J. Nonlinear Var. Anal. 7 (2023), No. 4, pp. 581-605
Available online at http://jnva.biemdas.com
https://doi.org/10.23952/jnva.7.2023.4.08

SEMI-IMPLICIT BACK PROPAGATION

REN LIU[1], XIAOQUN ZHANG[2,*]

[1]*School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai, China*
[2]*Institute of Natural Sciences, Shanghai Jiao Tong University, Shanghai, China*

Abstract. Deep neural network (DNN) has been attracting a great attention in various applications. Network training algorithms play essential roles for the effectiveness of DNN. Although stochastic gradient descent (SGD) and other explicit gradient-based methods are the most popular algorithms, there are still many challenges such as gradient vanishing and explosion occurring in training a complex and deep neural networks. Motivated by the idea of error back propagation (BP) and proximal point methods (PPM), we propose a semi-implicit back propagation method for neural network training. Similar to the BP, the update on the neurons are propagated in a backward fashion and the parameters are optimized with proximal mapping. The implicit update for both hidden neurons and parameters allows to choose large step size in the training algorithm. Theoretically, we demonstrate the convergence of the proposed method under some standard assumptions. The experiments on illustrative examples, and two real data sets: MNIST and CIFAR-10, demonstrate that the proposed semi-implicit BP algorithm leads to better performance in terms of both loss decreasing and training/test accuracy, with a detail comparison to SGD/Adam and a similar algorithm proximal back propagation (ProxBP).

Keywords. Back propagation; Neural network; Proximal mapping; SGD.

1. Introduction

Along with the rapid development of computer hardware, deep neural network methods achieved enormous success in diverse application fields, such as computer vision [1], speech recognition [2, 3], and nature language processing [4]. The key ingredient of neural network methods amounts to solve a highly non-convex optimization problem. The most basic and popular algorithm is stochastic gradient descent (SGD) [5] in the form of ”error” back propagation (BP) [6], which generally leads to high efficiency for training deep neural networks. Since then many variants of gradient based methods have been proposed, such as Adagrad[7], Nesterov momentum [8, 9], RMSprop [10], Adam [11] and AMSGrad [12]. Recently many research are dedicated to develop second-order algorithms, for example Newton method [13], quasi-Newton method [14], L-BFGS [15], and natural gradient method [16, 17, 18, 19, 20].

It is well known that the convergence of explicit gradient descent type approaches relies on sufficiently small step size. For example, for a loss function with Lipschitz continuous gradient, the stepsize should be in the range of $(0,2\beta)$ for $1/\beta$ being the Lipschitz constant, which is

*Corresponding author.

E-mail addresses: liur0810@sjtu.edu.cn (R. Liu), xqzhang@sjtu.edu.cn (X. Zhang).

Received April 14, 2023; Accepted May 10, 2023.

in general extremely large for real datasets. Another difficulty in gradient descent approaches is to propagate the ”error” deeply, which is commonly known as gradient vanishing due to composition with nonlinear activation functions. Implicit updates are proposed to partially overcome these problems. In [21], proximal back propagation, namely ProxBP, was proposed to utilize the proximal method for the weight updating. Alternative approach is to reformulate the training problem as a sequence of constrained optimization by introducing the constraints on the weights and hidden neurons at each layer. Block coordinate descent (BCD) methods [22, 23, 24, 25, 26, 27, 28] were also proposed and analyzed to solve this problem in constrained formulation. Without the multiconvexity and differentiability assumptions as in [29], the convergence of BCD for solving the learning problem was established in [27] based on the Kurdyka-Łojasiewicz (KŁ) inequality [30, 31]. LPOM[32] algorithm was proposed and used to train neural networks by solving a block multi-convex problem. Along this line, the Alternating direction method of multipliers (ADMM) [33, 34] were proposed with extra dual variables updating. In addition to the above methods, layer-parallel training methods[35, 36, 37] consider training residual networks as a optimal control problem. The SEDONA [38] algorithm divides a neural network into multiple sub-networks and trains them in a greedy manner, which improves the efficiency of neural network training.

Motivated by the work with implicit weight updates to overcome small step sizes and vanishing gradient problems in SGD, we propose a semi-implicit scheme, which has similar form as ”error” back propagation through neurons, while the parameters can be updated through solving small size optimization problems at each layer. It can be shown that the sequence generated by the scheme leads to a decreasing objective function and converges to a stationary point. In contrast to explicit gradient descent methods, the proposed method allows to choose large step sizes and leads to a better training performance per epoch. The performance is also stable with respect to the choice of stepsizes. In some sense, our semi-implicit scheme is similar to BCD methods as both are Gauss-Seidel type to propagate the error with its last updated parameters. BCD methods essentially optimize the merit function including neural network loss and the penalty term due to the constraints, while our scheme ensure the descent of the original neural network loss. Compared to the implicit method ProxBP, the errors in the proposed scheme are updated in a more implicit way, for which better training and test performances are achieved based on the experiments on several standard datasets.

The organization of the paper is as follows: Section 2 present some necessary notations. Section 3 present the background. In Section 4, we present the proposed semi-implicit back propagation algorithm. The numerical experiments are provided in Section 5. Finally we conclude the paper in Section 6.

2. Notations

Given input-output data pairs $(X,Y) = \{x_i, y_i\}_{i=1}^n$, we consider a N-layer feed-forward fully connected neural network for the simplification of notations. The network structure is demonstrated in Figure 1. Here, the parameters from the i-th layer to the $(i+1)$-th layer are the weight matrix W_i and bias b_i, and σ is a non-linear activation function, such as sigmod or ReLU. We denote the neuron vector at i-th layer before activation as G_i, and the neurons after activation as

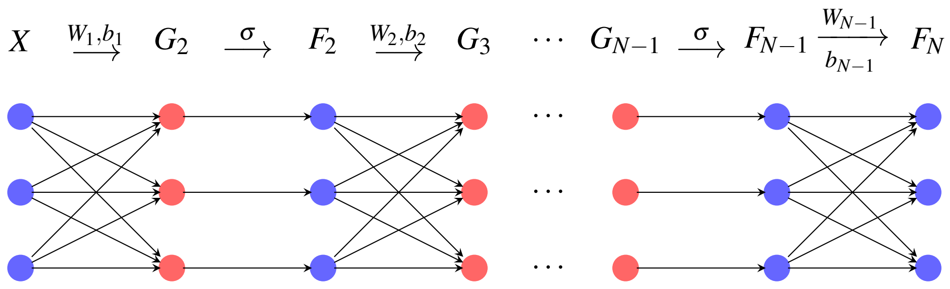

$X \xrightarrow{W_1,b_1} G_2 \xrightarrow{\sigma} F_2 \xrightarrow{W_2,b_2} G_3 \cdots G_{N-1} \xrightarrow{\sigma} F_{N-1} \xrightarrow[b_{N-1}]{W_{N-1}} F_N$

