Hindawi
Scientific Programming
Volume 2021, Article ID 9991859, 11 pages
https://doi.org/10.1155/2021/9991859

Research Article

Fast Training Logistic Regression via Adaptive Sampling

Yunsheng Song, Xiaohan Kong, Shuoping Huang, and Chao Zhang

College of Information Science and Engineering, Shandong Agricultural University, Tai'an 271018, Shandong, China

Correspondence should be addressed to Yunsheng Song; sys_sd@126.com

Received 6 March 2021; Revised 3 May 2021; Accepted 15 May 2021; Published 30 May 2021

Academic Editor: Pengwei Wang

Logistic regression has been widely used in artificial intelligence and machine learning due to its deep theoretical basis and good practical performance. Its training process aims to solve a large-scale optimization problem characterized by a likelihood function, where the gradient descent approach is the most commonly used. However, when the data size is large, it is very time-consuming because it computes the gradient using all the training data in every iteration. Though this difficulty can be solved by random sampling, the appropriate sampled examples size is difficult to be predetermined and the obtained could be not robust. To overcome this deficiency, we propose a novel algorithm for fast training logistic regression via adaptive sampling. The proposed method decomposes the problem of gradient estimation into several subproblems according to its dimension; then, each subproblem is solved independently by adaptive sampling. Each element of the gradient estimation is obtained by successively sampling a fixed volume training example multiple times until it satisfies its stopping criteria. The final estimation is combined with the results of all the subproblems. It is proved that the obtained gradient estimation is a robust estimation, and it could keep the objective function value decreasing in the iterative calculation. Compared with the representative algorithms using random sampling, the experimental results show that this algorithm obtains comparable classification performance with much less training time.

1. Introduction

Supervised learning is to train a learner with a labelled training set and correctly determine the outputs for the unseen instances [1]. As one of the most famous classification algorithms in supervised learning, logistic regression learner is a generalized linear regression model in which the output is discrete [2]. Logistic regression has been widely used in various kinds of applications owing to its good performance, as process tomography [3], customer churn prediction [4], spatial prediction [5], major chronic diseases and clinical risk prediction [6, 7], and so on [8–10]. The process of training a logistic regression model aims to solve an unconstrained convex optimization problem, where gradient descent is one of the most important solutions [11]. Because of the computation of the gradient using the whole training instances, it is very time-consuming to use gradient descend (GD) when the data size is large [12].

To speed up GD, many improved algorithms have been developed. According to the volume of data to obtain the gradient estimation, these algorithms can be divided into two groups: stochastic gradient descent and batch gradient descent [13]. Stochastic gradient descent (SGD) uses only one randomly selected training example to compute the gradient, and this can be very efficient for large datasets [14]. So, SGD is much faster and very suitable for online learning. However, the estimated gradient obtained by SGD is difficult to be a descent direction at each iteration, so that it needs a vast number of iterations. Furthermore, SGD is difficult to be suitable for the parallel environment [15].

Different from SGD, batch gradient descent (BGD) obtains the aim gradient estimation using randomly choosing a certain amount of training examples. In this way, BGD could largely reduce the error and instability of the estimation, and it also obtains an effective solution [16–18]. As the sampled examples play an important role in estimating the gradient, BGD needs to carefully choose an

appropriate sample size before sampling. However, it is difficult to predetermine an appropriate sample size for different datasets. Furthermore, samples of the same size could vary in terms of their qualities, because some examples are more representative or resembling the original data than others [19].

This paper presents an improved adaptive sampling (AS) algorithm for accelerating the logistic regression training process. This method firstly gives a rule for estimating the gradient by some examples, and the obtained gradient estimation can guarantee that the objective function value keeps decreasing in the iterative calculation of GD. Then, the problem of obtaining an appropriate vector that meets the rule can be decomposed into several subproblems, where each subproblem determines a component of the vector that satisfies the stopping rule. Finally, the examples are drawn successively from the training set into the sample, and it terminates as soon as each component of the estimated gradient over the obtained sample satisfies its own rule. To speed up this process, the estimated components satisfying their own stopping rules are not estimated in the subsequent iteration, and they are the corresponding components of the final estimate of the gradient. The main contributions of this paper are as follows:

(1) Giving the rule to judge whether the direction of a vector is a descent direction of the current objective function or not, it is critical for the execution efficiency of the gradient descend method.

(2) Providing an adaptive sampling method to overcome the difficulty of the predetermining sample size before sampling, this method can adaptively determine the sample size according to the character of datasets and avoid the influence of human subjective factors.

(3) Applying a strategy of divide-and-conquer to efficiently obtain the gradient estimation on the sampled examples, the aim gradient vector estimation problem is divided into several one-dimensional estimation subproblems, and each subproblem can be solved independently.

(4) Proving the obtained gradient estimation is robust using probably approximately correct theory, and this estimation could be a descent direction of the current objective function at each iteration.

(5) Designing an efficient mechanism to solve the multivariate estimation problem for large-scale data.

The rest of the paper is organized as follows: Section 2 reviews related methods according to their characteristic. Section 3 proposes a sampling on-demand algorithm for logistic regression and proves its effectiveness. Section 4 reports the experimental results through the comparison with existing methods. Section 5 gives the conclusion of this paper and shows some future research work.

2. Related Work

Related work in improving GD has been widely developed nowadays. According to the amount of data to obtain the gradient estimation, existing GD algorithms can be divided into two groups: SGD algorithm and BGD algorithm.

The original SGD (OSGD) algorithm computes the gradient with only one sample from the training set. The OSGD algorithm does not consider the effect of different dimensions on its convergence, so its rate of convergence could be slow when the surface of the objection function curves steeply for different dimensions. Qian [20] proposed the Momentum algorithm to an accelerated OSGD algorithm, where the current update vector is appended with a fraction of the obtained vector in the previous iteration. To efficiently solve the sparse data, Duchi et al. [21] proposed an Adagrad method to deal with the online learning task, and it set different learning rates for different components of the vector. Besides, there are lots of algorithms to determine the adaptive learning rate for different components [22–24]. These kinds of algorithms can deal with large-scale data, they must perform a vast number of iterations before an appreciable improvement of the objective function, and it is difficult to parallelize.

Different from the SGD algorithm, the BGD algorithm computes the gradient with some randomly sampled examples from the training set at each iteration. So, the BGD algorithm can reduce the variance of the estimate of the gradient, and it achieves more stable convergence. In order to sufficiently often achieve convergence to the optimal solution, the estimate of the gradient by the BGD algorithm needs to enforce descent in the objective function at every iteration. Therefore, the sample size is carefully determined. Byrd et al. [25] proposed a dynamic sample gradient (DSG) algorithm, which can dynamically determine the sample size before sampling. For the convex optimization problem, the DSG algorithm can get the optimal solution. However, the sample size determined by the DSG algorithm could increase with the increasing steps, so that the total running time of the DSG algorithm also increases. Furthermore, owing to the fact that samples of the same size could vary in terms of their qualities, this leads to the estimate of the gradient with the fixed-size sample that may not enforce descent in the objective function. On the other hand, choosing a proper learning rate is an important issue for the performance of the BGD algorithm. A smaller learning rate could cause the convergence rate to become slower, but a larger one fluctuates obviously around the optimal solution. Robbins and Sutton [26] proposed a schedule to select the appropriate learning rate during training, where the predefined schedule is conducive to reducing the learning rate. Liang et al. [27] have proposed a sampling on-demand to speed up logistic regression, but the theoretical proof about its robust result is not given, and it does not compare its classification performance with other state-of-the-art approaches. There are many similar algorithms to yield high accuracy in the solution of the optimization problem [28].

