A Hybrid Optimization Method For Acceleration Of Building Linear Classification Models

Junchao Lv, Qiang Wang and Joshua Zhexue Huang

Abstract—Linear classification is an important technique in machine learning and data mining, and development of fast optimization methods for training linear classification models is a hot research topic. Stochastic gradient descent (SGD) can achieve relatively good results quickly, but unstable to converge. Limited-memory BFGS (L-BFGS) method converges, but takes a long time to train the model, as it needs to compute the gradient from the entire data set to make an update. In this paper, we investigate a hybrid method that integrates SGD and L-BFGS into a new optimization process SGD-LBFGS to take advantages of both optimization methods. In SGD-LBFGS, SGD is used to run initial iterations to obtain a suboptimal result, and then L-BFGS takes over to continue the optimization process until the process converges and a better model is built. We present a theoretical result to prove that SGD-LBFGS converges faster than SGD and L-BFGS. Experiment analysis on 6 real world data sets have shown that SGD-LBFGS converged 77% faster than L-BFGS on average and demonstrated more stable results than SGD.

I. INTRODUCTION

Linear classification is a useful tool in machine learning and data mining, and plays an important role in applications such as text classification [1] and credit risk analysis [2]. To deal with large data, development of fast optimization methods for training linear classification models has become a hot research topic.

Given a set of \( l \) instance-label pairs \((x_i, y_i)\) where \( 1 \leq i \leq l \), \( x_i \in \mathbb{R}^n \) and \( y_i \in \{-1, +1\} \), building a linear classification model from the data set requires to solve the following unconstrained optimization problem:

\[
\min_w \ f(w) = \frac{\lambda}{2} r(w) + \frac{1}{l} \sum_{i=1}^{l} \xi(w; x_i, y_i) \quad (1)
\]

where \( r(w) \) is a regularization term, \( \xi(w; x_i, y_i) \) is a loss function, and \( \lambda > 0 \) is a user-specified parameter to balance \( r(w) \) and the sum of losses. Here, we omit the bias term for simplicity. When referring to support vector machine (SVM) and logistic regression (LR) models, the regularization term is \( w^2 \), the loss function is written as \( \min_x \{ \max(0, 1 - yw^T x) \} \) for SVM, and as \( \log(1 + e^{-yw^T x}) \) for LR.

The optimization methods for solving (1) can be divided into two classes: batch learning and online learning. Traditional optimization methods mainly focus on batch learning. Batch learning methods compute gradient or hessian matrix from all instances in the training data at each update step. Popular methods include nonlinear conjugate gradient descent [3], quasi Newton method (particularly, limited-memory BFGS (L-BFGS) [4]), and trust region Newton methods [5]. These optimization methods are very stable in training linear classification models and converge to the minimum. Their common shortcoming is time consuming.

Online learning methods randomly pick one instance to make an update at each step. The well known stochastic gradient descent (SGD) [6],[7],[8],[9] picks one instance at a time to directly compute the gradient. Averaged stochastic gradient descent (ASGD) [10],[11] performs stochastic gradient descent iterations and simultaneously computes the average of parameter vectors over time. SGD-QN [12],[13] utilizes the second order information carefully to compute the update. These online methods are fast to build classification models, but their optimization processes are unstable.

In this paper, we investigate a hybrid optimization method that integrates SGD and L-BFGS into a new optimization process SGD-LBFGS to take advantages of both optimization methods. In SGD-LBFGS, SGD is used to run initial iterations to obtain a suboptimal result, and then L-BFGS takes over to continue the optimization process until the process converges and a better model is built. We present a theoretical result to prove that SGD-LBFGS converges faster than SGD and L-BFGS. We conducted numerical analysis of SGD-LBFGS with 6 real world data sets and compared SGD-LBFGS with SGD and L-BFGS. The results have shown that SGD-LBFGS converged 77% faster than L-BFGS. The experiments also indicated that the optimization process of SGD-LBFGS was much smoother than SGD and the results by SGD-LBFGS were more stable than those by SGD. An important contribution of this paper is that we demonstrate integrating online learning method and batch learning method could achieve fairly good results on both theory analysis and empirical study.

