Optimization for Data Science Homework 1
Regularized Logistic Regression problem

Caria Natascia - 1225874
Cozzolino Claudia - 1227998
Petrella Alfredo - 1206627

Master degree in Data Science, Department of Mathematics, University of Padova

Abstract

The unconstrained optimization problem resulting from the Regularized Logistic Regression for the '4 VS 9' handwritten digit binary classification task has been numerically solved using three different methods: Gradient Descent (GM), Stochastic Gradient Descent (SG) and Stochastic Variance Reduced Gradient Descent (SVRG). The results of the different strategies have been then compared and discussed in terms of accuracy and CPU time.

1 Optimization problem definition

The task of interest is to perform binary classification given a training set

$$T = \{(x^i, y^i),\ x^i \in X \subseteq \mathbf{R}^n,\ y^i \in \{-1,+1\} \text{ and } i = 1, ..., m\}.$$

In other words, the aim is to learn a rule function f such that, given $x \in X$, the corresponding correct label y can be predicted as $f(x)$. According to the Logistic Regression the target function can be defined as $f(x, w) = \text{sign}(x^T w)$ and then the problem can be formularized as finding the parameter $w \in \mathbf{R}^n$ that minimize the logistic loss. Finally considering the regularized version with regularization term $\lambda > 0$, the binary classification task turns into the following optimization problem

$$\text{(RLR)} \qquad \min_{w \in \mathbf{R}^n} h(w) = \sum_{i=1}^{m} \left(\log\left(1 + \exp(-y^i w^T x^i)\right) + \frac{\lambda}{2m}\|w\|^2 \right)$$

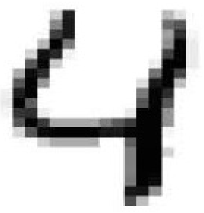
for which numerical solutions will be presented and discussed below.

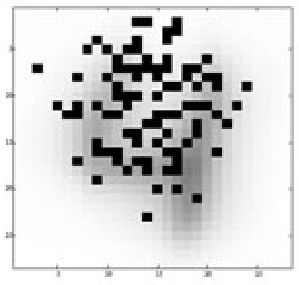
2 Dataset presentation

The numerical experiments have been conducted on the public GISETTE dataset, available on UCI archive. It stores records for an handwritten digit recognition problem, in particular only focusing on separating the highly confusible digits '4' and '9'. It consists of 6000 instances and 5000 attributes, not all useful for the aim of digits recognition. Indeed, the dataset was constructed from the MNIST data and modified for the purpose of the NIPS 2003 feature selection challenge. In particular, pixels were sampled at random in the middle top part of the feature containing the information necessary to distinguish '4' from '9' and higher order features were created as products of these pixels, leading the problem to a higher dimensional feature space. A number of distractor features having no predictive power were also added and the order of the features and patterns was randomized.

2.1 Data Preprocessing

Among the 5000 starting features, 45 were found to have zero values for each entry so they have been dropped from the dataset. In order to improve the performance of the methods, data has been standard-scaled to have zero mean and unit variance. Even if GISETTE dataset is made up of attributes having no predictive power, no feature selection process has been performed here, since we are


(a) Digit labelled as 4.


(b) Randomly selected subset of pixels in the region of interest.

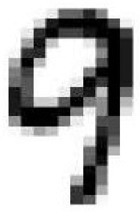

(c) Digit labelled as 9.

Figure 1. Two examples of digits from the MNIST database and and an example of a randomly selected subset of pixels in in the middle top part of the feature containing the information necessary to distinguish the digit.

interested in carrying out the numerical experiments on a high dimensional dataset, both in terms of number of instances and in terms of number of attributes. Finally, the dataset has been split into train and test set (30% of the initial one) in order to verify the accuracy of the applied methods.

3 Numerical methods

The unconstrained optimization problem (RLR) has been numerically solved using the following descent methods:

- Gradient Descent method with fixed step size;
- Stochastic Gradient Descent;
- Stochastic Variance Reduced Gradient Descent.

3.1 Hyper-parameters choice

Strictly speaking, the three algorithms have been implemented in MATLAB (see code in GitHub) choosing the hyper-parameters with a trial and error approach. The table below shows the final selected values for each algorithm (with the corresponding variable name in the implementation).