3. Main Content

3.1. Preliminary. The logistic regression classier is generated using the posterior probabilities of two labels ({0, 1}) denoted by a linear function in $x \in R^m$, where the sum of these two posterior probabilities is one. The form of this model is that

$$\log \frac{P(y=1|x)}{P(y=0|x)} = \omega^\top z, \tag{1}$$

where the weight vector $\omega = (\omega_1, \omega_2, \ldots, \omega_{m+1})^\top \in R^{m+1}$, $z = (1, x)^\top \in R^{m+1}$, $\omega^\top z$ is the inner product between ω and z, y is the label of x, and m is the dimensional size.

Let TS $= \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ be a training set, each training instance $x_i (i = 1, 2, \ldots, N)$ is represented by a m-dimensional vector $(x_{i1}, x_{i2}, \ldots, x_{im})^\top$, and its label is $y_i \in \{0, 1\}$. Therefore, the optimal weight vector is obtained by minimizing the following problem:

$$\min_{\omega \in R^{m+1}} -\frac{1}{N} \sum_{i=1}^{N} \{y_i \omega^\top z_i - \log(1 + \exp(\omega^\top z_i))\}, \tag{2}$$

where $z_i = (z_{i1}, z_{i2}, \ldots, z_{i,m+1})^\top = (1, x_{i1}, \ldots, x_{im})^\top$. Clearly, the objective function $J(\omega) = -1/N \sum_{i=1}^{N} \{y_i \omega^\top z_i - \log(1 + \exp(\omega^\top z_i))\}$ is a strictly convex function. Let ω^* be the optimal solution of the optimization problem (2), and the classifier of logistic regression is $I_{(f(z),0.5)}$:

$$I_{(f(z),0.5)} = \begin{cases} 1 & \text{if } f(z) \geq 0.5, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $f(z) = 1/(1 + \exp(\omega^{*\top} z))$. With these notations, the predicted label $\widehat{y}$ for a given test instance $x_0 = (x_{01}, x_{02}, \ldots, x_{0m})$ can be derived from the function $I_{(f(z_0),0.5)}$, where $z_0 = (1, x_{01}, x_{02}, \ldots, x_{0m})^\top$.

The GD algorithm is an iterative optimization algorithm, where it updates the current solution with the solution solved in the previous step and the gradient of the current objective function each time. Owning to its advantage of simple implementation and effectiveness, GD is widely adopted to solve the unconstraint optimization problem (2). Let ω^k be the optimal solution at the kth iteration. The gradient estimation $\nabla J(\omega^k)$ is obtained as follows:

$$\begin{aligned} \nabla J(\omega^k) &= \left(\nabla J_1(\omega^k), \nabla J_2(\omega^k), \ldots, \nabla J_{m+1}(\omega^k)\right) \\ &= \left(\frac{1}{N}\sum_{i=1}^{N} \nabla l_1(\omega^k, z_i, y_i), \ldots, \frac{1}{N}\sum_{i=1}^{N} \nabla l_{m+1}(\omega^k, z_i, y_i)\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} \left(\nabla l_1(\omega^k, z_i, y_i), \ldots, \nabla l_{m+1}(\omega^k, z_i, y_i)\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} \nabla l(\omega^k, z_i, y_i), \end{aligned} \tag{4}$$

where
$\nabla l_j(\omega^k, z_i, y_i) = -(\exp(\omega^{k\top} z_i)/1 + \exp(\omega^{k\top} z_i) - y_i) z_{ij}$, $j = 1, 2, \ldots, m+1$. Problem (2) is a convex optimization problem and the obtained solution is the global optimal solution. The computation of the gradient estimation needs all the data, so that it is very time-consuming for large-scale data. To overcome the deficiency, we propose a novel algorithm for fast training logistic regression via adaptive sampling (LLR-AS).

3.2. LLR-AS Algorithm. In fact, the LLR-AS algorithm is also a type of minibatch gradient descent algorithm. It randomly samples a subset from T to compute the gradient estimation $\nabla \widehat{J}(\omega)$ to replace $\nabla J(\omega)$ computed with all the data in every iteration. According to the feature of the GD algorithm, its convergence efficiency is closely related to the quality of the gradient estimation $\nabla J(\omega)$. The value of $J(\omega)$ is difficult to be smaller all the time and the final execution time becomes longer if the objective function value cannot keep decreasing using the gradient estimation in each iteration. Because gradient is the fastest direction in which the current objective function value decreases, a similar direction between $\nabla J(\omega)$,$\nabla J(\omega)$, and $\nabla \widehat{J}(\omega)$ should be considered to achieve this aim. In the following, a rule is given to obtain an estimate $\nabla \widehat{J}(\omega)$ with high quality. This results in the following inequality:

$$\left(\sqrt{(1-\varepsilon)}\, \|\nabla J(\omega)\|_2\right)^2 \leq \nabla \widehat{J}(\omega)^\top \nabla J(\omega), \tag{5}$$

where $\|X\|_2$ is a norm of the vector X and the relaxation parameter $0 \leq \varepsilon < 1/2$. According to the conclusion [29], the vector $-\nabla \widehat{J}(\omega)$ must keep the objective function $J(\omega)$ value decreasing in the iterative calculation of GD if it satisfies inequality (5). Moreover, parameter ε controls the descent speed for the function $J(\omega)$. The larger the value of ε, the smaller the directional derivative, and the lower the descent speed for the function $J(\omega)$. Because the optimization problem (2) is convex, the LLR-AS algorithm can get the globally optimal solution. With the rule, our algorithm is outlined in Algorithm 1.

3.3. AS Algorithm. In order to get the vector $\nabla \widehat{J}(\omega)$ meeting inequality (5), a novel adaptive sampling algorithm is proposed in the following: let $T = \{(z_1, y_1)(z_2, y_2), \ldots, (z_N, y_N)\}$ be the new dataset generated by the set TS, where $z_i = (1, x_{i1}, \ldots, x_{im})^\top$, $i = 1, 2, \ldots, N$. According to inequality (5), the simplest way of getting the vector $\nabla \widehat{J}(\omega)$ is to sample a subset S from the set T and obtain an estimate from $\nabla J(\omega; S) = 1/|S| \sum_{(z_i, y_i) \in S} \nabla l(\omega^k, z_i, y_i)$ as $\nabla \widehat{J}(\omega)$, where $|S|$ is the size of subset S. However, the sample size is difficult to determine for different tasks though there exist some theoretical and empirical results in the literature such as PAC [30] learning theory and learning curves [31]. Moreover, the theoretical results are usually worst-case and learning curves are average-case, so they are not necessarily consistent with each other [19]. To tackle this difficulty, we propose an adaptive sampling algorithm. Our method obtains the aim subset S by continually sampling examples from T until the gradient estimation on the sampled subset S satisfying the stopping rule. It decides the sample size

Input: Dataset TS = $\{(x_1, y_1), (x_1, y_2), \ldots, (x_N, y_N)\}$, the stepsize $\lambda > 0$.
Output: The optimal vector ω^*.
(1) Initialize: $\omega^0 \in R^{m+1}$ and $k = 0$
repeat
(2) Obtain the vector $\nabla\widehat{J}(\omega_k)$ satisfying inequality (5)
(3) $\omega^{k+1} = \omega^k - \lambda\nabla\widehat{J}(\omega_k)$
(4) $k \longrightarrow k + 1$
until a convergence test is satisfied;
(5) **Return** $\omega^* = \omega^k$

Algorithm 1: LLR-AS algorithm.

through the information of the sampled examples and solves the difficulty of predetermining the sample size. Therefore, the key issue becomes the problem that how to design the stopping rule for our adaptive sampling to satisfy inequality (5).

