MLoN Computer Assignment 2

Group 1

Consider

$$f(\boldsymbol{w}) = \frac{1}{N} \sum_{i \in [N]} f_i(\boldsymbol{w}) + \lambda \left\| \boldsymbol{w} \right\|_2^2 \tag{1}$$

1

Question
Solve the optimization problem using GD, SGD, SVRG, and SAG.
Solution
In the "Greenhouse gas observing network" dataset, there is in total 2921 data samples. Each data sample contains 16 sequences. Each sequence represents a time series of GHG tracers data. For the sake of simplicity, in our problem definition, we define the last value of the last sequence as our target y and the rest part as feature input X. As shown in the homework 2.1 answer, we know that the gradient of logistic ridge regression loss function can be calculated as:

$$\bigtriangledown f(\boldsymbol{w}) = 2\lambda \boldsymbol{w} - \frac{1}{N} \sum_{i \in [N]} \frac{y_i \boldsymbol{x}_i}{1 + e^{y_i \boldsymbol{w}^T \boldsymbol{x}_i}} \tag{2}$$

Where X is 5231-D vector and y is a scalar. The input features are L2 normalized before throwing to the model. We implement the optimization algorithms in Python. In the implementation of stochastic algorithms, we calculate the stochastic vector g by randomly selecting one training samples instead of using large batch size. We show the result with different configurations in the next answer. We initialize the weight vector w from a gaussian distribution with $\mu = 0$ and $\sigma = 0.1$.

2

Question
Tune a bit the hyper-parameters (including λ)
Solution
In the following table, we changed α and λ for these four different solvers, and showed the convergence time, loss after convergence and number of iterations.

Solver	configuration	convergence time (sec)	loss	iter nums
GD	$\alpha = 0.01$, $\lambda = 0.1$	17.17	0.0508	2479
SGD	$\alpha = 0.01$, $\lambda = 0.1$	24.99	0.0508	2563
SVRG	$\alpha = 0.01$, $\lambda = 0.1$, $T = 2000$	3.32	0.0505	3 (outer loop)
SAG	$\alpha = 0.01$, $\lambda = 0.1$	19.00	0.0503	2383
GD	$\alpha = 0.1$, $\lambda = 0.1$	1.81	0.0507	247
SGD	$\alpha = 0.1$, $\lambda = 0.1$	2.65	0.0508	259
SVRG	$\alpha = 0.1$, $\lambda = 0.1$, $T = 2000$	3.51	0.0505	3
SAG	$\alpha = 0.1$, $\lambda = 0.1$	4.22	0.0507	350
GD	$\alpha = 0.5$, $\lambda = 0.1$	0.45	0.0872	48
SGD	$\alpha = 0.5$, $\lambda = 0.1$	0.77	0.0872	52
SVRG	$\alpha = 0.5$, $\lambda = 0.1$, $T = 2000$	3.54	0.0872	4
SAG	$\alpha = 0.5$, $\lambda = 0.1$	3.99	0.0510	359
GD	$\alpha = 0.01$, $\lambda = 0.01$	94.48	0.0242	13291
SGD	$\alpha = 0.01$, $\lambda = 0.01$	137.13	0.0242	13332
SVRG	$\alpha = 0.01$, $\lambda = 0.01$, $T = 2000$	9.42	0.0232	7
SAG	$\alpha = 0.01$, $\lambda = 0.01$	139.88	0.0221	13440
GD	$\alpha = 0.01$, $\lambda = 0.001$	129.10	0.0334	18551
SGD	$\alpha = 0.01$, $\lambda = 0.001$	190.76	0.0338	18341
SVRG	$\alpha = 0.01$, $\lambda = 0.001$, $T = 2000$	15.23	0.0321	13
SAG	$\alpha = 0.01$, $\lambda = 0.001$	151.12	0.0338	18797
GD	$\alpha = 0.01$, $\lambda = 0.0001$	28.64	0.0146	3837
SGD	$\alpha = 0.01$, $\lambda = 0.0001$	30.873	0.0156	2951
SVRG	$\alpha = 0.01$, $\lambda = 0.0001$, $T = 2000$	3.60	0.0130	2
SAG	$\alpha = 0.01$, $\lambda = 0.0001$	2.058	0.0114	257
GD	$\alpha = 0.01$, $\lambda = 0.00001$	26.84	0.0098	3878
SGD	$\alpha = 0.01$, $\lambda = 0.00001$	27.77	0.0112	2455
SVRG	$\alpha = 0.01$, $\lambda = 0.00001$, $T = 2000$	2.33	0.0039	2
SAG	$\alpha = 0.01$, $\lambda = 0.00001$	2.100	0.0095	269

3

Question
Compare there solvers in terms of complexity the hyper-parameter tuning, convergence time, convergence rate, and memory requirement.

Solution

As we can see from the table above, in terms of the convergence time, most of the time SVRG has the least training time to achieve convergence. As for the rest, GD and SGD have similar performance at convergence time, but SAG is not very stable: sometimes outperforms GD and SGD a lot, yet sometimes slightly worse. And in terms of the convergence rate (consider outer loop iterations), SVRG has outperformed others a lot.

In terms of complexity of the hyper-parameter tuning, since the SVRG mostly gives the least time to converge, the complexity of tuning hyper-parameters is relatively low. As for the GD, SGD, and SAG, it could take a lot of time to converge if the hyper-parameters are not that good.

As for the memory usage, it not only depends on the theoretical usage, but also depends highly on the implementation. But for SAG, at least a large memory of size N (number of samples) is need to store the gradient for every data sample.