FIGURE 1. N-layer Neural Network

F_i, i.e. $F_1 = X$, and for $i = 1, \cdots, N-1$

$$\begin{aligned} G_{i+1} &= W_i F_i + b_i; \\ F_{i+1} &= \sigma(G_{i+1}). \end{aligned} \tag{2.1}$$

We note that, in general at the last layer, there is no non-linear activation function and $F_N = G_N$. For ease of notation, we can use an activation function σ as identity at the last layer. It should be noticed that non-linear activation function at the last layer can be directly adapted from the algorithm. The generic training model aims to solve the following minimization problem:

$$\min_{\Theta} J(\Theta; X, Y) := L(F_N, Y) \tag{2.2}$$

where Θ denotes the collective parameter set $\{W_i, b_i\}_{i=1}^{N-1}$ and L is a cost function with 0 lower bound.

3. BACKGROUND

We introduce the classical back propagation method and two implicit methods: Block coordinate descent and Proximal back propagation (ProxBP). In order to simplify notations, we leave out the bias term in this section.

3.1. **Back propagation method.** The widely used BP method [6] is based on gradient descent algorithm:

$$\Theta^{k+1} = \Theta^k - \eta \frac{\partial J(\Theta^k; X, Y)}{\partial \Theta},$$

where $\eta > 0$ is the stepsize. The main idea of the BP algorithm is to use an efficient error propagation scheme on the hidden neurons for computing the partial derivatives of the network parameters at each layer. According to equation (2.1) and (2.2), we denote $\frac{\partial J}{\partial F_i}$ to be the partial derivative of the loss function J with respect to the neurons F_i at i-th layer. The so-called "error" signal $\delta_N \equiv \frac{\partial J}{\partial F_N}$ at the last level is propagated to the hidden neurons $\delta_i \equiv \frac{\partial J}{\partial F_i}$ using the chain rule. In fact, for a square loss function, the gradient at the last layer $\delta_N = \frac{\partial L(F_N, Y)}{\partial F_N} = F_N - Y$ is indeed corresponding to an error. Let $\odot$ denote Hadamard product. The propagation from δ_{i+1} to δ_i for $i = N-1, \cdots, 1$ is then calculated as

$$\delta_i : \frac{\partial J}{\partial F_i} = \frac{\partial G_{i+1}}{\partial F_i} \frac{\partial F_{i+1}}{\partial G_{i+1}} \frac{\partial J}{\partial F_{i+1}} = W_i^T (\partial \sigma(G_{i+1}) \odot \delta_{i+1}), \tag{3.1}$$

and the partial derivative to W_i can be computed as

$$\frac{\partial J}{\partial W_i} := \frac{\partial G_{i+1}}{\partial W_i}\frac{\partial F_{i+1}}{\partial G_{i+1}}\frac{\partial J}{\partial F_{i+1}} = (\partial\sigma(G_{i+1}) \odot \delta_{i+1})F_i^T. \tag{3.2}$$

At $k-$th iteration, after a forward update of the neurons G_i^k, F_i^k by (2.1) using the current parameter set $\{W_i^k\}_{i=1}^{N-1}$, we can compute the "error" signal at each neurons sequentially from the $i+1$ level to i level by (3.1) and the parameters W_i^{k+1} is updated according to the gradient at the point W_i^k computed by (3.2).

3.2. **Proximal back propagation.** The algorithm ProxBP [21] developed from the classical BP method also uses the chain rule to propagate the error (perturbation) back for updating the weights and neurons. To introduce ProxBP, we leave out the bias term and start an iteration from $\Theta^k = \{W_i^k\}_{i=1}^{N-1}$. Firstly, we need to propagate the error back like BP method to calculate the gradient $\frac{\partial J}{\partial G_{i+1}}$ based on $\{W_i^k\}_{i=1}^{N-1}$. This gradient is denoted as $\frac{\partial J}{\partial G_{i+1}^k}$. Secondly, we update the parameter set $\{W_i^k\}_{i=1}^{N-1}$ as

$$W_i^{k+1} = \arg\min_{W_i} ||W_iF_i^k - G_{i+1}^k + \eta\frac{\partial J}{\partial G_{i+1}^k}||^2 + \frac{1}{2\lambda}||W_i - W_i^k||^2, \text{ for } i = 1,\cdots,N-1,$$

where η and λ denote the stepsize. Finally, we need to forward propagate according to the parameter set $\{W_i^{k+1}\}_{i=1}^{N-1}$. Different from BP, ProxBP update the parameter implicitly by solving several quadratic programming problems.

3.3. **Block coordinate descent.** In the following, we present the splitting BCD method proposed in [27]. The BCD method considers to solve the following penalized merit function J_{BCD}

$$J_{\text{BCD}} := L(F_N,Y) + \frac{\gamma}{2}\sum_{i=1}^{N-1}||F_{i+1} - \sigma(W_{i-1}F_{i-1})||^2.$$

Given $\{W_i^k\}_{i=1}^{N-1}$ and $\{F_i^k\}_{i=1}^{N}$, the updates can be written as

$$\begin{aligned}
F_N^{k+1} &= \arg\min_{F_N} L(F_N,Y) + \frac{\gamma}{2}||F_N - \sigma(W_{N-1}^kF_{N-1}^k)||^2,\\
W_i^{k+1} &= \arg\min_{W_i} \frac{\gamma}{2}||\sigma(W_iF_i^k) - F_{i+1}^{k+1}||^2 + \frac{1}{2\lambda_i}||W_i - W_i^k||^2,\\
F_i^{k+1} &= \arg\min_{F_i} \frac{\gamma}{2}||F_{i+1}^{k+1} - \sigma(W_i^{k+1}F_i)||^2 + \frac{\gamma}{2}||F_i - \sigma(W_{i-1}^kF_{i-1}^k)||^2 + \frac{1}{2\lambda_i}||F_i - F_i^k||^2,\\
&\text{for } i = N-1,\cdots,2.
\end{aligned}$$

To simplify, the bias term is left out. The BCD method do not contain the classical forward and backward propagation compared to BP and ProxBP. Correspondingly, the penalty function is brought in BCD to solve the constraints between layers. When updating the parameter set $\{W_i^k\}_{i=1}^{N-1}$, the BCD method takes a similar implicit approach like ProxBP. Theoretically, BCD ensures the descent of the merit function J_{BCD} but not the original loss function J.