From a statistical point of view, the estimation of gradient $\nabla J(\omega)$ satisfying inequality (5) is a $m+1$-dimensional vector estimating problem. However, the existing sampling procedures mainly focus on a one-dimensional estimate problem, and they cannot be directly applied to the multidimensional problem. Although there exists a close relationship between the components of the gradient $\nabla J(\omega)$, each component of $\nabla J(\omega)$ can still be seen as a one-dimensional estimating subproblem. Therefore, the multivariate estimation problem (5) can be solved by solving these one-dimensional estimating subproblems. Inequality (5) is equivalent to $\sum_{j=1}^{m+1}\{\nabla J_j(\omega; S) \times \nabla J_j(\omega) - (\sqrt{1-\varepsilon}\nabla J_j(\omega))^2\} \geq 0$ according to the formula of vector inner product. In other words, inequality (5) must hold if each component $\nabla J_j(\omega; S)$ of $\nabla J_j(\omega; S)$ simultaneously satisfies its own inequality $(1-\varepsilon)(\nabla J_j(\omega))^2 \leq \nabla J_j(\omega; S) \times \nabla J_j(\omega)$, where $j = 1, 2, \ldots, m+1$. So, the problem of seeking for the gradient estimation satisfying inequality (5) can be approximatively divided into $m+1$ subproblems, where each subproblem is solved by $\nabla J_j(\omega; S) \times \nabla J_j(\omega) - (1-\varepsilon)(\nabla J_j(\omega))^2 \geq 0$ at the same time.

Each component $\nabla J_j(\omega)$ of $\nabla J(\omega)$ can be considered as the exception of the population T, and $\nabla J_j(\omega, S)$ is the estimation of $\nabla J_j(\omega)$ on the subset S sampled from the population T. According to the central limit theorem, the difference of value between $\nabla J_j(\omega, S)$ and $\nabla J_j(\omega)$ continually becomes small with the enlarging sampled subset S. There exists a critical value of the sample subset S size for satisfying inequality $(1-\varepsilon)(\nabla J_j(\omega))^2 \leq \nabla J_j(\omega; S) \times \nabla J_j(\omega)$ with a large probability. Therefore, both the early stopping rule and consecutive sampling are adopted to get the sampled subset S and estimation over it. Given the aim vector ω and the objective function $J(\omega)$, each element $\nabla J_j(\omega)$ can be a constant value and its upper bound of the absolute value could be estimated by a function $\alpha_j(t, s) = d_j\sqrt{\ln(6(m+1)/\delta)/(2(t^3 s^3))}$ $(j = 1, 2, \ldots, m+1)$, where $d_j = 2(|\min_{1\leq i\leq N} z_{i,j}| + |\max_{1\leq i\leq N} z_{i,j}|)$ and t and s are the size of sampled subset per sampling and total number of sampling, $0 < \delta < 1/2$). In the next section, we will show that such a function can achieve this goal, and the stopping condition can be finally transformed into $|\nabla J_j(\omega; S)| \geq (1 + 1/\varepsilon)\alpha_j(t, s)$ for each component $\nabla J_j(\omega)$. However, it needs to recompute $\nabla J(\omega; S)$ on the sampled subset S and test whether all the components $\nabla J_j(\omega; S)$ satisfy their own stopping rule during each round of sampling, and this computational burden could cost much more execution time to achieve this aim. Two improvements are made to solve this problem (Algorithm 2).

Let $\widetilde{S}_{t-1}$ be the cumulatively sampled subset obtained by the first $t-1$ iterations sampling. The set $\widetilde{S}_t = \widetilde{S}_{t-1} \cup S_t$, where S_t is the sampled subset at the t−th iteration. We have

$$\begin{aligned}
\nabla J\left(\omega; \widetilde{S}_t\right) &= \frac{1}{\left|\widetilde{S}_t\right|} \sum_{(z_i, y_i)\in\widetilde{S}_t} l(\omega, z_i, y_i) \\
&= \frac{1}{\left|\widetilde{S}_t\right|} \sum_{(z_i, y_i)\in\widetilde{S}_{t-1}} l(\omega, z_i, y_i) + \frac{1}{\left|\widetilde{S}_t\right|} \sum_{(z_i, y_i)\in S_{t-1}} l(\omega, z_i, y_i) \\
&= \frac{\left|\widetilde{S}_{t-1}\right|}{\left|\widetilde{S}_t\right|} \left\{ \frac{1}{\left|\widetilde{S}_{t-1}\right|} \sum_{(z_i, y_i)\in\widetilde{S}_{t-1}} l(\omega, z_i, y_i) \right\} \\
&\quad + \frac{1}{\left|\widetilde{S}_t\right|} \sum_{(z_i, y_i)\in S_{t-1}} l(\omega, z_i, y_i) \\
&= \frac{\left|\widetilde{S}_{t-1}\right|}{\left|\widetilde{S}_{t-1}\right| + |S_t|} \nabla J\left(\omega; \widetilde{S}_t\right) + \frac{1}{\left|\widetilde{S}_{t-1}\right| + |S_t|} \sum_{(z_i, y_i)\in S_{t-1}} l(\omega, z_i, y_i).
\end{aligned} \tag{6}$$

The computation of $\nabla J(\omega; \widetilde{S}_t)$ can be largely reduced using the result on the set $\widetilde{S}_{t-1}$ at each iteration. On the other hand, we adopt an asynchronous way to get each component of $\nabla J(\omega; \widetilde{S})$. If one or more components satisfy their own stopping rules, then they are directly the corresponding components of the final result without considering in subsequent iteration. Algorithm 2 describes the multivariate adaptive sampling method.

3.4. The Effectiveness of AS Algorithm. In this subsection, we study the effectiveness of AS algorithm. To derive our main result, we need the following lemmas and theorems.