The two-stage based ensemble optimization evolutionary algorithm recently proposed by [14] also uses different methods to improve the optimization process. Firstly, the algorithm uses a fast search technique to find a small region which is promising to contain the optimal points. Then, it uses a cooperative co-evolution based search technique to further explore the small region extensively to obtain the optimal solution. In the terascale linear learning system by [15], a hybrid online+batch method is introduced to accelerate the convergence of the learning system. The learning process starts with each node to make one online pass over its local
data according to adaptive gradient updates, then computes the weighted average over all nodes by AllReduce, and finally the weighted average initiates L-BFGS optimization process. Unlike this system, we proposed to integrate SGD and L-BFGS directly, theoretically analyzed the integrated process in this paper, and empirically demonstrated the advantages of SGD-LBFGS over SGD and L-BFGS.

II. PRELIMINARIES

For online learning, we can rewrite the objective function (1) as

$$
\min_w f(w) = \frac{1}{l} \sum_{i=1}^{l} \frac{\lambda}{2} r(w) + \xi(w; x_i, y_i)
$$

where each term in the sum refers to the objective function $f_i(w)$ computed from one instance $i$

$$
f_i(w) = \frac{\lambda}{2} r(w) + \xi(w; x_i, y_i)
$$

Respecting online learning, parameter $w$ is updated at each instance $(x_i, y_i)$, whereas batch learning updates parameter $w$ as the average of online learning updates from all instances.

In the following, we briefly review the batch learning method L-BFGS and the online learning method SGD.

A. Limited-memory BFGS

BFGS is a popular quasi-Newton method for batch learning. Instead of computing inverse Hessian matrix $H_l$ directly, BFGS first generates an estimate of $H_l$, and then updates the inverse Hessian matrix, the limited-memory BFGS by [4] computes the low-rank estimates of $H_l \nabla f(w_t)$ from the previous $m$ updates. This method does not need to store the inverse Hessian matrix, thus saving a lot of memory. The limited-memory BFGS algorithm is summarized in Algorithm 1.

Algorithm 1 Limited-memory BFGS

Given: $w_0$, $H_0$, and a small integer $m$.

for $t = 0, 1, \ldots$ do

if $\nabla f(w_t) = 0$ then break.

$d_0 = -\nabla f(w_0)$

for $t = 0, 1, \ldots$ do

if $\nabla f(w_t) = 0$ then break.

$d_{t+1} = -H_{t+1} \nabla f(w_{t+1})$, where $H_{t+1} \nabla f(w_{t+1})$ is evaluated by the $m$ vectors from previous updates.

end for

$w_{t+1} = w_t + \eta_t d_t$, where $\eta_t > 0$ is achieved by line search method.

end if

end for

The convergence property of L-BFGS is specified in Assumption 6.1 of [4]. The convergence conditions include that $f(w)$ must be convex and twice continuously differentiable, the level set $\{ w : f(w) \leq f(w_0) \}$ is convex, etc. If the objective function meets the above prerequisites, the following inequalities hold:

$$
f(w_t) - f_\star \leq r^l (f(w_0) - f_\star), r \in (0, 1)
$$

$$
\| w_t - w_\star \|^2 \leq Ar^l (f_0 - f_\star), r \in (0, 1)
$$

where $A > 0$ is a constant value. The two inequalities indicate that the convergence rate of L-BFGS is $O(r^l)(0 < r < 1)$.

B. Stochastic gradient descent

Stochastic gradient descent (SGD) randomly picks an instance-label pair $(x_t, y_t)$ in each iteration to compute the update of parameter $w$ as (3). The computation process is illustrated in Algorithm 2.