4. Semi-implicit Back Propagation Method

In this section, we present the proposed semi-implicit back propagation method and establish the convergence theory.

4.1. **Semi-implicit back propagation.** Compared to the BP method, ProxBP and BCD have solved the gradient vanishing problem to a certain degree, as they both update the parameter set in an implicit way. Inspired by ProxBP and BCD, we propose to update the parameter set implicitly. At the iteration k, given the current estimate $\Theta^k : \{W_i^k, b_i^k\}_{i=1}^{N-1}$, we first update the neuron F_i^k and G_i^k in a feedforward fashion as BP method by using (2.1) for $i = 1, \cdots, N-1$. For the backward stage, we start with updating neuron F_N at the last layer using the gradient descent:

$$\delta_N^k = \frac{\partial L(F_N^k, Y)}{\partial F_N}, \quad F_N^{k+\frac{1}{2}} = F_N^k - \eta \delta_N^k.$$

For $i = N-1, \cdots, 1$, given $F_{i+1}^{k+\frac{1}{2}}$, the parameters W_i, b_i are updated by solving the following optimization problem sequentially

$$\begin{cases} W_i^{k+1} = \arg\min_{W_i} \|\sigma(W_i F_i^k + b_i^k) - F_{i+1}^{k+\frac{1}{2}}\|^2 + \frac{1}{2\lambda_i}\|W_i - W_i^k\|^2, \\ b_i^{k+1} = \arg\min_{b_i} \|\sigma(W_i^{k+1} F_i^k + b_i) - F_{i+1}^{k+\frac{1}{2}}\|^2 + \frac{1}{2\lambda_i}\|b_i - b_i^k\|^2, \end{cases} \tag{4.1}$$

where $\lambda_i > 0$ is a parameter that is corresponding to stepsize, and we consider the Frobenious norm for matrix norm and distance. The two subproblems at each layer can be solved by a nonlinear conjugate gradient method. This update of parameters is related to using an implicit gradient based on so-called proximal mapping. Taking W_i as an example, the optimality in (4.1) gives $W_i^{k+1} = W_i^k - \lambda_i \nabla f(W_i^{k+1})$, where $f(W_i) = \|\sigma(W_i F_i^k + b_i^k) - F_{i+1}^{k+\frac{1}{2}}\|^2$. Compared to a direct gradient descent step, this update is unconditionally stable for any stepsize λ. We note that the proximal mapping was previously proposed for training neural network as ProxBP in [21]. However, the update of the parameter set in (4.1) is different as ProxBP uses G_{i+1} for the data fitting at each layer. Using the neural F_{i+1} for data fitting can avoid differentiating non-smooth activation function σ temporarily.

After the update of W_i^{k+1} and b_i^{k+1}, we need to update the hidden neuron F_i. As in classical BP, we first consider the gradient at F_i^k as

$$\frac{\partial J}{\partial F_i^k} = \frac{\partial F_{i+1}^k}{\partial F_i^k} \frac{\partial J}{\partial F_{i+1}^k} = \frac{\partial G_{i+1}^k}{\partial F_i^k} \frac{\partial F_{i+1}^k}{\partial G_{i+1}^k} \frac{\partial J}{\partial F_{i+1}^k}.$$

It can be seen that the partial derivative $\frac{\partial G_{i+1}^k}{\partial F_i^k} := W_i^k$. Different from BP and ProxBP, we use the newly updated W_i^{k+1} instead of W_i^k to compute the error:

$$\delta_i^k := (W_i^{k+1})^T \cdot (\partial\sigma(G_{i+1}^k) \odot \delta_{i+1}^k), \quad F_i^{k+\frac{1}{2}} = F_i^k - \eta \delta_i^k. \tag{4.2}$$

By this formula, the difference can be propagated from the level N to 1. At the last level $i = 1$, we only need to update W_1^{k+1} and b_1^{k+1} as $F_1 = X$. The parameters $\{W_j^{k+1}\}_{j=i+1}^{N-1}$ updated previously are used when W_i^k is updated to W_i^{k+1}, which is similar to the coordinate descent method and Gauss-seidel method. The overall semi-implicit back propagation (SIBP) method is summarized in Algorithm 1.

Algorithm 1 Semi-implicit back propagation

```
**Input:** Current parameters $\Theta^k = \{W_i^k, b_i^k\}$
// ***Forward pass***
$F_1 = X$
**for** $i = 1$ **to** $N-1$ **do**
    $G_{i+1}^k = W_i^k F_i^k + b_i^k$
    $F_{i+1}^k = \sigma(G_{i+1}^k)$
**end for**
$\delta_N^k = \frac{\partial L(F_N^k, Y)}{\partial F_N}$
Update on $F_N^k \xrightarrow{\delta_N^k} F_N^{k+\frac{1}{2}}$
**for** $i = N-1$ **to** 2 **do**
    Implicit update on $(W_i^k, b_i^k) \xrightarrow{F_{i+1}^{k+\frac{1}{2}}} (W_i^{k+1}, b_i^{k+1})$ by (4.1)
    Error propagation $\delta_{i+1}^k \xrightarrow{W_i^{k+1}} \delta_i^k$ by (4.2)
    Explicit update on $F_i^k \xrightarrow{\delta_i^k} F_i^{k+\frac{1}{2}}$ by (4.2)
**end for**
Implicit update on $(W_1^k, b_1^k) \xrightarrow{F_2^{k+\frac{1}{2}}} (W_1^{k+1}, b_1^{k+1})$
**Output:** New parameters $\Theta^{k+1} = \{W_i^{k+1}, b_i^{k+1}\}$
```

Remark 4.1. The algorithm can be easily adapted for solving training with regularization. For example, when there is a separable regularization for each W_i and b_i, i.e.

$$\min_{\Theta} J(\Theta; X, Y) := L(F_N, Y) + \gamma r(\Theta),$$

where $\gamma > 0$. Then we can add directly in (4.1) as

$$W_i^{k+1} = \arg\min_{W_i} \|\sigma(W_i F_i^k + b_i^k) - F_{i+1}^{k+\frac{1}{2}}\|^2 + \frac{1}{2\lambda_i}\|W_i - W_i^k\|^2 + \gamma r(W_i)$$

$$b_i^{k+1} = \arg\min_{b_i} \|\sigma(W_i^{k+1} F_i^k + b_i) - F_{i+1}^{k+\frac{1}{2}}\|^2 + \frac{1}{2\lambda_i}\|b_i - b_i^k\|^2 + \gamma r(b_i).$$

This problem can be easily solved in small size when r is a simple function, for example ℓ_1 regularization.