Lemma 1. *Let $A_1, A_2, \ldots, A_{m+1}$ be $m+1$ random events. For any positive integer m,*

Input: Dataset $T = \{(z_1, y_2), (z_1, y_2), \ldots, (z_N, y_N)\}$, a weight vector ω, parameters $\varepsilon \in (0, 1/2), \delta \in (0, 1/2)$.
Output: The vector $\nabla\widehat{J}(\omega) = \left\{\nabla\widehat{J}_1(\omega), \nabla\widehat{J}_2(\omega), \langle/p\rangle \ldots, \nabla\widehat{J}_{m+1}(\omega)\right\}$.
(1) Initialize: $\Lambda = \{1, 2, \ldots, m+1\}$, $\widetilde{S}_0 = \varnothing$, $t = 0$
(2) Compute $d_j = 2(|\min_{1\le i\le N} z_{i,j}| + |\max_{1\le i\le N} z_{i,j}|)$, where $j = 1, 2, \ldots, m+1$
while $|\Lambda| \ge 1$ **do**
(3) $t = t + 1$
(4) Sample a random subset S_t with the size of s from the set T, $\widetilde{S}_t = \widetilde{S}_{t-1} \cup S_t$
for each $j \in \Lambda$ **do**
(5) Compute $\nabla J_j(\omega; \widetilde{S}_t)$
if $|\nabla J_j(\omega; \widetilde{S}_t)| \ge (1 + 1/\varepsilon) d_j \sqrt{\ln((6m+6)/\delta)/(2s^3t^3)}$ **then**
(6) $\Lambda = \Lambda - \{j\}$
(7) $\nabla\widehat{J}_j(\omega) = \nabla J_j(\omega; \widetilde{S}_t)$
end
end
end
(8) **Return** $\nabla\widehat{J}(\omega) = \left\{\nabla\widehat{J}_1(\omega), \nabla\widehat{J}_2(\omega), \langle/p\rangle \ldots, \nabla\widehat{J}_{m+1}(\omega)\right\}$

Algorithm 2: AS algorithm.

$$P\left(\bigcap_{i=1}^{m+1} A_i\right) \ge \sum_{i=1}^{m+1} P(A_i) - m. \tag{7}$$

Proof. The lemma is proved by mathematical induction.

(1) Consider $m = 1$ [32]. We have $A_1 \cap A_2^c \subseteq A_2^c$ and $P(A_1 \cap A_2^c) \le P(A_2^c)$ for $m = 1$, where A^c is the complementary set of the set A. Meanwhile, $P(A_1 \cap A_2) + P(A_2^c) \ge P(A_1 \cap A_2) + P(A_1 \cap A_2^c) = P(A_1)$. So, $P(A_1 \cap A_2) \ge P(A_1) - P(A_2^c) = P(A_1) - (1 - P(A_2)) = P(A_1) + P(A_2) - 1$.
(2) Assume that the inequality holds for $m = k - 1$; that is, $P(\cap_{i=1}^{k} A_i) \ge \sum_{i=1}^{k} P(A_i) - k + 1$.
(3) Consider $m = k$. We have

$$\begin{aligned} P\left(\bigcap_{j=1}^{k+1} A_i\right) &= P\left(\left(\bigcap_{j=1}^{k} A_i\right) \cap A_{k+1}\right) \\ &\ge P\left(\bigcap_{j=1}^{k} A_i\right) + P(A_{k+1}) - 1 \\ &\ge \left(\sum_{j=1}^{k} P(A_j) - k + 1\right) + P(A_{k+1}) - 1 \\ &\ge \sum_{j=1}^{k+1} P(A_j) - k. \end{aligned} \tag{8}$$

The lemma follows immediately from mathematical induction. □

Lemma 2. *Let $\widetilde{S} \subseteq T$ be a subset, which is obtained through independent and random sampling from the training set T with the size of n. For any given weight vector ω, $\lambda \ge 0$ and $j \in \{1, 2, \ldots, m+1\}$, we have*

$$P\left(\left|\nabla J_j(\omega; \widetilde{S}) - \nabla J_j(\omega)\right| \ge d_j\lambda\right) \le \exp\left(-2n\lambda^2\right). \tag{9}$$

Proof. According to $\nabla l_j(\omega, z_i, y_i) = 1/1 + \exp(\omega^\top z_i - y_i) z_{i,j}$ and $y_i \in \{0, 1\}$, we have $\min\{a_j, -b_j\} \le \nabla l_j(\omega, z_i, y_i) \le \max\{-a_j, b_j\}$, where $a_j = \min_{1\le i\le N} z_{i,j}$, $b_j = \max_{1\le i\le N} z_{i,j}$, $j = 1, 2, \ldots, m+1$. Moreover,

$$\begin{aligned} &\max_{1\le i\le N}\left\{\nabla l_j(\omega, z_i, y_i)\right\} - \min_{1\le i\le N}\left\{\nabla l_j(\omega, z_i, y_i)\right\} \\ &\le \max\{-a_j, b_j\} - \min\{a_j, -b_j\} \\ &\le \max\{b_j - a_j, -2a_j, 2b_j\} \\ &\le 2\left(|a_j| + |b_j|\right) \\ &= d_j. \end{aligned} \tag{10}$$

Therefore, for any $j \in \{1, 2, \ldots, m+1\}$, it follows from inequality (10) and the Hoeffding inequality [32] that

$$\begin{aligned} &P\left(\left|\nabla J_j(\omega; S) - \nabla J_j(\omega)\right| \ge d_j\lambda\right) \\ &\le \exp\left(-2\frac{\left(d_j\lambda\right)^2}{\left(\max_{1\le i\le N} \nabla l_j(\omega; X_i) - \min_{1\le i\le N} \nabla l_j(\omega; X_i)\right)^2} n\right) \\ &\le \exp\left(-2n\lambda^2\right). \end{aligned} \tag{11}$$

For the convenience of the following developments, we make the following remarks. Let t_j be the number of iterations until the AS algorithm satisfies the inequality $|\nabla\widehat{J}_j(\omega; \widetilde{S}_{n_t})| \ge (1 + 1/\varepsilon)\alpha_j^t$, where $\widetilde{S}_{n_t} \subseteq T$ is a subset obtained through random and independent sampling from T with the size of $n_t = t \times s$, $j = 1, 2, \ldots, m+1$. Note that t_j

is a random variable depending on the sample drawn from T. Let t_j^1 be the smallest integer meeting the following inequalities:

$$\alpha_j^{t_j^1} < \varepsilon \frac{\left|\nabla J_j(\omega)\right|}{(1+2\varepsilon)} \text{ and } \alpha_j^{t_j^1-1} \geq \varepsilon \frac{\left|\nabla J_j(\omega)\right|}{(1+2\varepsilon)}. \tag{12}$$

Since α_j^t is a strictly decreasing function with t and $J_j(\omega)$ is fixed under any given ω, hence, t_j^1 is uniquely determined for $j = 1, 2, \ldots, m+1$. □

Lemma 3. *If $t_j < t_j^1$, then we have $\nabla J_j(\omega; \widetilde{S}_{t_j}) \nabla J_j(\omega) \geq (1-\varepsilon)(J_j(\omega))^2$ with probability $>1-\delta/(2+2m)$, where $j = 1, 2, \ldots, m+1$, $0 < \delta < 1/2$.*

Proof. We get from the AS algorithm the estimate $\nabla J_j(\omega; \widetilde{S}_{t_j})$ satisfying $|\nabla J_j(\omega; \widetilde{S}_{t_j})| \geq (1+1/\varepsilon)\alpha_j^{t_j}$ at the t_j-th step. When $t_j < t_j^1$, we have $\alpha_j^{t_j^1} \geq \alpha_j^{t_j^1-1} \geq \varepsilon|\nabla J_j(\omega)|/(1+2\varepsilon)$ and

$$\begin{aligned}
&\left|\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\right| \\
&\geq \left(1+\frac{1}{\varepsilon}\right)\alpha_j^{t_j} \geq \left(1+\frac{1}{\varepsilon}\right)\alpha_j^{t_j^1-1} \\
&\geq \frac{(1+1/\varepsilon)\varepsilon\left|\nabla J_j(\omega)\right|}{(1+2\varepsilon)} \\
&= \left(\frac{1-\varepsilon}{(1+2\varepsilon)}\right)\left|\nabla J_j(\omega)\right| \\
&\geq (1-\varepsilon)\left|\nabla J_j(\omega)\right|.
\end{aligned} \tag{13}$$

Thus, it always holds that $\nabla J_j(\omega; \widetilde{S}_{t_j}) \nabla J_j(\omega) \geq (1-\varepsilon)(\nabla J_j(\omega))^2$ as long as $\nabla J_j(\omega; \widetilde{S}_{t_j})$ and $\nabla J_j(\omega)$ have the same sign. On the other hand, if $\nabla J_j(\omega; \widetilde{S}_{t_j})$ and $\nabla J_j(\omega)$ have different signs, then they are quite different as $|\nabla J_j(\omega; \widetilde{S}_{t_j})| \geq \alpha_j^{t_j}(1+1/\varepsilon)$. Next, we show the difference is large enough, and it is conducive to prove the probability that this situation occurs is small.

