PTML 5: 05/05/2022

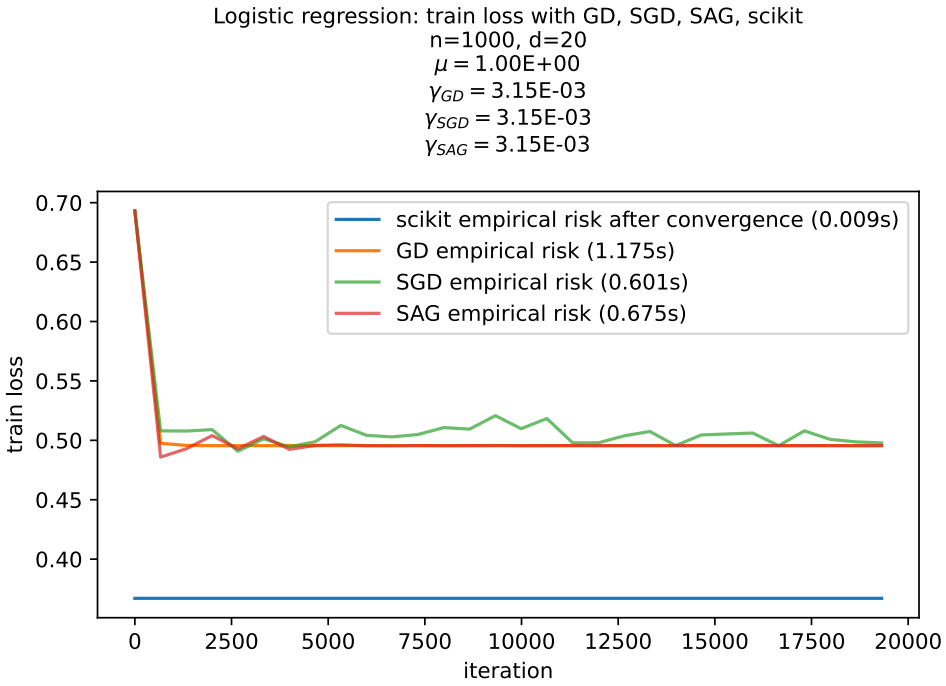


TABLE DES MATIÈRES

INTRODUCTION

Section 1 is more fondamental while section 3 is more applied. You can start with either one.

1 STOCHASTIC AVERAGE GRADIENT

1.1 Context

In TP4 and TP3 we studied Gradient descent (GD), Stochastic gradient descent (SGD), Stochastic gradient descent with averaging (SGA), and some variants of GD such as Heavy Ball. In this section we will implement another important variant of SGD, namely *Stochastic Average Gradient* (SAG).

SAG belongs to **variance reduction methods** [Schmidt et al., 2013,]. The goal of variance reduction methods is to gather the advantages of GD (convergence speed in terms of the number of algorithm iterations) and SGD (low computational cost of one iteration, and more precisely a cost that does not depend on n).

SAG keeps in memory and updates an estimation of the full gradient, built by averaging the previous estimates. The expected value of the estimate is still the batch gradient, but the variance is smaller than for SGD. See also : SAGA, an extension of SAG [Defazio et al., 2014].

To be fully understood and optimized, these methods require some tuning and preparation, which will depend on the considered problem. However, we can implement the core SAG method with a constant step-size on our example (Logistic regression).

1.2 Exercise

Exercice 1 : Implement basic SAG (we use the same setting as in TP4).
You can find the algorithm in [Schmidt et al., 2013], page 10, the pdf is in the TP folder. Try to read the abstract of this paper, the beginning of he introduction (or obviously more if you wish).
Files :

— **TP5_LR_SAG.py** : main file, launches the algorihms and sets some hyperparameters such as the learnin rate.

— **TP5_LR_algorithms.py** : contains the algorithms, you just need to edit **SAG()**.

— **TP5_utils.py** : utils.