Algorithm 2 Stochastic gradient descent

Given: $w_0$.

for $t = 0, 1, \ldots$ do

$w_{t+1} = w_t + \eta_t \nabla f(w_t)$, where $f_t(w_t)$ is the same as (3), and for experiment, we set $\eta_t = \frac{1}{m\lambda}$ (where $\lambda$ is the regularization term in (2)).

end for

The convergence property of SGD is specified in Theorem 1 of [16]. Briefly speaking, the objective function $f(w)$ must be smooth, and the learning rate $\eta_t$ is set being proportional to $1/t$. Theorem 1 also gives the following inequality:

$$
E[f(w_t) - f(w_\star)] \leq \frac{B}{t}
$$

where $B > 0$ is a constant value. From this inequality, the convergence rate is $O(1/t)$.

III. INTEGRATION OF SGD AND L-BFGS

In this section, we present SGD-LBFGS, a hybrid method that integrates SGD and L-BFGS into one optimization process to take advantages of both methods.

Let SGD($n_1$) and L-BFGS($n_2$) denote that both algorithms run $n_1$ iterations. Fig. 1 illustrates the convergence rate trends of the two algorithms. From the figure, we can see that when $t < T_\star$, SGD converges faster than L-BFGS and when $t > T_\star$, L-BFGS converges faster than SGD.

Using this property, we can integrate these two optimization methods to accelerate the convergence of the optimization process as follows. We use SGD to run a few iterations to produce an initial optimal result and use the initial result from SGD as start point of L-BFGS to continue the optimization process until the process converges.

The hybrid method SGD-LBFGS($n_1$, $n_2$) is composed of two steps:

1) SGD runs $n_1$ iterations to produce the initial weight $w_{n_1}$.

2) Using the weight $w_{n_1}$ as the start point, L-BFGS runs $n_2$ iterations to accelerate the convergence and produce the final weight $w_{n_1+n_2}$. 

2218
The convergence of SGD-LBFGS can be discussed as follows. As the objective function is required to be convex, if the descent direction is toward the negative gradient direction, the objective function will converge. Generally, SGD moves toward the negative gradient direction in the initial runs before L-BFGS starts, and the descent direction of L-BFGS is toward the negative gradient direction, so SGD-LBFGS will converge to the minimum.

Next, we analyze the convergence rate of SGD-LBFGS and compare it with those of SGD and L-BFGS.

**Theorem 1:** Suppose the objective function $f(w)$ satisfies Assumption 6.1 of [4] and the prerequisites of Theorem 1 in [16], SGD-LBFGS($n_1, n_2$) holds the following convergence property that for any $n_1, n_2$

$$E[f(w_{n_1+n_2}) - f_*] \leq \frac{B}{n_1} r^{n_2}, 0 < r < 1$$

where $B > 0$ is a constant value.

**Proof:** From inequality (4), we have

$$E[f(w_{n_1+n_2}) - f_*] \leq r^{n_2} E(f(w_{n_1}) - f_*)$$

According to (6), we can obtain

$$E(f(w_{n_1}) - f_*) \leq \frac{B}{n_1}$$

Substituting (8) to (7), we obtain

$$E[f(w_{n_1+n_2}) - f_*] \leq \frac{B}{n_1} r^{n_2}$$

**Theorem 2:** Suppose the objective function $f(w)$ satisfy Assumption 6.1 of [4] and the prerequisites of Theorem 1 in [16], SGD-LBFGS($n_1, n_2$) holds following convergence property that for any $n_1, n_2$

$$E[\|w_{n_1+n_2} - w_*\|^2] \leq \frac{C}{n_1} r^{n_2}, 0 < r < 1$$

where $C > 0$ is a constant value.

**Proof:** Similar to the proof of Theorem 1, from inequality (5), we can get

$$E[\|w_{n_1+n_2} - w_*\|^2] \leq Ar^{n_2} E(f(w_{n_1}) - f_*)$$

According to inequality (6), the following inequality holds:

$$E(f(w_{n_1}) - f_*) \leq \frac{B}{n_1}$$

Substituting (10) to (9), we obtain:

$$E[\|w_{n_1+n_2} - w_*\|^2] \leq \frac{C}{n_1} r^{n_2}$$

where $C = AB$.