In the following, we will give this difference from two cases. If $\nabla J_j(\omega; \widetilde{S}_{t_j}) \leq 0$ and $\nabla J_j(\omega) \geq 0$, then $|\nabla J_j(\omega; \widetilde{S}_{t_j}) - \nabla J_j(\omega)| = -\nabla J_j(\omega; \widetilde{S}_{t_j}) + \nabla J_j(\omega) \geq -\nabla J_j(\omega; \widetilde{S}_{t_j}) = |\nabla J_j(\omega; \widetilde{S}_{t_j})| \geq \alpha_{t_j}(1+1/\varepsilon) \geq \alpha_{t_j}$. If $\nabla J_j(\omega; \widetilde{S}_{t_j}) \geq 0$ and $\nabla J_j(\omega) \leq 0$, then $|\nabla J_j(\omega; \widetilde{S}_{t_j}) - \nabla J_j(\omega)| = \nabla J_j(\omega; \widetilde{S}_{t_j}) - \nabla J_j(\omega) \geq \nabla J_j(\omega; \widetilde{S}_{t_j}) = |\nabla J_j(\omega; \widetilde{S}_{t_j})| \geq \alpha_j^{t_j}(1+1/\varepsilon) \geq \alpha_j^{t_j}$. Therefore, the difference is $|\nabla J_j(\omega; \widetilde{S}_{t_j}) - \nabla J_j(\omega)| \geq \alpha_j^{t_j}$ as $\nabla J_j(\omega; \widetilde{S}_{t_j}) \nabla J_j(\omega) \leq 0$. Then, the probability that $\nabla J_j(\omega; \widetilde{S}_{t_j})$ and $\nabla J_j(\omega)$ have different labels is estimated using Lemma 2 as follows:

$$\begin{aligned}
&P\left(\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\nabla J_j(\omega) \leq 0\right) \\
&\leq P\left(\left|\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right) - \nabla J_j(\omega)\right| \geq \alpha_j^{t_j}\right) \\
&= P\left(\left|\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right) - \nabla J_j(\omega)\right| \geq d_j\sqrt{\frac{\ln((6m+6)/\delta)}{\left(2\left(t_j \cdot s\right)^3\right)}}\right) \\
&\leq \exp\left(\frac{-2\left(t_j \cdot s\right)\ln((6m+6)/\delta)}{\left(2\left(t_j \cdot s\right)^3\right)}\right) \\
&= \frac{\delta}{(6m+6)}\exp\left(\frac{1}{\left(t_j \cdot s\right)^3}\right)\Bigg) \\
&\leq \frac{\delta}{(6m+6)}\exp(1) \\
&\leq \frac{\delta}{(2m+2)}.
\end{aligned} \tag{14}$$

□

Lemma 4. *$P(t_j < t_j^1) \geq 1-\delta/(2+2m)$ for $j = 1, 2, \ldots, m+1$.*

Proof. When $t_j < t_j^1$, it always holds that $|\nabla J_j(\omega; \widetilde{S}_{t_j})| < (1+1/\varepsilon)\alpha_j^{t_j^1}$ and $\alpha_j^{t_j^1} < \varepsilon(1+2/\varepsilon)\nabla|J_j(\omega)|$. Then,

$$\begin{aligned}
&P\left(t_j < t_j^1\right) \\
&\leq P\left(\left|\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\right| < \left(1+\frac{1}{\varepsilon}\right)\alpha_j^{t_j^1}\right) \\
&= P\left(\left|\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\right| < \frac{(1+2\varepsilon)}{\varepsilon\alpha_j^{t_j^1} - \alpha_j^{t_j^1}}\right) \\
&\leq P\left(\left|\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\right| < |\nabla|J_j(\omega)| - \alpha_j^{t_j^1}\right) \\
&= P\left(|\nabla|J_j(\omega)| - \left|\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\right| > \alpha_j^{t_j^1}\right).
\end{aligned} \tag{15}$$

It follows from the triangle inequality that $|\nabla J_j(\omega) - \nabla J_j(\omega; \widetilde{S}_{t_j})| \geq |\nabla J_j(\omega)| - |\nabla J_j(\omega; \widetilde{S}_{t_j})| > \alpha_j^{t_j^1}$. Thus, $P(|\nabla J_j(\omega) - \nabla J_j(\omega; \widetilde{S}_{t_j})| \geq \alpha_j^{t_j^1}) \geq P(|J_j(\omega)| - |\nabla J_j(\omega; \widetilde{S}_{t_j})| > \alpha_j^{t_j^1})$. Combining inequality (15) with Lemma 2, we have

$$\begin{aligned} P\left(t_j < t_j^1\right) &= 1 - P\left(t_j \geq t_j^1\right) \\ &\geq 1 - P\left(\left|\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\right| - \left|\nabla J_j(\omega)\right| > \alpha_j^{t_j^1}\right) \\ &\geq 1 - P\left(\left|\nabla J_j(\omega) - \nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\right| > \alpha_j^{t_j^1}\right) \\ &\geq 1 - \frac{\delta}{(2+2m)}. \end{aligned} \tag{16}$$

Combining with Lemma 3 and 4, we can easily get the following theorem. □

Theorem 1. *For any* $1/2 > \varepsilon > 0$ *and* $1/2 > \delta > 0$, *the estimation* $\nabla J_j(\omega; \widetilde{S}_{t_j})$ *generated by AS algorithm satisfies the following inequality:*

$$P\left(\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\nabla J_j(\omega) \geq (1-\varepsilon)\left(J_j(\omega)^2\right)\right) \geq \frac{1-\delta}{(1+m)}, \tag{17}$$

where $j = 1, 2, \ldots, m+1$.

Theorem 2. *For any* $1/2 > \varepsilon > 0$ *and* $1/2 > \delta > 0$, *final estimation* $\nabla\widehat{J}(\omega) = (\nabla J_1(\omega; \widetilde{S}_{t_1}), \ldots, \nabla J_{m+1}(\omega; \widetilde{S}_{t_{m+1}}))$ *generated by the AS algorithm satisfies inequality* $P\left((1-\varepsilon)\left(\|\nabla J(\omega)\|_2\right)^2 \leq \nabla\widehat{J}(\omega)^\top \nabla J(\omega) \geq 1-\delta\right)$.