Remark 4.2. For training with large number of samples, the BP method is used in the form of stochastic gradient descent (SGD) by using a small set of samples at each iteration. The proposed SIBP method can be easily extended to mini-batch version by replacing $(X, Y) = \{x_i, y_i\}_{i=1}^n$ by a subset $\{x_i, y_i\}_{i \in \mathscr{B}_k}$ at $k-$th iteration where $\mathscr{B}_k$ denotes the random mini-batch index set. Both the convergence and numerical experiments in mini-batch SIBP will be shown in later sections.

4.2. **Comparison to BP, ProxBP, and BCD.** The algorithms BP, ProxBP, and SIBP all use the chain rule to propagate the error back for updating the weights and neurons except the BCD method. Different from BP and ProxBP, both BCD and SIBP update the neurons sequentially in Gauss-Seidel type. For updating the weights $\{W_i\}_{i=1}^{N-1}$, BCD, ProxBP and SIBP all need to solve an implicit subproblem, while SIBP and BCD solve from the next-level's updated neurons after activation. The updates on the neuron variables F_i^{k+1} in BCD and $F_i^{k+\frac{1}{2}}$ in SIBP are conceptionally different, as the former aims to solve the two constraints resulted penalty function, while the latter update in a backward propagation fashion as BP method. It can also shown that SIBP ensure the descent of the original loss function in next section. Practically, SIBP can be easily extended to mini-batch version, while the extension of BCD to stochastic setting is unknown, which may limit its application to large scale problems. We summarize the property of these four algorithms in Table 1.

	BP	ProxBP	BCD	SIBP
Propagate by chain rule	✓	✓	×	✓
Implicit subproblem	×	✓	✓	✓
Implicit subproblem(including activation)	×	×	✓	✓
Descent of neural network loss	✓	✓	×	✓
Gauss-Seidel type update	×	×	✓	✓

TABLE 1. Comparison between BP, ProxBP, BCD and SIBP.

4.3. **Convergence analysis.** In the following, we drop the bias term b and the regularization. We only consider the set of weights $\Theta = W = \{W_i\}_{i=1}^{N-1}$ for the simplicity of proof. To simplify, we let the loss function $J(\Theta;X,Y) = J(W)$. The following are some assumptions.

Assumption 4.1. The loss function $J(W) \in C^2$, coercive and bounded below.

Assumption 4.2. The activation function $\sigma \in C^2(\mathbb{R})$ and $\sigma' \geq 0$.

Remark 4.3. Assumption 4.2 is strong and many activation functions do not satisfy the second-order smoothness, such as ReLU and leaky ReLU. Nevertheless, these kinds of activation function are usually smooth almost everywhere. Considering this fact, we can still use SIBP, a variant of BP, to train a neural network with ReLU or other activation functions as a practical algorithm.

Theorem 4.1 (Convergence of SIBP). *Suppose that Assumptions 4.1 and 4.2 hold. For any start point W^0, there exists $\bar{\eta} > 0$, which is dependent on $\{\lambda_i\}$ and W^0. When $0 < \eta < \bar{\eta}$, the sequence $J(W^k)$ decreases and converges, and $\nabla J(W^k)$ converges to $\mathbf{0}$, i.e.,*

$$J(W^{k+1}) \leq J(W^k) \leq \cdots \leq J(W^0),$$
$$\lim_{k\to\infty} ||\nabla J(W^k)|| = 0.$$

And there exists a subsequence of $\{W^k\}$ that converges to a stationary point of the loss function J.

Proof. Given that the loss function J is coercive, the sub-level set $\{W|J(W) \leq J(W^0)\} \subseteq B(r_0)$ for $B(r_0)$ being a closed ball with radius r_0. The radius r_0 determines the upper bound of W^0, F_i^0

and G_i^0 by the activation function and the loss function. Recall the solution at i-th layer is given by

$$W_i^1 = \arg\min_{W_i} \|\sigma(W_i F_i^0) - F_{i+1}^{\frac{1}{2}}\|^2 + \frac{1}{2\lambda_i}\|W_i - W_i^0\|^2. \tag{4.3}$$

By the optimality, we obtain

$$\|\sigma(W_i^1 F_i^0) - F_{i+1}^{\frac{1}{2}}\|^2 + \frac{1}{2\lambda_i}\|W_i^1 - W_i^0\|^2 \le \|\sigma(W_i^0 F_i^0) - F_{i+1}^{\frac{1}{2}}\|^2 = \|F_{i+1}^0 - F_{i+1}^{\frac{1}{2}}\|^2.$$

Considering that $F_{i+1}^{\frac{1}{2}} = F_{i+1}^0 - \eta\delta_{i+1}^0$, for $\eta < \bar{\eta}_0$, we have

$$||W_i^1 - W_i^0||^2 \le 2\lambda_i \eta^2 ||\delta_{i+1}^0||^2$$

and

$$||W_i^1|| \le ||W_i^0|| + \eta\sqrt{2\lambda_i}||\delta_{i+1}^0||.$$

For $0 \le i < N$, it can be demonstrated that $||\delta_{i+1}^0|| \le C_{i+1}(r_0, \bar{\eta}_0)$ for some $C_{i+1}(r_0, \bar{\eta}_0) > 0$ that only depends on the initial bound r_0 and $\bar{\eta}_0$. In fact, we first have $||\delta_{N-1}^0|| \le C_{N-1}(r_0, \bar{\eta}_0)$, and, for $\eta < \bar{\eta}_0$,

$$\begin{aligned}
||\delta_{i+1}^0|| &= ||{W_{i+1}^1}^T (\partial\sigma(G_{i+2}^0) \odot \delta_{i+2}^0)|| \\
&\le (||W_{i+1}^0|| + \eta\sqrt{2\lambda_{i+1}} C_{i+2}(r_0, \bar{\eta}_0))(\partial\sigma(G_{i+2}^0) \odot \delta_{i+2}^0)|| \\
&\le (r_0 + \bar{\eta}_0\sqrt{2\lambda_{i+1}} C_{i+2}(r_0, \bar{\eta}_0)) M(r_0) C_{i+2}(r_0, \bar{\eta}_0) \\
&:= C_{i+1}(r_0, \bar{\eta}_0).
\end{aligned}$$