Let $t_1$ and $t_2$ be the execution times of SGD and L-BFGS in one iteration, respectively. Assume $t_2 = nt_1$ where $n > 0$. We can easily prove that the execution time of SGD-LBFGS($n_1, n_2$), SGD($n_1 + n_2$) and L-BFGS($n_2 + n_1/n$) is equal.

**Theorem 3:** If the objective function $f(w)$ satisfies Assumption 6.1 of [4] and the prerequisites of Theorem 1 in [16], the following two propositions hold:

- When $r \geq e^{-\pi r}$, there exist $n_1 > 0$ and $n_2 > 0$ such that:

$$\frac{B}{n_1} r^{n_2} \leq r^{n_2+n_1/n}$$

where $B > 0$ is a constant. When $B = -n \log e$, SGD-LBFGS($n_1, n_2$) achieves the best convergence performance, in comparison with L-BFGS($n_2 + n_1/n$).

- There exist $n_1$ and $n_2$ such that the following inequality holds:

$$\frac{B}{n_1} r^{n_2} \leq \frac{1}{n_1 + n n_2}$$

where $B > 0$ is a constant. When $B \leq 1$, SGD-LBFGS($n_1, n_2$) converges faster than L-BFGS($n_1 + n_2$) for any $n_1 \geq -n \log e$ and $n_2 \geq - \log e - n_1/n$.

**Proof:** To prove the first proposition, let

$$g(n_1) = \frac{B}{n_1} r^{n_2} = r^{n_2+n_1/n} \frac{1}{n_1}$$

Minimizing $g(n_1)$ with respect to $n_1$, we get

$$n_1^* = \arg \min_{n_1} g(n_1) = -n \log e.$$

$$g(n_1^*) = \frac{B}{n_1^*} = \frac{B}{n \log e}.$$

In practice, we can use $n_1^*$ as a cross over point like $T_*$ in Fig. 1.

As $r \geq e^{-\pi r}$, $g(n_1^*) \leq 1$ so that $\frac{B}{n_1^*} r^{n_2} \leq r^{n_2+n_1/n}$.

This shows that SGD-LBFGS($n_1, n_2$) converges faster than L-BFGS($n_2 + n_1/n$). When $n_1 = n_1^*$, SGD-LBFGS achieves the best convergence performance.

To prove the second proposition, let

$$h(n_1, n_2) = \frac{B}{n_1} r^{n_2} = r^{n_2} \frac{B(n_1 + n_2)}{n_1}$$

The convergence rate for SGD is set as $1/\log$ and the convergence rate for L-BFGS is set as $0.9^t$. SGD runs faster than L-BFGS before the cross over point $T_*$ and becomes slower than L-BFGS after $T_*$. 
Replacing $n_1$ by $x$ and $n_2$ by $y$, we have

$$h(x, y) = r^y \frac{B(x + ny)}{x}.$$  

Compute the first derivative of $h(x, y)$ with respect to $x$:

$$\frac{\partial h}{\partial x} = -Br^ynyx^{-2} < 0$$
Similarly, compute the first derivative with respect to \( y \):
\[
\frac{\partial h}{\partial y} = B r^1((ny/x + 1) \ln r + n/x)
\]
Let \( \frac{\partial h}{\partial y} = 0 \) and by some manipulation, we get
\[
y^* = -\log_e e - x/n
\]
and
\[
y < y^*, \frac{\partial h}{\partial y} > 0
\]
\[
y > y^*, \frac{\partial h}{\partial y} < 0
\]
Let \( x = -n \log_e e, y = y^* \). We have
\[
h(x, y) = B
\]
When \( B > 1 \), as \( h(n_1, +\infty) = 0 \), we can find an appropriate \( n_2'(n_2') > -\log_e e - n_1/n \) to make \( h(n_1, n_2') = 1 \). Therefore, when \( n_2 \geq n_2' \), \( h(n_1, n_2) \leq 1 \).
When \( B \leq 1 \), for any \( n_1 \geq -n \log_e e \) and \( n_2 \geq -\log_e e - n_1/n \), \( h(n_1, n_2) \leq 1 \).