Proof. Combining Theorem 1 and Lemma 1, then we have

$$\begin{aligned} &P\left((1-\varepsilon)\left(\|\nabla J(\omega)\|_2\right)^2 \leq \nabla\hat{J}(\omega)^\top \nabla J(\omega)\right) \\ &= P\left(\sum_{j=1}^{m+1} \nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\nabla J_j(\omega) \geq (1-\varepsilon)\sum_{j=1}^{m+1}\left(\nabla J_j(\omega)\right)^2\right) \\ &\geq P\left(\bigcap_{j=1}^{m+1}\left(\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\nabla J_j(\omega) \geq (1-\varepsilon)\left(\nabla J_j(\omega)\right)^2\right)\right) \\ &\geq \sum_{j=1}^{m+1} P\left(\nabla J_j\left(\omega; \widetilde{S}_{t_j}\right)\nabla J_j(\omega) \geq (1-\varepsilon)\left(\nabla J_j(\omega)\right)^2\right)\Bigg) - m \\ &\geq \sum_{j=1}^{m+1}\left(\frac{1-\delta}{(m+1)}\right) - m = 1-\delta. \end{aligned} \tag{18}$$

In Theorem 2, the obtained estimation using the AS algorithm could keep the decreasing value of the current objective function in each iteration calculation of GD. Therefore, it could guarantee that LLR-AS algorithm gets the optimal solution of the convex optimization problem (2), and this conclusion is verified in the experiment. Besides the optimal solution, we also pay attention to the number of sampled examples (NSE). AS algorithm is an iterative sampling algorithm that samples a fixed number of samples each time, then we can estimate NSE using the total number of iterations. We have already proved that AS algorithm terminates finally within t^1 steps from Lemmas 3 and 4, where $t^1 = \max_{1\leq j\leq m+1} t_j^1$. It is well known that t^1 could be the minimum number of iterations satisfying condition (12); then, we could assume that $\alpha_1^j \approx \varepsilon|\nabla J_j(\omega)|/(1+2\varepsilon)$, where $j = 1, 2\cdots, m+1$. Finally, we can estimate NSE as follows:

$$\mathrm{NSE} = s \cdot t^1 \leq s \cdot \max_{1\leq j\leq m+1} \sqrt{[3]}\left\{\frac{(1+2\varepsilon)d_j}{\sqrt{2}\,\varepsilon\left|\nabla J_j(\omega)\right|}\right\}^2 \ln\frac{6m+6}{\delta} + 1. \tag{19}$$

Next, we will discuss the above formula. The effect of the parameter m and δ on formula (19) is small because they are in the logarithmic function. Namely, the AS algorithm has a low possibility to sample too many examples. So, it is useful for sampling examples from the large-scale data. □

4. Experiments

4.1. Experimental Setup. Two representative gradient descend methods were selected in this study: a common gradient descend with all the data and a dynamic sample gradient algorithm [25]. These two algorithms are used to solve the logistic regression, and they are named LR-GD and LR-DSG. Seven benchmark datasets are selected for making a fair comparison between our proposal and others [33, 34]; their information is shown in Table 1. All of these selected datasets have larger than 50000 instances.

Owning to the simplicity and successful application, we select the classification accuracy (Acc) and training time as the performance measure. LLR-AS and LR-DSG are both accelerated algorithms of LR-GD, so we compare their relative speeds (RS) and the difference in their solutions. RS is a ratio of training time between LLR-AS and each of the others. For estimating these three performance measures Acc and RS, we used a 10-fold cross-validation method. To compare the difference between our method and others under a performance measure, we adopt the Wilcoxon signed-rank test (WSRT) [35]. The reason for selecting WSRT is that it does not require a strict data distribution hypothesis and has stable performance. It is empirically considered to be stronger than other tests [36]. The null hypothesis of WSRT denotes that there exists no significant difference between our algorithm and each one of the others under a performance measure, while the alternative is that there exists a significant difference.

In the following experiments, we fix an initial sample S_0 of size 1% of the total training set and $\theta = 0.5$ for DSG algorithm according to [25], and $\varepsilon = 0.5$, $\delta = 0.1$ for LLR-AS. The stopping condition for these three iterative algorithms is that the Euclidean distance between the current and the previous solution is smaller than 0.001 and the maximum iteration 5000. The significance level of 0.05 is used. All the experiments are executed in Python 3.8 on the same computer of Intel Xeon E5-2650 CPU and 32 GB of RAM.

4.2. Experimental Results and Analysis. In this section, we adopt the classification ability and training efficiency as two important measurements to evaluate the performance of

Table 1: Summary of datasets.

Dataset	Size	Features	Class
Cifa	60000	3072	2
Cod-rna	59535	8	2
Covtype	581012	54	2
Ijcnn1	141691	22	2
Mnist	350000	85	2
Skin-nonskin	245057	3	2
Susy	5000000	18	2

these algorithms and give a detailed comparing analysis and the reason for the experiment result.

4.2.1. Classification Performance Analysis. Under the theoretical hypothesis of logistic regression, its classification performance of logistic regression depends on the solution obtained by gradient descend. LR-GD algorithm is trained by gradient descend with all the training data; then its weight vector is the optimal solution, as well as its classification performance. In the following, we compare the predictive ability of these algorithms from the difference in the obtained solution vector and classification accuracy.

(1) The analysis of the difference on solution vector: Pearson correlation coefficient $\rho \in [0, 1]$ is chosen to evaluate the difference between two solution vectors for its high effectiveness. Its value is inversely proportional to the difference. The larger the value of ρ, the smaller difference between the two vectors. The correlation coefficients between the solution vector obtained by LLP-AS and each one of LR-GD and LR-DSG algorithms on the test data of each dataset are computed, all the statistics results on different datasets are listed in Table 2.

Table 2 shows that the correlation coefficient ρ of the weight vector between LLR-AS and LR-GD is nearly closed to 1 on almost all datasets except for the Cifa dataset, and this same result can be obtained between LLR-AS and LR-DSG. The mean of correlation coefficients on all the datasets is 0.986 and 0.989, and their medians are 0.996 and 0.986. Moreover, it can get a detailed comparison from the descriptive statistics. The following can be seen: (1) there exists a negligible difference of solution vector between the LLR-AS algorithm and each one of these two algorithms, and then the LLR-AS algorithm can get nearly the same solution vector as the LR-GD algorithm. (2) The standard deviation of the correlation coefficients on each data is tiny; then, this experiment result validates that the proposed algorithm could get a robust solution.

Own to the properties of the obtained vector estimation at each iteration, the LLR-AS algorithm performs multiple iterations to continually minimize the objective function value. Meanwhile, the original optimization problem has a unique optimal solution because it is a convex optimization. So, the LLR-AS algorithm is able to guarantee convergence and obtain the optimal solution as the LR-GD algorithm. Furthermore, Theorem 2 has also been verified by this experiment result, and the gradient estimation is stable for different datasets.

Table 2: The correlation coefficient ρ between different solutions on seven datasets.