Thus $||W_i^1 - W_i^0||^2 \le \eta^2 C_i(r_0, \bar{\eta}_0)$. Therefore, there exists a closed ball $B(r_1)$ for $r_1 \le r_0 + \bar{\eta}_0 \max_{1\le i\le N-1} C_i(r_0, \bar{\eta}_0)$ such that $W^0 \in B(r_0)$ and $W^1 \in B(r_1)$. By the smoothness of the function, it is easy to see $\nabla_W J$ is Lipschitz continuous on $B(r_1)$. Let l be the corresponding Lipschitz coefficient. We denote the partial derivative of $J(W^0)$ with respect to W_i as $\nabla_{W_i} J(W_i^0)$ for simplification, which leads to

$$J(W^1) \le J(W^0) + \sum_{i=1}^{N-1} \langle \nabla_{W_i} J(W_i^0), W_i^1 - W_i^0 \rangle + \frac{l}{2}\sum_{i=1}^{N-1} \|W_i^1 - W_i^0\|^2. \tag{4.4}$$

In the following, we prove that $J(W^1) \le J(W^0)$. We first consider the Taylor expansion of $\sigma(W_i^1 F_i^0)$ at the point $W_i^0 F_i^0$.

$$\sigma(W_i^1 F_i^0) = \sigma(W_i^0 F_i^0) + \partial\sigma(W_i^0 F_i^0) \odot (W_i^1 F_i^0 - W_i^0 F_i^0) + \mathbf{O}(||W_i^1 F_i^0 - W_i^0 F_i^0||^2).$$

Combining with the optimality of (4.3), we have

$$\begin{aligned}
W_i^1 &= W_i^0 - \lambda_i \nabla_{W_i} f_i(W_i^1) \\
&= W_i^0 - \lambda_i \partial\sigma(W_i^1 F_i^0) \odot (\sigma(W_i^1 F_i^0) - F_{i+1}^0 + \eta\delta_{i+1}^0){F_i^0}^T \qquad (4.5) \\
&= W_i^0 - \lambda_i \eta \partial\sigma(W_i^1 F_i^0) \odot \delta_{i+1}^0 {F_i^0}^T \\
&\quad - \lambda_i [\partial\sigma(W_i^1 F_i^0) \odot (\partial\sigma(W_i^0 F_i^0) \odot (W_i^1 F_i^0 - W_i^0 F_i^0))]{F_i^0}^T \\
&\quad - \lambda_i \partial\sigma(W_i^1 F_i^0) \odot \mathbf{O}(||W_i^1 F_i^0 - W_i^0 F_i^0||^2){F_i^0}^T. \qquad (4.6)
\end{aligned}$$

Reformulating (4.6) leads to

$$\begin{aligned}
&-\lambda_i\eta\partial\sigma(W_i^1F_i^0)\odot\delta_{i+1}^0{F_i^0}^T-\lambda_i\partial\sigma(W_i^1F_i^0)\odot\mathbf{O}(||W_i^1F_i^0-W_i^0F_i^0||^2){F_i^0}^T\\
=&W_i^1-W_i^0+\lambda_i[\partial\sigma(W_i^1F_i^0)\odot(\partial\sigma(W_i^0F_i^0)\odot(W_i^1F_i^0-W_i^0F_i^0))]{F_i^0}^T\\
=&W_i^1-W_i^0+\lambda_i[(W_i^1-W_i^0)F_i^0\odot\partial\sigma(W_i^0F_i^0)\odot\partial\sigma(W_i^1F_i^0)]{F_i^0}^T\\
\triangleq&(W_i^1-W_i^0)(I+\lambda_i\mathscr{A}_i^0)
\end{aligned}$$

where $\mathscr{A}_i^k$ is a semi-positive definite linear transform as $\sigma'\geq 0$

$$\mathscr{A}_i^0\triangleq[(\cdot F_i^0)\odot\partial\sigma(W_i^0F_i^0)\odot\partial\sigma(W_i^1F_i^0)]{F_i^0}^T. \tag{4.7}$$

The above formula can be rearranged as

$$\begin{aligned}
&\frac{1}{\lambda_i\eta}(W_i^1-W_i^0)(I+\lambda_i\mathscr{A}_i^0)\\
=&-\partial\sigma(W_i^1F_i^0)\odot\delta_{i+1}^0{F_i^0}^T-\frac{1}{\eta}\partial\sigma(W_i^1F_i^0)\odot\mathbf{O}(||W_i^1F_i^0-W_i^0F_i^0||^2){F_i^0}^T.
\end{aligned}\tag{4.8}$$

For the right hand side (RHS) of (4.8), we make a connection of the first term $\partial\sigma(W_i^1F_i^0)\odot\delta_{i+1}^0{F_i^0}^T$ with $\nabla_{W_i}J$, and the second term can be controlled by reducing η as $\mathbf{O}(||W_i^1F_i^0-W_i^0F_i^0||^2)\sim\mathbf{O}(\eta^2)$. Combining with (4.8), we consider

$$\begin{aligned}
&\langle\nabla_{W_i}J(W_i^0),W_i^1-W_i^0\rangle\\
=&\langle-\frac{1}{\lambda_i\eta}(W_i^1-W_i^0)(I+\lambda_i\mathscr{A}_i^0)+\nabla_{W_i}J(W_i^0)-\partial\sigma(W_i^1F_i^0)\odot\delta_{i+1}^0{F_i^0}^T\\
&-\frac{1}{\eta}\partial\sigma(W_i^1F_i^0)\odot\mathbf{O}(||W_i^1F_i^0-W_i^0F_i^0||^2){F_i^0}^T,W_i^1-W_i^0\rangle.
\end{aligned}\tag{4.9}$$

The RHS of (4.9) is split into three parts. In Appendix, we prove the following three inequalities which is essential to ensure $J(W^1)\leq J(W^0)$:

- **Part I**

$$\langle-\frac{1}{\lambda_i\eta}(W_i^1-W_i^0)(I+\lambda_i\mathscr{A}_i^0),W_i^1-W_i^0\rangle\leq-\frac{1}{\lambda_i\eta}||W_i^1-W_i^0||^2. \tag{4.10}$$

- **Part II**

$$\langle\nabla_{W_i}J(W_i^0)-\partial\sigma(W_i^1F_i^0)\odot\delta_{i+1}^0{F_i^0}^T,W_i^1-W_i^0\rangle\leq\sum_{j=i}^{N-1}A_{i,j}(r_0,\bar{\eta}_0)||W_j^1-W_j^0||^2. \tag{4.11}$$

- **Part III**

$$\langle-\frac{1}{\eta}\partial\sigma(W_i^1F_i^0)\odot\mathbf{O}(||W_i^1F_i^0-W_i^0F_i^0||^2){F_i^0}^T,W_i^1-W_i^0\rangle\leq B_i(r_0,\bar{\eta}_0)||W_i^1-W_i^0||^2. \tag{4.12}$$