Theorem 3 shows that SGD-LBFGS converges faster than SGD and L-BFGS under certain conditions. SGD-LBFGS \((n_1, n_2)\) achieves its best convergence performance compared with L-BFGS \((n_2 + n_1/n)\) when \( n_1 = -n \log_e e \).
In the next section, we demonstrate these properties through empirical studies.

IV. EXPERIMENTS

We chose two famous linear classification algorithms, logistic regression and support vector machine to evaluate the hybrid optimization method SGD-LBFGS with real world data sets. We compared the performance of SGD-LBFGS and its two component methods, SGD and L-BFGS. The experiment results showed that SGD-LBFGS converged faster than L-BFGS, and was more stable than SGD.

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary of date sets</strong></td>
</tr>
<tr>
<td>Data set</td>
</tr>
<tr>
<td>icjnn1</td>
</tr>
<tr>
<td>a9a</td>
</tr>
<tr>
<td>w8a</td>
</tr>
<tr>
<td>real-sim</td>
</tr>
<tr>
<td>rcv1.binary</td>
</tr>
<tr>
<td>webspam</td>
</tr>
</tbody>
</table>

In this section, we present some experiment results from the datasets in Table I, which includes six datasets icjnn1, a9a, w8a, real-sim, rcv1.binary and webspam. The data sets were taken from LIBSVM [17]. Data icjnn1 was used in IJCNN challenge 2001 [18]. We performed preprocessing on the data with the same transformation methods in [19]. Data a9a was used to predict whether income exceeds $50K/yr. Data w8a was used to classify whether a web page belonged to a category or not. Data real-sim is real/simulated text classification dataset from the SRAA corpus of UseNet articles, and we used the first 80% of samples as training data, remaining samples as test data. Data rcv1.binary is a large benchmark data set for text categorization. Data webspam was used to identify whether the webpage was spam.

In the experiments, we set parameters \( n_1 = l \) and \( n_2 \geq 0 \), where \( l \) is the number of instances in the data set. The reason is that one iteration in SGD uses only one instance of training data whereas one iteration of L-BFSGS uses all instances in the training data, so the time cost of one iteration in L-BFSGS approximates that of \( l \) iterations in SGD. In reference to Theorem 3, \( n \) is assumed as \( l \), and \( r \) is assumed as \( e^{-1} \). Therefore, we chose \( n_1 = l \) and \( n_2 \geq 0 \). Parameter \( \lambda \) in objective function (2) was chosen from the range of \{1e-1, 1e-2, 1e-3, 1e-4, 1e-5\} for each data set, and a specific value was determined by a five-fold cross validation.

Owing to the stochastic property, we ran SGD and SGD-LBFGS ten times on each data set and calculated the average test error and the average training time for evaluation. All algorithms were initialized with \( w_0 = 0 \). Due to unstable convergence of SGD, we didn’t set any stop condition for SGD. For both L-BFGS and SGD-LBFGS, the program stops when the following form holds:
\[
\frac{\| \nabla f(w) \|}{\| w \|} \leq \varepsilon
\]

In the following sections, we present evaluation results of building logistic regression and support vector machine models on the six data sets.

A. Experiments on Logistic Regression

For building a logistic regression model, we rewrote the objective function (2) as
\[
\min_w f(w) = \frac{1}{l} \sum_{i=1}^{l} \left( \frac{\lambda}{2} w_i^2 + \log(1 + e^{-y_i w_i x_i}) \right)
\]
(11)
It is easy to verify that \( f(w) \) of (11) satisfies the prerequisites of Theorem 3.

We used SGD, L-BFGS and SGD-LBFGS to build a series of logistic regression models on six data sets and computed the average test errors and the average training time. Fig. 2 shows the training time versus test error on the six data sets.