Find a setting where SAG converges faster than SGD. As always, you can generate different data and choose the dimensions. In this exercise you can use constant learning rate but is you wish you are free to experiment with more complex schedules as in TP4. It is actually possible to use SAG with a constant step size, as discussed in [Schmidt et al., 2013,].
For instance, some parameters to experiment with include

— n

— d

— standard deviation of the distribution that generates the data

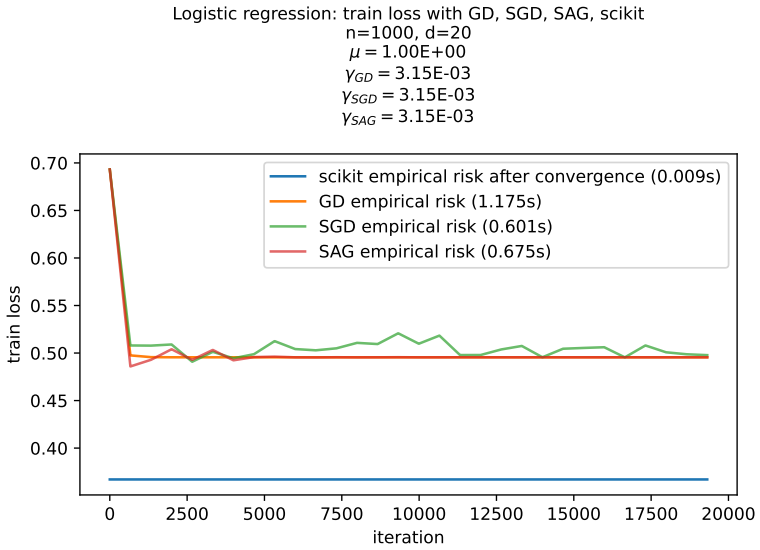
— the learning rates

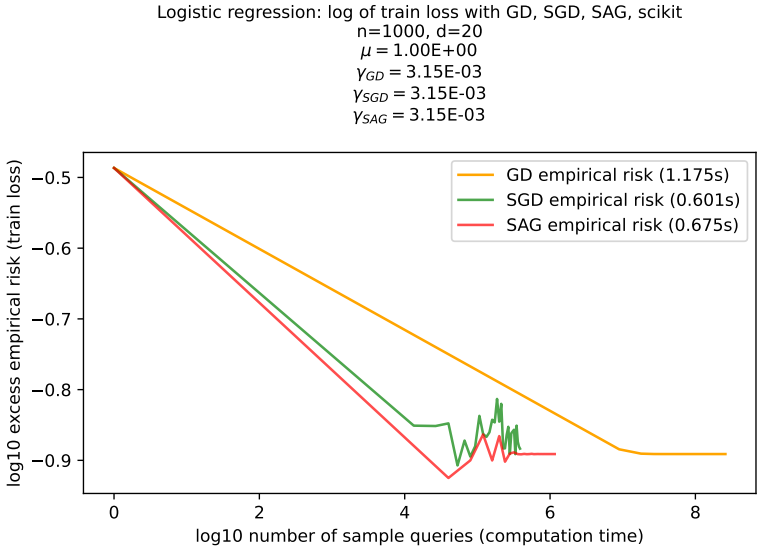
Exercice 2 : What is the computationnal complexity of one SAG update ?

