and use either a standard QP algorithm for the solution of the subproblem or the analytic solution of the subproblem.

In the report you must state:

1. the final setting for C and for the kernel parameter and if you can put in evidence over/under fitting;
2. which optimization routine did you use for solving the optimization 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 objective function; the average value of the error on the validation set; the final value of the test error.
4. a comparison of performance of the two optimization methods implemented (number of function/gradient evaluation and computational time needed to get the solution). Please put these values in a table at the end.

Question 3. (multi-class classification) Implement a multiclass SVM classificator using either one-against-all strategy or one-against-one. Use the code developed in Question 1 or 2 as a tool for the solution of the two class subproblems.

Instructions for python code

You are allowed to organize the code as you prefer. For each question you can create as many files as you want but among the files you must provide:

Question 1

Two files, called `run_1_1_GroupName.py` and `run_1_2_GroupName.py`. These files will be the only one executed in phase of verification of the work done. They can include all the classes, functions and libraries you used for solving the question but they have to print **only**:

- Number of neurons N
- Initial Training Error (defined as E(w;v) = $\frac{1}{2P_{Train}}\sum_{i=1}^{P}(y_i - \tilde{y})^2)$
- Final Training Error
- Final Test Error (defined as above but on the test set)
- Norm of the gradient at the final point
- Optimization solver chosen (e.g. bfgs, CG,NTC, Nelder-Mead....)
- Total number of function evaluations
- Total number of gradient evaluations
- Time for optimizing the network (from when the solver is called, until it stops)
- Value of σ
- Value of ρ

Please, in order to maintain the code as clean/clear as possible, do not define functions inside the `run_1_1_GroupName.py` and `run_1_2_GroupName.py` files, but make them call functions defined in other files.

Question 2

Two files, called `run_2_1_GroupName.py` and `run_2_2_GroupName.py`. These files will be the only one executed in phase of verification of the work done. They can include all the classes, functions and libraries you used for solving the question but will have to print **only**:

- Misclassification rate on the training set
- Misclassification rate on the test set
- Time for finding the best weights(from when the solver is called, until it stops)

- Value of γ and C

Furthermore, the file `run_2_1_GroupName.py` must print also

- Norm of the gradient at the final point
- Optimization solver chosen (e.g. bfgs, CG,NTC, Nelder-Mead....)
- Total number of function evaluations
- Total number of gradient evaluations

While the file `run_2_2_GroupName.py` must print also:

- Number of iterations needed to find the optimal point (how many times you consider two blocks of variables)
- The value of the violation of the optimality conditions (norm of the gradient in the unconstrained case or KKT violation in the constrained case)

Question 3

A file, called `run_3_GroupName.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 question but it has to print **only**:

- How you decided to solve the classification task (one-against-one / one-against-all)
- Optimization solver (bfgs etc.. or other decomposition techniques)
- Misclassification rate on the training set
- Misclassification rate on the test set
- Time for finding the best weights(the sum of the time needed for optimizing each model)

Further instructions: 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.

Include in the code **all** the things you have done (e.g: the code for the cross validation, grid search etc.) **but** make sure they won't be executed during the run.