LINMA 2471 – Optimization models and methods II

Homework III-IV – First-order and second-order methods for prediction (second part)

[v1.1]

This homework is dedicated to the implementation of first-order methods and the use of conic model with a second-order solver with the goal of performing a classification task that consists in identifying handwritten digits.

The data for this homework comes from the MNIST database of handwritten digits (Le Cun, Cortes and Burges), which is provided in MATLAB file `Homework-3-data.dat` (see also function `display_digits.m` for visualization, and file `Homework-3-data-information.txt` for information about the format).

The total length of your report should not exceed ten pages. The deadline for submitting your report and all accompanying files (source code) in a single zip file is **Thursday December 21**.

The following late homework policy will be used for all homeworks: you can use during the whole semester, without justification, up to *two days of extension* for the homework deadlines (i.e. you can be two days late for one homework, or one day late for two homeworks). This policy also covers unforeseen events (computer breakdown, sickness, etc.).

A. Implementation and test of a first-order method for empirical loss minimization

In this first part we consider the problem of finding a linear classifier that is able to separate two sets of points (or *patterns*) in $\mathbb{R}^n$, corresponding to two categories of data that one wishes to distinguish from each other. Denote those two sets by $A = \{a_i\}_{1 \leq i \leq n_a}$ and $B = \{b_i\}_{1 \leq i \leq n_b}$. For this homework, each point corresponds to a 28×28 grayscale image of a handwritten digit, i.e. $n = 28^2$.

A linear classifier is simply a linear function $x \mapsto \ell(x) = h^T x + c$ (where $h \in \mathbb{R}^n$ and $c \in \mathbb{R}$), and one wishes to find h and c such that all points from A satisfy $\ell(a_i) > 0$ and all points from B satisfy $\ell(b_i) < 0$.

One way to find such a linear classifier consists in performing the so-called *empirical loss minimization*, which consists in minimizing the following objective function

$$\min_{h \in \mathbb{R}^n, c \in \mathbb{R}} \ \frac{1}{n_A} \sum_{1 \leq i \leq n_A} f(\ell(a_i)) + \frac{1}{n_B} \sum_{1 \leq i \leq n_B} f(-\ell(b_i))$$

where $f(x)$ is a suitable loss function (see previous homework). Note the presence of normalizing weights, that are useful if n_A and n_B are very different.

In addition, one observes that performance of the classifier is usually improved if the problem is regularized in order to prevent solutions where h has a large norm. This can be done in two ways:

1. add the extra term $\frac{\lambda}{2}\|h\|^2$ to the objective function (where λ is a positive parameter)
2. add an extra constraint $\|h\| \leq R$ to the problem (where R is a positive parameter)

In the section you will implement four different methods:

1. subgradient method on the hinge loss function with λ-regularized objective
2. gradient method on the logistic loss function with R-bounded variable
3. gradient method on the logistic loss function with λ-regularized objective
4. accelerated gradient method on the logistic loss function with λ-regularized objective

(pick suitable values for λ and R ; using $h = 0$ and $c = 0$ for the starting point).

Should you need to compute a bound on the Lipschitz constant of the gradient of a function of the type $g(y) = f(A^T y + c)$, you can use the easily proven fact that if the Lispschitz constant of f is equal to L_f, then the constant for g is at most equal to $L_g = L\ \lambda_{\max}(AA^T)$ (where the maximal eigenvalue term is also equal to $\|A\|_2$, see also MATLAB's `norm` and `normest` functions).

For your tests you will use the training set of the provided MNIST database: more specifically you will use (a subset of) the 0 digits for set A and (a subset of) the 1-9 digits for set B (this is known as the *one-vs-rest* technique).

Compare the performance of those four methods from the optimization point of view (in particular the speed of convergence). At this stage you will not comment on the classification performance. You are free to select one or several suitable comparison frameworks, e.g. test for a fixed number of iterations, or a fixed computational budget, or a given final objective accuracy.

Use graphs, and display the theoretical worst-case bound. You can vary the number of points in sets A and B (e.g. use the same percentage of the total number of available points).

Comment and try to explain the observed behaviors. Based on this comparison, identify the best performing method and try to explain why it performs better.

B. Implementation and test of conic second-order model for SVM.

In this second part we will build a conic second-order model that implements the so-called Support Vector Machine (SVM) technique. That model will be solved using an external solver (based on a second-order interior-point method).

We start with one of the equivalent models for maximum-margin linear separation as seen during the SeDuMi lab[1]:

[1] Only the objective is slightly modified using a squared norm, which does not change the problem.

$$\min ||h||^2 \text{ such that } h^T a_i + c \geq 1 \ \forall \ 1 \leq i \leq n_a \text{ and } h^T b_i + c \leq -1 \ \forall \ 1 \leq i \leq n_b \ .$$

Question: what is the main drawback of this model when trying to solve a practical separation problem ?

To address that drawback one needs to introduce a nonnegative *slack* variable for each point, which leads to the following model (where λ is again a positive parameter):

$$\min \frac{\lambda}{2}||h||^2 + \sum_{1\leq i\leq n_A} s_i + \sum_{1\leq i\leq n_B} t_i \text{ such that } \begin{cases} s_i \geq 0, \ h^T a_i + c \geq 1 - t_i \ \forall \ 1 \leq i \leq n_a \\ t_i \geq 0, \ h^T b_i + c \leq -1 + s_i \ \forall \ 1 \leq i \leq n_b \end{cases} .$$

Formulate this problem as a conic optimization problem. Implement and solve it (using SeDuMi or MOSEK) on the MNIST database, using the same sets A and B as in section A. Report and comment your results.

C. Comparison of a first-order method and a second-order method for classification of handwritten digits.

In this last part, you will compare the best performing first-order method identified in section A. and the SVM technique tested in section B. We are mostly interested in the generalization error. This means that after training a classifier using one of the two methods on the training set you will report the percentage of errors made by this classifier on the testing set.

You will again use the *one-vs-rest* approach, which means that you will need to train a different classifier for each of the ten digits, and propose a procedure to aggregate the output of those ten classifiers on a given test image into a single label prediction for that image.

Report and comment your results.

After taking all your observations into account, you will identify what you believe is your best version of the classifier and package it as a separate MATLAB file that can be run independently (without any external dependency, apart possibly from loading a data file). This classifier will be submitted as a zip file on Moodle, following the template given in file `final_predict.zip` on Dropbox. In particular, the main function will be called `final_predict_groupXX.m` and will return two outputs: one containing the predicted digit, and one containing a size-10 vector that expresses your confidence in each of the digits from 0 to 9.

Changelog. [v1.0, 2017-11-30] initial release [v1.1, 2017-12-13] typos corrected, hint about Lipschitz constant, instructions for final "best" classifier submission