By substituting these inequalities into (4.9), we can update (4.4) as

$$\begin{aligned}
J(W^1)&\leq J(W^0)+\sum_{i=1}^{N-1}\langle\nabla_{W_i}J(W_i^0),W_i^1-W_i^0\rangle+\sum_{i=1}^{N-1}\frac{l}{2}\|W_i^1-W_i^0\|^2\\
&\leq J(W^0)-(\frac{1}{\max(\lambda_i)\eta}-\gamma(r_0,\bar{\eta}_0))||W^1-W^0||^2.
\end{aligned}$$

The parameter $\gamma(r_0, \bar{\eta}_0)$ is positive and related to the aggregation of constants $A_{i,j}(r_0, \bar{\eta}_0)$, $B_i(r_0, \bar{\eta}_0)$, and l. Thus there exists $\eta \leq \bar{\eta}_1 = 1/(\max(\lambda_i)\gamma(r_0, \bar{\eta}_0))$ such that $J(W^1) \leq J(W^0)$, where $\bar{\eta}_1$ is only determined by r_0 and $\bar{\eta}_0$. Thus we can obtain $W^1 \in B(r_0)$ as $J(W^1) \leq J(W^0)$. By repeating this procedure, we can conclude that: if Assumptions 4.1 and 4.2 hold, there exists $\bar{\eta}_0, \bar{\eta}_1$ such that when $\eta < \bar{\eta} = \min(\bar{\eta}_0, \bar{\eta}_1)$ $0 < J(W^{k+1}) \leq J(W^k) \leq \cdots \leq J(W^0)$ and $\lim_{k\to\infty} J(W^k) - J(W^{k+1}) = 0$. Furthermore, we have

$$\lim_{k\to\infty} ||W_i^{k+1} - W_i^k||^2 = 0, i = 1 \cdots N-1. \tag{4.13}$$

Combined with equation (4.5), we obtain the following limit

$$W_i^{k+1} = W_i^k - \lambda_i \partial\sigma(W_i^{k+1}F_i^k) \odot (\sigma(W_i^{k+1}F_i^k) - F_{i+1}^k + \eta\delta_{i+1}^k) F_i^{k^T}$$
$$\Longrightarrow \lim_{k\to\infty} ||\lambda_i \partial\sigma(W_i^{k+1}F_i^k) \odot (\eta\delta_{i+1}^k) F_i^{k^T}|| = 0.$$

Using the inequality (A.2) in Appendix and changing the superscript, it yields

$$||\nabla_{W_i} J(W_i^k) - \partial\sigma(W_i^{k+1}F_i^k) \odot \delta_{i+1}^0 F_i^{0^T}||^2 \leq \sum_{j=i}^{N-1} \bar{D}_j(r_0, \bar{\eta}_0) ||W_j^{k+1} - W_j^k||^2.$$

Finally, we obtain $\lim_{k\to\infty} ||\nabla_{W_i} J(W_i^k)|| = 0$. By the boundedness of $\{W^k\}$ and equation (4.13), we can conclude that there exist a subsequence of $\{W^k\}$ that converges to a stationary point of the loss function J. □

In the following, we first introduce some new notations before presenting the convergence of mini-batch SIBP. Define

$$J(W) := \frac{1}{n}\sum_{t=1}^{n} J(W, x_t, y_t) = \frac{1}{n}\sum_{t=1}^{n} J_t(W).$$

Denote $\mathscr{B}_k \subseteq \{1, \cdots, n\}$ as a random mini-batch index set at iteration k. Define

$$J_{\mathscr{B}}(W) := \frac{1}{|\mathscr{B}|}\sum_{t\in\mathscr{B}} J_t(W),$$

where $|\mathscr{B}|$ denotes the number of elements in the mini-batch index set $\mathscr{B}$. Denote $\mathbb{B}$ as the collection of all mini-batch index sets, which means $\mathscr{B} \in \mathbb{B}$. Denote η_k as the stepsize of gradient descent applied on the activated neurons, when W^k is updated to W^{k+1}.

Assumption 4.3. $\mathbb{E}[\|\nabla_W J(W^k) - \nabla_W J_{\mathscr{B}_k}(W^k)\|^2] \leq V^2$ for some $V > 0$ and all $k \in \mathbb{N}$, where the expectation is taken over random choice of mini-batch in each step.

Theorem 4.2 (Convergence of mini-batch SIBP). *Suppose that Assumptions 4.1, 4.2, and 4.3 hold. For any initial point* W^0, *choose* η_k *such that* $0 < \eta_{k+1} \leq \eta_k \leq \eta_0 < \tilde{\eta}$, $\sum_{k=0}^{\infty}\eta_k = \infty$, *and* $\sum_{k=0}^{\infty}\eta_k^2 < \infty$, *for some* $\tilde{\eta} > 0$. *Then* $\lim_{k\to\infty}\mathbb{E}[\|W^{k+1} - W^k\|^2] = 0$ *and* $\liminf_{k\to\infty}\mathbb{E}[(\|W^{k+1} - W^k\|/\eta_k)^2] \leq \lambda^2 V^2$.

Proof. From the proof of Theorem 4.1, if $W^0 \in \{W | J(W) \leq \alpha\}$, then there exists a constant $\bar{\eta}$ such that, when $\eta < \bar{\eta}$,

$$\langle \nabla_W J(W^0), W^1 - W^0 \rangle + \frac{l}{2}\|W^1 - W^0\|^2 \leq -(\frac{1}{\max(\lambda_i)\eta} - \gamma(r_0, \bar{\eta}_0))||W^1 - W^0||^2. \tag{4.14}$$

Considering the mini-batch SIBP, the boundedness of sequence $\{W^k\}_{k=0}^{\infty}$ can be still established. Define $A = \cup_{\mathscr{B}\in\mathbb{B}}\{W|J_{\mathscr{B}}(W) \le J_{\mathscr{B}}(W^0)\}$ and the coercivity of the loss function $J_{\mathscr{B}}$ leads to the boundess of A. For any mini-batch index set $\mathscr{B} \in \mathbb{B}$, there exists a set $A_{\mathscr{B}}$ and a constant $\alpha_{\mathscr{B}}$ satisfying the following condition:

$$W^0 \in A \subseteq A_{\mathscr{B}} = \{W|J_{\mathscr{B}}(W) \le \alpha_{\mathscr{B}}\}, \forall \mathscr{B} \in \mathbb{B}.$$

