Optimization Methods for Machine Learning - Fall 2018

FINAL PROJECT

Laura Palagi

Posted on December 13, 2018- due date: Exams' date

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

Data set.

Classification of normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The images here have been deslanted and size normalized, resulting in 16 x 16 grayscale images (Le Cun et al., 1990).

The training data are available as separate files per digit (0 1 2 3 4 5 6 7 8 9). Each line consists of the 256 grayscale values. You are receiving only the data set for the numbers 1 2 8.

The training and test observations, are distributed as follows:

number	0	1	2	3	4	5	6	7	8	9	Total
Train	1194	1005	731	658	652	556	664	645	542	644	7291
Test	359	264	198	166	200	160	170	147	166	177	2007

The test set can be notoriously "difficult", and a 2.5% error rate is excellent.

The data set must be divided into a training set and a test set (choose percentage of training data between 70-80%). Report the dimension of the training set that you used.

For the two class classification report which of the digits you choose as data set among 1 2 8.

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 to be chosen. 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
- the regularization parameter ρ

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 (to be provided by you); 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.

Please note that as optimization routine you can also use a minimization algorithm developed by yourself.

Modify the optimization routines 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 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 2. (Two class classification)

Implements a nonlinear SVM for classification which is trained by solving the QP dual formulation. As a kernel you must use a Gaussian Kernel

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

For the definition of the decision function of the SVM

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 kernel used;
2. the final setting for C and for the kernel parameter and if you can put in evidence over/under fitting;
3. 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;
4. the initial and final values of the objective function; the final value of the test error.
5. 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 classificator (1,2,8 digits) 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 it has 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 it has to print **only**:

- Which numbers you decided to classify
- 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

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)
- A boolean (True/False) which states if the final point is optimal (you need to check the KKT optimality conditions in the special form for the SVM problem)

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)