Table 1. Hyper-parameters variable names and values for each algorithm

Method	regularization term λ	step size factor	maxit	epochs length
GM	$\text{reg}_{gm} = 10^{-2}$	$\text{L}_{gm} = 10^{7}$	$\text{maxit}_{gm} = 50$	
SG	$\text{reg}_{sg} = 10$	$\text{LC}_{sg} = 10^{-3}$	$\text{maxit}_{sg} = 10^{4}$	
SVRG	$\text{reg}_{svrg} = 10$	$\alpha_{svrg} = 10^{-2}$	$\text{maxit}_{svrg} = 10^{4}$	$\text{eplen}_{svrg} = 2500$

In detail the methods depend on:

- $\boldsymbol{\lambda}$, the **regularization term** in the (RLR) formulation; this complexity penalty has been chosen big enough to improve generalization and avoid underfitting. Note that for the SG and SVRG methods, due to the fact that at each iteration only one component of the gradient is used to update the search direction, the regularization factor is about m times bigger, where m is the number of instances in the training set.
- The **step size factor**. In the case of the GM, it is set to a big value in order to overestimate the actual Lipschitz constant and ensure the fastest possible convergence (linear) of the method. For the SG the decreasing step size is of the form $\sqrt{\frac{\text{LC}_{sg}}{(1+\text{it})}}$, which has turned out to be better w.r.t. the same value squared. Regarding the SVRG, the step size can be chosen fixed and greater with respect to

the others thanks to the stabilizing effect of the full gradient computation at the beginning of each epoch.
- **maxit**, which fixes the maximum number of iterations that could be performed. Note that, in our particular case, since the dataset dimensions are quite large, we haven't implemented other stopping conditions besides the one due to overflow errors, so it coincides with the actual number of iterations. In fact, it is not reasonable to use as stopping criteria the value of the gradient norm, which remains far from zero, nor an upperbound for the target value of the loss since we would have needed a deeper mathematical knowledge of the problem.
- moreover, just for the SVRG method, the **epochs lenght** determines every how many iterations a new $\widetilde{w}$ is pinned and the relative full gradient is calculated (it is obviously chosen among all the divisors of $maxit_{svrg}$).

3.2 Starting point

As iterative methods, GM, SG and SVRG require a guessed starting point for w. Many initializations have been tried, such as the all-zeroes and different random generated vectors. Again adopting an euristic *modus operandi* the best choice turned out to be a min-max normalized vector of random elements from the Standard Normal distribution. In this way all the components w_i belongs to the interval $[-1, +1]$, a reasonable assumptions after the standardization of the dataset features.

4 Results discussion

In this section the obtained results using the hyper-parameters in table 1 are presented and summarized in table 2. The methods are compared in terms of CPU time, loss function, square norm of the full gradient and accuracy scores.

Table 2. Numerical results in terms of CPU time, loss function, square norm of the full gradient, accuracy scores.

Method	iter	CPU time	Loss	$\|\|\nabla L\|\|^2$	Train acc	Train F1	Test acc	Test F1
GM	50	12.550	3.24×10^4	2.95×10^8	0.48	0.47	0.46	0.46
SG	10^4	3.249	4.84×10^3	2.39×10^6	0.92	0.92	0.88	0.89
SVRG	10^4	2.170	3.18×10^3	6.05×10^5	0.99	0.99	0.96	0.96

4.1 Accuracy measures definition

As said in the previous pages, the GISETTE dataset has been split for the train and the test phases. The models accuracy has been then calculated on both the set of instances according to the following score measures:

- **Precision** $\text{prec} = \dfrac{\text{TP}}{(\text{TP+FP})}$;
- **Recall** $\text{rec} = \dfrac{\text{TP}}{(\text{TP+FN})}$;
- **Accuracy** $\text{acc} = \dfrac{\text{TP+TN}}{(\text{TP+TN+FP+FN})}$;
- **F1 score** $\text{F1} = 2\dfrac{\text{prec} \times \text{rec}}{\text{prec} + \text{rec}}$;

where TP, TN, FP, FN are respectively the number of true positive, true negative, false positive, false negative predicted values.

4.2 Graphical results

To avoid costly computations at each iteration such as the current accuracy or the loss function and the full gradients in the stochastic methods a *rate* parameter is defined (chosen among the divisors of *maxit*). In general it controls every how many iterations to update or save different possible variables according to the algorithm. Note that while this strategy can be efficient in time, it is largely memory consuming; in particular, after the weight vector w is updated a number of times equal to *rate*, the current one is stored in order to then compute, at the end of the algorithm, the accuracy improvements over time. As a matter of fact the graphical plots have been then produced using only a uniform sampling of the measures of interest, loss and accuracy. Finally, for a better visualization, the plotting window has been limited to 10 seconds for the x axis.