As the choice of $\mathscr{B}_0$ is finite, if η_0 is upper bounded by a constant $\bar{\eta}_0$, there exist two constant r_0 and r_1 such that $W^0 \in A_{\mathscr{B}} \subseteq B(r_0)$ for all $\mathscr{B} \in \mathbb{B}$ and $W^1 \in B(r_1)$. Let $l_{\mathscr{B}}$ be the local Lipschitz constant of $\nabla_W J_{\mathscr{B}}(W)$ in the set $B(r_1)$ and l be the local Lipschitz constant of $\nabla_W J(W)$ in the set $B(r_1)$. Denote $l_{\max} = \max_{\mathscr{B}\in\mathbb{B}}(l_{\mathscr{B}}, l)$. For each $\mathscr{B} \in \mathbb{B}$, if W^1 is updated by the mini-batch index set $\mathscr{B}$, by exploiting inequality (4.14), there exists a corresponding constant $\bar{\eta}_1(\mathscr{B}) = \min(\bar{\eta}_0, 1/(\max(\lambda_i)\gamma_{\mathscr{B}}(r_0,\bar{\eta}_0)))$, when $\eta_0 < \bar{\eta}_1(\mathscr{B})$, we have:

$$\begin{aligned}&\langle \nabla_W J_{\mathscr{B}}(W^0), W^1 - W^0\rangle + \frac{l_{\max}}{2}\|W^1 - W^0\|^2 \\ &\le -(\frac{1}{\max(\lambda_i)\eta_0} - \gamma_{\mathscr{B}}(r_0,\bar{\eta}_0))||W^1 - W^0||^2. \end{aligned} \tag{4.15}$$

It should be noted that we use $l_{\max}$ instead of $l_{\mathscr{B}}$ in inequality (4.15) and the corresponding loss function is $J_{\mathscr{B}}$. Furthermore, one has

$$\begin{aligned} J_{\mathscr{B}}(W^1) &\le J_{\mathscr{B}}(W^0) + \langle \nabla_W J_{\mathscr{B}}(W^0), W^1 - W^0\rangle + \frac{l_{\mathscr{B}}}{2}\|W^1 - W^0\|^2 \\ &\le J_{\mathscr{B}}(W^0) + \langle \nabla_W J_{\mathscr{B}}(W^0), W^1 - W^0\rangle + \frac{l_{\max}}{2}\|W^1 - W^0\|^2 \\ &< J_{\mathscr{B}}(W^0). \end{aligned}$$

Define $\tilde{\eta} = \frac{1}{2}\min_{\mathscr{B}\in\mathbb{B}}(\bar{\eta}_0, \bar{\eta}_1(\mathscr{B}))$ and let $\eta_0 < 2\tilde{\eta}$. For all $\mathscr{B} \in \mathbb{B}$, it holds

$$\begin{aligned}&\langle \nabla_W J_{\mathscr{B}}(W^0), W^1 - W^0\rangle + \frac{l_{\max}}{2}\|W^1 - W^0\|^2 \\ &\le -(\frac{1}{\max(\lambda_i)\eta_0} - \gamma_{\mathscr{B}}(r_0,\bar{\eta}_0))||W^1 - W^0||^2 \\ &\le -(\frac{1}{\max(\lambda_i)\eta_0} - \max_{\mathscr{B}\in\mathbb{B}}\gamma_{\mathscr{B}}(r_0,\bar{\eta}_0))||W^1 - W^0||^2 \\ &< 0. \end{aligned}$$

By randomly choosing a mini-batch index set $\mathscr{B}_0$ to update W^0, if $\eta_0 < 2\tilde{\eta}$, one obtains

$$\begin{aligned} J_{\mathscr{B}_0}(W^1) \le& J_{\mathscr{B}_0}(W^0) + \langle \nabla_W J_{\mathscr{B}_0}(W^0), W^1 - W^0\rangle + \frac{l_{\mathscr{B}_0}}{2}\|W^1 - W^0\|^2 \\ \le& J_{\mathscr{B}_0}(W^0) + \langle \nabla_W J_{\mathscr{B}_0}(W^0), W^1 - W^0\rangle + \frac{l_{\max}}{2}\|W^1 - W^0\|^2 \\ \le& J_{\mathscr{B}_0}(W^0) - (\frac{1}{\max(\lambda_i)\eta_0} - \gamma_{\mathscr{B}_0}(r_0,\bar{\eta}_0))||W^1 - W^0||^2 \\ \le& J_{\mathscr{B}_0}(W^0) - (\frac{1}{\max(\lambda_i)\eta_0} - \max_{\mathscr{B}\in\mathbb{B}}\gamma_{\mathscr{B}}(r_0,\bar{\eta}_0))||W^1 - W^0||^2 \\ \le& J_{\mathscr{B}_0}(W^0). \end{aligned}$$

Therefore $W^1 \in \{W | J_{\mathscr{B}_0}(W) \leq J_{\mathscr{B}_0}(W^0)\} \subseteq A$. Under the following condition, $\eta_k < 2\tilde{\eta}$ for all $k \geq 0$, we can repeat this procedure. Therefore, the sequence $\{W^k\}_{k=0}^{\infty}$ is bounded and for all $k, W^k \in A$. In the following, we consider the stepsize sequence $\{\eta_k\}_{k=0}^{\infty}$ under the conditions as below $0 < \eta_{k+1} \leq \eta_k \leq \eta_0 < \tilde{\eta}$, $\sum_{k=0}^{\infty} \eta_k = \infty$, and $\sum_{k=0}^{\infty} (\eta_k)^2 < \infty$. To simplify, let $\lambda = \max(\lambda_i)$ and $\gamma = \max_{\mathscr{B} \in \mathbb{B}} \gamma_{\mathscr{B}}(r_0, \bar{\eta}_0)$. We have

$$\eta_0 < \tilde{\eta} \leq \frac{1}{2} \min_{\mathscr{B} \in \mathbb{B}} (\bar{\eta}_1(\mathscr{B})) = \frac{1}{2\lambda\gamma}.$$

Consider the local Lipschitz continuity of $\nabla_W J(W)$ and establish the following inequality

$$\begin{aligned} J(W^{k+1}) \leq & J(W^k) + \langle \nabla_W J(W^k), W^{k+1} - W^k \rangle + \frac{l}{2} \|W^{k+1} - W^k\|^2 \\ = & J(W^k) + \langle \nabla_W J_{\mathscr{B}_k}(W^k), W^{k+1} - W^k \rangle + \frac{l}{2} \|W^{k+1} - W^k\|^2 \\ & + \langle \nabla_W J(W^k) - \nabla_W J_{\mathscr{B}_k}(W^k), W^{k+1} - W^k \rangle \\ \leq & J(W^k) - (\frac{1}{\lambda\eta_k} - \gamma) \|W^{k+1} - W^k\|^2 + \frac{1}{2\lambda\eta_k} \|W^{k+1} - W^k\|^2 \\ & + \frac{\lambda\eta_k}{2} \|\nabla_W J(W^k) - \nabla_W J_{\mathscr{B}_k}(W^k)\|^2. \end{aligned}$$