From Fig. 2, we can see that the test errors of SGD and SGD-LBFGS drop much faster than the test error of L-BFGS in the early iterations. However, for the two medium-size dataset icjnn1 and w8a, SGD fluctuates after the test error dropped to a certain level whereas the test errors SGD-LBFGS drops continuously in the later iterations until the test errors approach to minimum. We can see that SGD-LBFGS always converges early in less time than L-BFGS on all six data sets, whereas SGD fluctuates around a test error level. These figures demonstrated a superior performance of SGD-LBFGS to both SGD and L-BFGS in building logistic regression models.
Table II lists the training time of L-BFGS and SGD-LBFGS on the six data sets. We can see that SGD-LBFGS was about 77% faster than L-BFGS.

To compare SGD-LBFGS with SGD, we analyzed the stability of the two methods. Inspired by [20], we define a stability metric $SC$ as

$$SC = \frac{1}{\sum_{i=start}^{end} ||err(i) - err(i-1)||}$$

where $err(i)$ is the error rate of SGD-LBFGS($l, i$) or SGD($i+1$), $start$ is an integer specifying the iteration from which the error rate drops below $(1 + TR) * Err$ ($TR$ is a threshold set as $TR = 0.1$ where $Err$ is the error rate of L-BFGS at the end of the optimization process, and $end$ indicates the last iteration of the optimization process for each data set as shown in Fig. 2. Table III shows the comparison of stability $SC$ between SGD and SGD-LBFGS on six data sets. From Table III, we can see that SGD-LBFGS was about 149% more stable than SGD.

To analyze the influence of the iteration number $n_1$ specified in the first step of SGD, we tested SGD-LBFGS with different values of $n_1$ on logistic regression. Table IV shows results of training time for different values of parameter $n_1$ in SGD-LBFGS($n_1, n_2$). We can see that the optimal $n_1$ for different data sets fluctuated around $l$, the total number of instances in data. This result is consistent with the theoretical analysis.

B. Experiments on Support Vector Machine

For support vector machine, we rewrote the objective function (2) as

$$\min_w f(w) = \frac{1}{l} \sum_{i=1}^{l} \left( \frac{\lambda}{2} w^2 + \max(0, 1 - y_i w^T x_i) \right)$$

Although the objective function $f(w)$ is unsmooth, and does not completely satisfy the prerequisites of Theorem 3, the experiment results showed that SGD-LBFGS still produced better results compared with SGD and L-BFGS.

Fig. 3 shows the training time versus the test error on the six data sets. From these figures, we can see the similar trends like logistic regression that the test errors of SGD and SGD-LBFGS fluctuated after the test error dropped to a certain level whereas the test errors of L-BFGS and SGD-LBFGS dropped continuously in the later iterations until the test errors approached to minimum.

Besides, comparing Fig. 2 with Fig. 3, we can find that for support vector machine, SGD-LBFGS does not show significant advantage over L-BFGS, and this phenomenon is owing to that the objective function of support vector machine is unsmooth, thus difficult to converge. In addition, the initial sudden drop of SGD-LBFGS in Fig. 3(a) may be attributed to the fact that both much noisy data and unsmooth objective function make transition from SGD to L-BFGS unsmooth.

We can see that although the objective function $f(w)$ of support vector machine does not meet the prerequisites of Theorem 3, SGD-LBFGS still achieved the similar results as logistic regression. Therefore, SGD-LBFGS is still applicable to support vector machine model building and can accelerate the convergence.

V. DISCUSSION

In this paper, we have presented the hybrid optimization method SGD-LBFGS that integrates two optimization methods SGD and L-BFGS to improve optimization performance
of building linear classification models. We theoretically investigated the advantages of the hybrid method and presented experiment results on logistic regression and support vector machine to demonstrate the advantages of SGD-LBFGS in convergence speed and stability over both L-BFGS and SGD.

The experiment results showed that SGD-LBFGS behaved well in building support vector machine models even the objective function does not meet theoretical requirements completely.

There are several remaining open issues to be investigated.
Specifically, the reason why SGD-LBFGS works fine on support vector machine needs to be explained. Another interesting topic is that whether integrating online learning methods and batch learning methods can also work in other machine learning algorithms.

REFERENCES