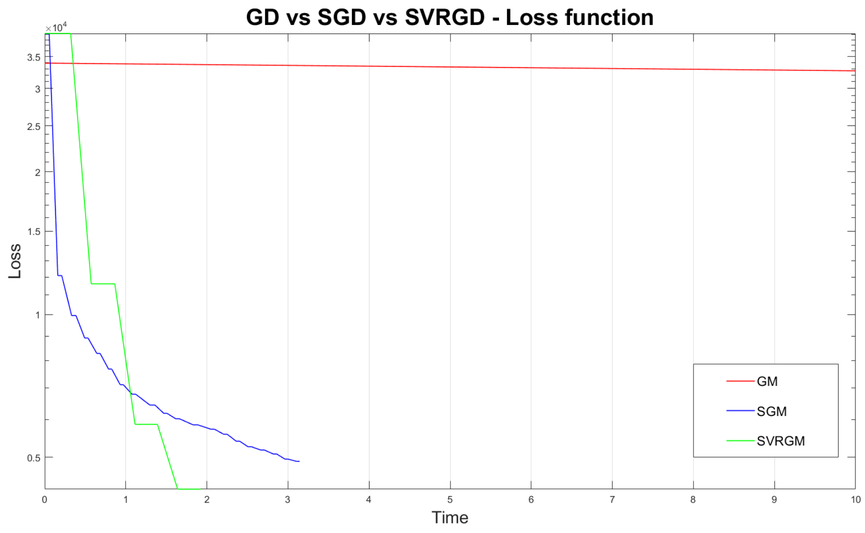


Figure 2. Comparison between the loss function decrease during time in the training phase of each method.

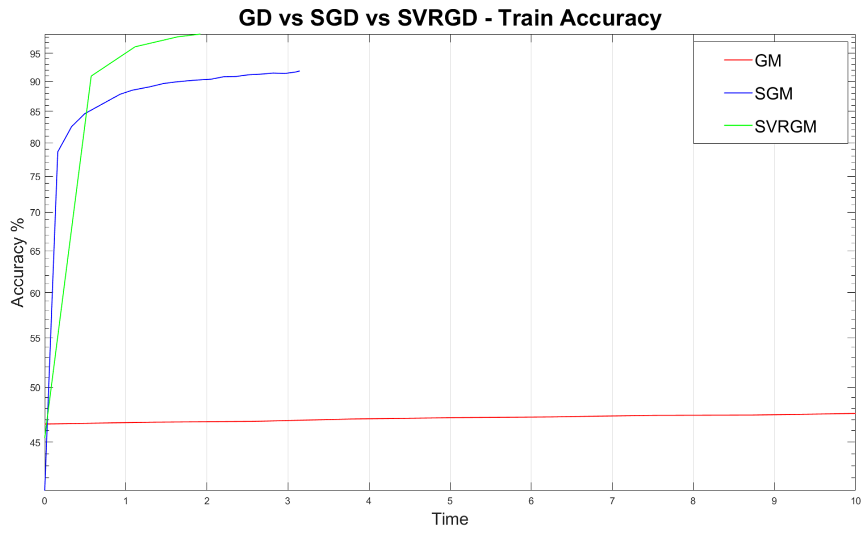


Figure 3. Comparison between the three methods' classification accuracy behavior (in percentage) as time increases during the training phase.

4.3 Results analysis

The huge difference in terms of cost per operation between the GM and the stochastic methods made difficult for us noticing the advantages of the linear convergence rate of the classic descent method. On the other hand, it allowed us to compute many more iterations for the SG and SVRG, still at a cheaper CPU time cost, leading to definitely better final accuracy. In particular, as expected, the gradient method performs worse than the others: it takes more than 10 seconds to execute 50 iterations, reaching a train accuracy of 0.48 against the approximately 2-3 seconds needed for SG and SVRG to execute 10^4 iterations for achieving a train accuracy respectively of 0.92 and 0.99.
Given the evidence of overfitting highlighted by the comparison with the test accuracy, we tried to raise the regularization terms, without evident improvements. Therefore, we decided to run again the random methods alone with a new set of hyper-parameters which fully showed their effectiveness. In particular it was possible to avoid 2000 iterations of the SG method without any significant loss variation and to discover that with an $eplen_{svrg}$ parameter which is half the maximum number of iterations the accuracy keeps improving (without overfitting!) up to values around 2×10^4 for *maxit* and 10^4 for $eplen_{svrg}$, reaching a 97% accuracy over the test set and a 98% over the train set.