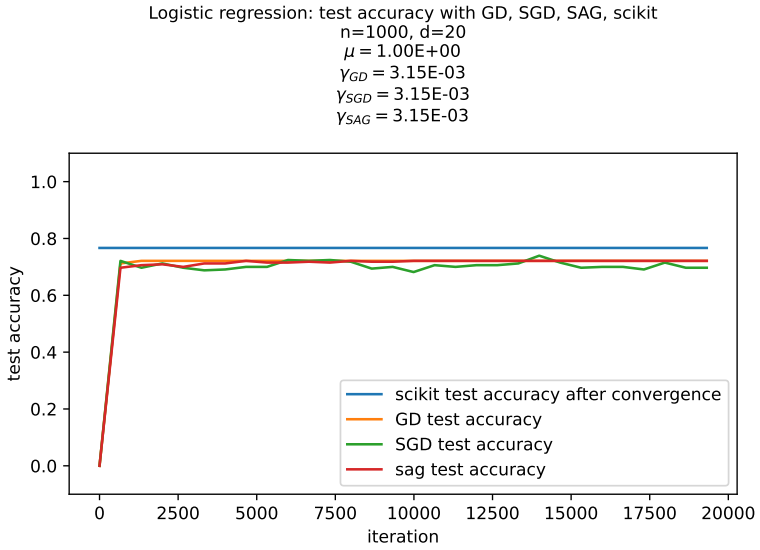
1.3 Examples

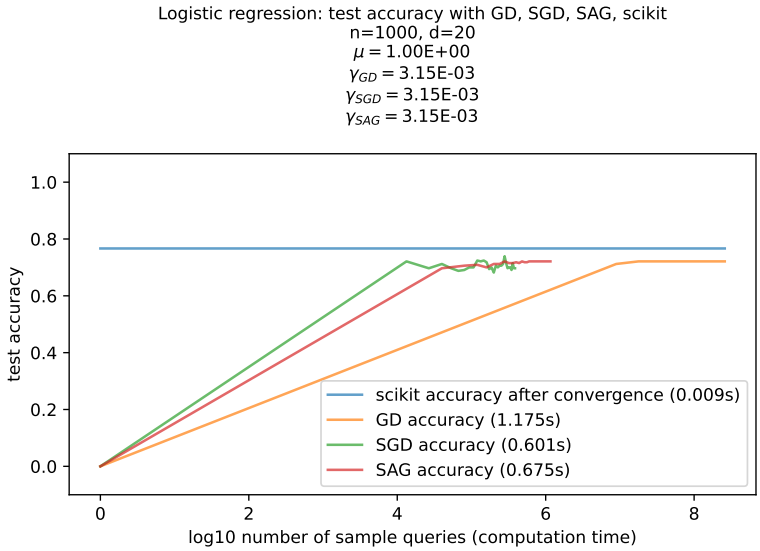
In this example, the training loss decreases slightly faster with SAG. Note that since SGD and SAG involve randomness, the curves will change if you launch the algorithms several times. The advantage of using SAG will be more significant on lar-

ger datasets, as the one studied in the experimental section of [Schmidt et al., 2013,].









1.4 Extensions of SAG

As explained in [Schmidt et al., 2013,], the practical implementation of SAG can be improved in several ways :

— better choice of the learning rate, with a line search type method.

— non-uniform sampling

— regularization

— use of mini-batches

In the paper, the authors use one of these variants called SAG-LS (SAG Line Search) and compare it to other classical algorithms (GD, SGD, AFG, etc) on several benchmark datasets. SAG-LS performs best among these algorithms on most of these datasets (see section 5 of the paper).

2 VIRTUAL ENVIRONMENTS

When coding in python, it is a good practice to use **virtual environments**. For each project, you can set the required packages independently from other projects. For instance from a console, the following lines will create a virtual environment called **env** in the current folder and after actication, it will install the packages listed in **requirements.txt**.

— python3 -m venv env

— source env/bin/activate

— python3 -m pip install -r requirements.txt

https://docs.python.org/fr/3/library/venv.html
You can exit the environment by typing "deactivate".

3 DIGITS DATASET

The goal of this section is to manipulate scikit-learn methods by applying them to a simple dataset. With the files **TP5_RF_digits.py** and **TP5_LR_digits.py**, perform a classification on the **digits** dataset (that is loaded in the files), with a random forest classifier and a logistic regression, respectively. Experiment with the different parameters. Print the confusion matrix and the classification report (find some information in the scikit documentation).

Useful links : https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html
You can do the same work with a different dataset too.

RÉFÉRENCES

[Defazio et al., 2014] Defazio, A., Bach, F., and Lacoste-Julien, S. (2014). SAGA : A fast incremental gradient method with support for non-strongly convex composite objectives. Advances in Neural Information Processing Systems, 2(January) :1646–1654.

[Schmidt et al., 2013] Schmidt, M., Le Roux, N., and Bach, F. (2013). Minimizing finite sums with the stochastic average gradient. Mathematical Programming, 162(1-2) :83–112.