Dataset	ρ (LLR-AS, LR-GD)		ρ (LLR-AS, LR-DSG)	
	Std	Mean	Std	Mean
Cifa	0.006	0.935	0.003	0.986
Cod-rna	0.003	0.998	0.003	0.991
Covtype	0.007	0.979	0.048	0.984
Ijcnn1	0.001	0.996	0.004	1.000
Mnist	0.000	0.995	0.003	0.985
Skin-nonskin	0.000	1.000	0.004	0.979
Susy	0.001	0.998	0.009	0.998
Average	0.986		0.989	
Median	0.996		0.986	

(2) The analysis of the difference in classification. the classification accuracy Acc is adopted to compare the performance between the proposed algorithm and two state-of-the-art approaches, and WSRT is performed to test whether there exists a significant difference among them. Table 3 lists the descriptive statistics of classification accuracy of each algorithm obtained by 10-fold cross-validation, and their results are also plotted in Figure 1.

The result on each dataset plotted in Figure 1 shows that the LLR-AS achieves better classification accuracy than the LR-GD algorithm and LR-DSG algorithm on the Ijcnn1 dataset, and it has not the worst classification accuracy on the rest of the datasets. To assess the overall classification performance on all the datasets, the mean and median of the result of each algorithm on eight datasets are computed in the last two rows of Table 3. Their mean values of classification accuracy are 0.750, 0.731, and 0.747, and the median values are 0.722, 0.711, and 0.713. Therefore, there exists a negligible difference in classification accuracy among these algorithms. Finally, the obtained p values using WSRT between LLR-AS and each one of the other algorithms are 0.1563 and 0.688, both larger than the given significant level of 0.05. Then, it gets that (1) the LLR-AS algorithm has no significant difference in classification accuracy with the LR-GD algorithm and LR-DSG algorithm on the selected datasets. (2) The LLR-AS algorithm has a stable classification performance because its standard deviation of the classification accuracy of the LLR-AS algorithm on every data is relatively small.

The reason for the similar classification result is that the classification performance of logistic regression depends on the solution vector, and the LLR-AS algorithm has no significant different solution vector from the LR-GD algorithm and the LR-DSG algorithm. Moreover, the obtained solution vector of the algorithm is robust according to Theorem 2, and the small standard deviation of the correlation coefficient on different datasets also verifies this fact.

4.2.2. Training Efficiency Analysis. Besides the classification performance, training speed is another important measurement to evaluate algorithm training performance. The relative speed RS can evaluate the accelerating extent of the LR-DSG algorithm. Table 4 lists RS on different datasets.

Table 3: Acc of three algorithms on seven datasets.

Dataset	LR-GD		LR-DSG		LLR-AS	
	Std	Mean	Std	Mean	Std	Mean
Cifa	0.003	0.904	0.003	0.901	0.003	0.904
Cod-rna	0.142	0.782	0.005	0.888	0.060	0.851
Covtype	0.103	0.665	0.117	0.682	0.145	0.655
Ijcnn1	0.004	0.794	0.004	0.794	0.011	0.841
Mnist	0.002	0.711	0.003	0.713	0.005	0.722
Skin-nonskin	0.002	0.583	0.002	0.573	0.005	0.589
Susy	0.002	0.680	0.002	0.681	0.005	0.689
Average	0.731		0.747		0.750	
Median	0.711		0.711		0.722	

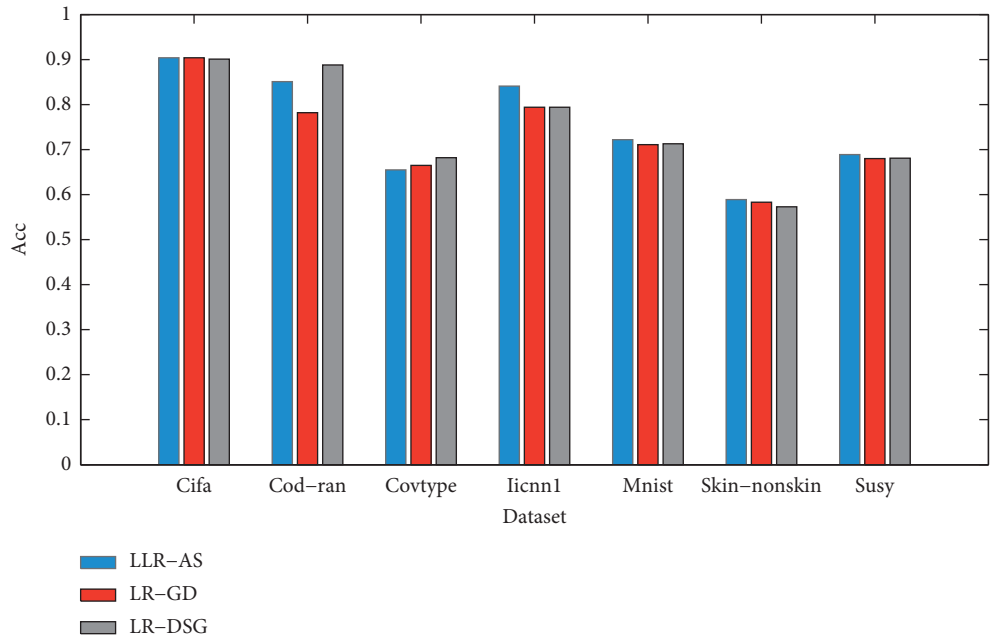

Figure 1: Acc of different algorithms on different datasets.

Table 4: The relative speed on seven datasets.

Dataset	LLR-AS vs LR-GD		LLR-AS vs LR-DSG	
	Std	Mean	Std	Mean
Cifa	3.049	109.339	0.003	1.461
Cod-rna	3.456	111.260	0.203	3.067
Covtype	9.659	270.898	0.348	3.945
Ijcnn1	4.641	64.132	0.004	1.070
Mnist	5.014	83.377	0.003	1.082
Skin-nonskin	1.036	20.282	0.004	1.243
Susy	4.887	91.942	0.009	1.031
Average	107.319		1.843	

It finds that the value of RS between the LLR-AS algorithm and LR-GD algorithm is larger than 20 on all the datasets from Table 4, and its value is larger than 109 on Cifa, Cod-rna, and Covtype dataset. So, the LLR-AS algorithm can largely reduce the training time of the LR-GD algorithm. On the other hand, the value of RS between the LLR-AS algorithm and LR-DSG algorithm is larger than one on these seven datasets, and its average value of RS on all the datasets is 1.843. Therefore, the LLR-AS algorithm indeed needs less training time than the LR-DSG algorithm.

There exist three reasons for explaining that the proposed mechanism can achieve a good result on training efficiency. (1) The obtained gradient estimation could be a descent direction of the current objective function at each iteration. Thus, the total number of iterations that is positively correlated with the training efficiency will reduce. (2) The divide-and-conquer approach is adopted to compute each component of the gradient vector at each iteration, and it can be executed in a parallel environment. (3) It is proved that there exists a low possibility to sample too many examples to estimate the gradient; the time of estimating gradient could become short at each iteration. Therefore, the proposed algorithm has a better performance than other representative algorithms owning to the above three merits.

5. Summary

A novel algorithm for fast training logistic regression via adaptive sampling has been proposed to effectively handle the massive dataset in this paper. The proposed algorithm solves the difficulty that the sample size needs to be fixed before sampling, and it also offers an idea of dividing the multivariate estimation problem into several easy-to-solve subproblems. Experimental results on real datasets demonstrate that LLR-AS has obtained a similar classification performance with less execution time in comparison with other representative algorithms. Moreover, this proposed algorithm can deal with the multiclassification problem using the one-vs-all scheme, and paper [37] has shown that this scheme is as accurate as any other approach. The proposed algorithm solves the binary classification problem, but it can be used for the multiclassification problem using the one-vs-all scheme. It needs to train several different classifiers, where each classifier is obtained by distinguishing the instances of the same class from the others in the rest classes. When given an unlabeled instance, the final output is the largest result among the results of all the classifiers.

Though the proposed algorithm has a good performance for large-scale data, there exist two limitations for dealing with various kinds of real datasets. The gradient estimation needs all the features of the data at each iteration, so that it may take a great challenge of its training efficiency for high dimensional data [38–40]. Furthermore, this algorithm does not consider the label distribution of the data, and then its performance on imbalanced data could decrease. In the future, we will study how to combine sampling and feature selection to scale up machine learning algorithms and design an effective mechanism to deal with the class imbalance problem.

Data Availability

This publication was supported by LIBSVM datasets, which are openly available at location cited in [33].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Shandong Provincial Natural Science Foundation, China (no. ZR2020MF146), Major Scientific and Technological Innovation Project of Shandong Province (no. 2019JZZY010716), and Key R&D Plan of Shandong Province (no. 2019GGX101061).

References

[1] M. Kubat, *An Introduction to Machine Learning*, Springer, Berlin, Germany, 2017.

[2] M. Scott, *Applied Logistic Regression Analysis*, Sage, Thousands Oaks, CA, USA, 2002.

[3] T. Rymarczyk, E. Kozłowski, G. Kłosowski, and K. Niderla, "Logistic regression for machine learning in process tomography," *Sensors*, vol. 19, no. 15, p. 3400, 2019.

[4] A. De Caigny, K. Coussement, and K. W. De Bock, "A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees," *European Journal of Operational Research*, vol. 269, no. 2, pp. 760–772, 2018.

[5] W. Chen, X. Zhao, H. Shahabi et al., "Spatial prediction of landslide susceptibility by combining evidential belief function, logistic regression and logistic model tree," *Geocarto International*, vol. 34, no. 11, pp. 1177–1201, 2019.

[6] S. Nusinovici, Y. C. Tham, M. Y Chak Yan et al., "Logistic regression was as good as machine learning for predicting major chronic diseases," *Journal of Clinical Epidemiology*, vol. 122, no. 56–69, pp. 56–69, 2020.

[7] E. Christodoulou, J. Ma, G. S. Collins et al., "A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models," *Journal of Clinical Epidemiology*, vol. 110, no. 12–22, 2019.

[8] W. Chen, Z. Sun, and J. Han, "Landslide susceptibility modeling using integrated ensemble weights of evidence with logistic regression and random forest models," *Applied Sciences*, vol. 9, no. 1, p. 171, 2019.

[9] C. Bonte and F. Vercauteren, "Privacy-preserving logistic regression training," *BMC Medical Genomics*, vol. 11, no. 4, pp. 13–21, 2018.

[10] P. C. Austin and E. W. Steyerberg, "The integrated calibration index (ici) and related metrics for quantifying the calibration of logistic regression models," *Statistics in Medicine*, vol. 38, no. 21, pp. 4051–4065, 2019.

[11] H. Li, *Statistical Learning Method*, Tsinghua University Press, Beijing, China, 2012.

[12] H.Y. Wang, R. Zhu, and P. Ma, "Optimal subsampling for large sample logistic regression," *Journal of the American Statistical Association*, vol. 113, no. 522, pp. 829–844, 2018.

[13] A. Mustapha, L. Mohamed, and K. Ali, "An overview of gradient descent algorithm optimization in machine learning: application in the ophthalmology field," in *Proceedings of the International Conference on Smart Applications and Data Analysis*, Marrakesh, Morocco, June 2020.

[14] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical Programming*, vol. 120, no. 1, pp. 221–259, 2009.

[15] A. Agarwal and J. Duchi, "Distributed delayed stochastic optimization," in *Proceedings of the Neural Information Processing Systems (NIPS)*, Granada, Spain, December 2011.

[16] G. Andrew and J. Gao, "Scalable training of L1-regularized log-linear models," in *Proceedings of the International Conference on Machine Learning (ICML)*, Corvallis, OR, USA, July 2007.

[17] S. V. N. Vishwanathan, N. Schraudolph, M. Schmidt, and K. Murphy, "Accelerated training of conditional random fields with stochastic gradient methods," in *Proceedings of the International Conference on Machine Learning (ICML)*, Pittsburgh, PA, USA, July 2006.

[18] S. Wright, "Accelerated block-coordinate relaxation for regularized optimization," *SIAM Journal on Optimization*, vol. 22, no. 1, pp. 159–186, 2012.

[19] H. Liu and H. Motoda, "On issues of instance selection," *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 115–130, 2002.

[20] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.
[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.
[22] R. Ranganath, C. Wang, B. David, and E. Xing, "An adaptive learning rate for stochastic variational inference," in *Proceedings of the International Conference on Machine Learning Machine Learning (ICML)*, Atlanta, GA, USA, January 2013.
[23] D. Kingma and B. Jimmy, "Adam: a method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, PR, USA, May 2014.
[24] Y.-A. Ma, T. Chen, and E. Fox, "A complete recipe for stochastic gradient mcmc," in *Proceedings of the Neural Information Processing Systems (NIPS)*, Montréal, Canada, December 2015.
[25] R. Byrd, G. Chin, J. Nocedal, and Y. Wu, "Sample size selection in optimization methods for machine learning," *Mathematical Programming*, vol. 134, no. 1, pp. 127–155, 2012.
[26] H. Robbins and M. Sutton, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.
[27] J. Liang, Y. Song, D. Li, Z. Wang, and C. Dang, "An accelerator for the logistic regression algorithm based on sampling on-demand," *Science China Information Sciences*, vol. 63, no. 6, pp. 226–228, 2020.
[28] S. Gopal, "Adaptive sampling for sgd by exploiting side information," in *Proceedings of the International Conference on Machine Learning (ICML)*, New York, NY, USA, July 2016.
[29] H. A. Taha, *Operations Research*, Pearson New International Edition, London, UK, 2013.
[30] L. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
[31] F. Provost, D. Jensen, and Tim Oates, "Efficient progressive sampling," in *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, San Diego, CA, USA, August 1999.
[32] Z. Lin and Z. Bai, *Probability Inequalities*, Springer, Berlin, Germany, 2011.
[33] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.
[34] C.-J. Hsieh, S. Si, and I. Dhillon, "A divide-and-conquer solver for kernel support vector machines," in *Proceedings of the International Conference on Machine Learning (ICML)*, Beijing, China, July 2014.
[35] W. Frank, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
[36] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
[37] R. Ryan and A. Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, pp. 101–141, 2004.
[38] X. Song, Y. Zhang, Y. Guo, X. Sun, and Y. Wang, "Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 882–895, 2020.
[39] Y. Hu, Y. Zhang, and D. Gong, "Multiobjective particle swarm optimization for feature selection with fuzzy cost," *IEEE Transactions on Cybernetics*, vol. 51, no. 2, pp. 874–888, 2021.
[40] X.-F. Song, Y. Zhang, D.-W. Gong, and X.-Z. Gao, "A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data," *IEEE Transactions on Cybernetics*, pp. 1–14, 2021.