Taking the expectation gives

$$\mathbb{E}_k[J(W^{k+1})] \leq J(W^k) - (\frac{1}{2\lambda\eta_k} - \gamma) \mathbb{E}_k[\|W^{k+1} - W^k\|^2] + \frac{\lambda\eta_k}{2} V^2$$

where $\mathbb{E}_k$ means taking the expectation on the random choice of $\mathscr{B}_k$. Now taking the expectation on random choice of all steps before, we have

$$\mathbb{E}[J(W^{k+1})] \leq \mathbb{E}[J(W^k)] - (\frac{1}{2\lambda\eta_k} - \gamma) \mathbb{E}[\|W^{k+1} - W^k\|^2] + \frac{\lambda\eta_k}{2} V^2. \tag{4.16}$$

Multiplying by η_k and using $\eta_k < \tilde{\eta}$, one has

$$\begin{aligned} \eta_{k+1} \mathbb{E}[J(W^{k+1})] \leq & \eta_k \mathbb{E}[J(W^{k+1})] \\ & \leq \eta_k \mathbb{E}[J(W^k)] - (\frac{1}{2\lambda} - \gamma\eta_0) \mathbb{E}[\|W^{k+1} - W^k\|^2] + \frac{\lambda\eta_k^2}{2} V^2. \end{aligned}$$

Taking sum over k yields

$$\begin{aligned} \sum_{k=0}^{\infty} (\frac{1}{2\lambda} - \gamma\eta_0) \mathbb{E}[\|W^{k+1} - W^k\|^2] \leq & \tilde{\eta} J[W^0] - \lim_{k \to \infty} \eta_k \mathbb{E}[J(W^k)] + \sum_{k=0}^{\infty} \frac{\lambda\eta_k^2}{2} V^2 \\ & < \infty. \end{aligned}$$

Therefore, $\lim_{k \to \infty} \mathbb{E}[\|W^{k+1} - W^k\|^2] = 0$. Rearranging terms in inequality (4.16), we have

$$\eta_k[(\frac{1}{2\lambda} - \gamma\eta_k) \mathbb{E}[(\|W^{k+1} - W^k\|/\eta_k)^2] - \frac{\lambda}{2} V^2] \leq \mathbb{E}[J(W^k)] - \mathbb{E}[J(W^{k+1})].$$

Summing over k leads to

$$\sum_{k=0}^{\infty} \eta_k[(\frac{1}{2\lambda} - \gamma\eta_k) \mathbb{E}[(\|W^{k+1} - W^k\|/\eta_k)^2] - \frac{\lambda}{2} V^2] < \infty.$$

The condition $\sum_{k=0}^{\infty} \eta_k = \infty$ yields

$$\liminf_{k\to\infty} (\frac{1}{2\lambda} - \gamma\eta_k)\mathbb{E}[(\|W^{k+1} - W^k\|/\eta_k)^2] \leq \frac{\lambda}{2}V^2.$$

Thus

$$\liminf_{k\to\infty} \mathbb{E}[(\|W^{k+1} - W^k\|/\eta_k)^2] = \lambda^2 V^2.$$

□

5. Numerical Experiments

In this section, we compare the performance of BP, ProxBP [21], BCD [27] and the proposed SIBP using MNIST or CIFAT-10 datasets. As in the published code of BCD all the samples of dataset have to be loaded at once, we compare with BCD only using small data set such as MNIST. All the experiments are performed in Python with NVIDIA GeForce GTX 1080Ti and the same network settings and initializations are used for a fair comparison. The softmax cross-entropy for the loss function L and ReLU are used for the activation function, as usually chosen in classification problems

$$L(x, class) = -\log(\frac{\exp(x[class])}{\sum_j \exp(x[j])}),$$
$$\text{ReLU}(x) = \max(x, 0).$$

For the linear Conjugate Gradient used in ProxBP and nonlinear Conjugate Gradient in SIBP, the iterations number is set as 5. All the neural networks in the experiment exclude batch normalization.

5.1. **A gradient vanishing example.** In the following, we first consider the following synthesized optimization problem from [39]:

$$\arg\min_{w_1, w_2, \cdots, w_L \in \mathbb{R}} F(w) = (w_L w_{L-1} \cdots w_1 - 1)^2.$$

The gradient of the weight w_i is given as

$$\nabla_{w_i} F = 2 w_1 w_2 \cdots w_{i-1} w_{i+1} \cdots w_L (w_L w_{L-1} \cdots w_1 - 1).$$

The product of w_i leads to gradient vanishing and gradient explosion. For example, the gradient $\nabla_{w_i} F$ is extremely large for all $w_j = 2$ while it is extremely small for $w_j = 0.5$ for a large L. To illustrate the gradient vanishing/explosion problem, Figure 2 shows the gradient of $f(w) = (w^7 - 1)^2$ for all $w_i = w$. It can be seen that gradient vanishing occurs when $w \in [-0.8, 0.8]$ and gradient explosion occurs when w is outside of $[-0.8, 1.2]$.

In the simulation, we randomly generate 10 points $\{x_i, y_i\}_{i=1\cdots 10}$ and consider:

$$\arg\min_{w_1, w_2, \cdots, w_7 \in \mathbb{R}} F(w) = \sum_{i=1}^{10} (w_7 w_6 \cdots w_1 x_i - y_i)^2.$$

For both SIBP and SGD, the variable w_j is initialized as normal distribution $N(0.15, 0.01)$. The function loss and the difference $|w^{k+1} - w^k|$ are compared to identify whether gradient vanishing occurs. Figure 3 shows that SIBP can escape from the region where gradient vanishing occurs faster than SGD. The curve of the sequential difference of the first component

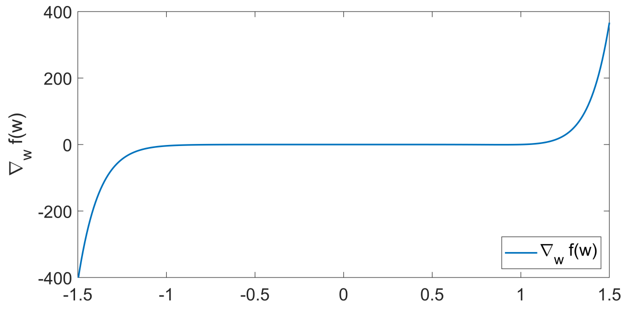
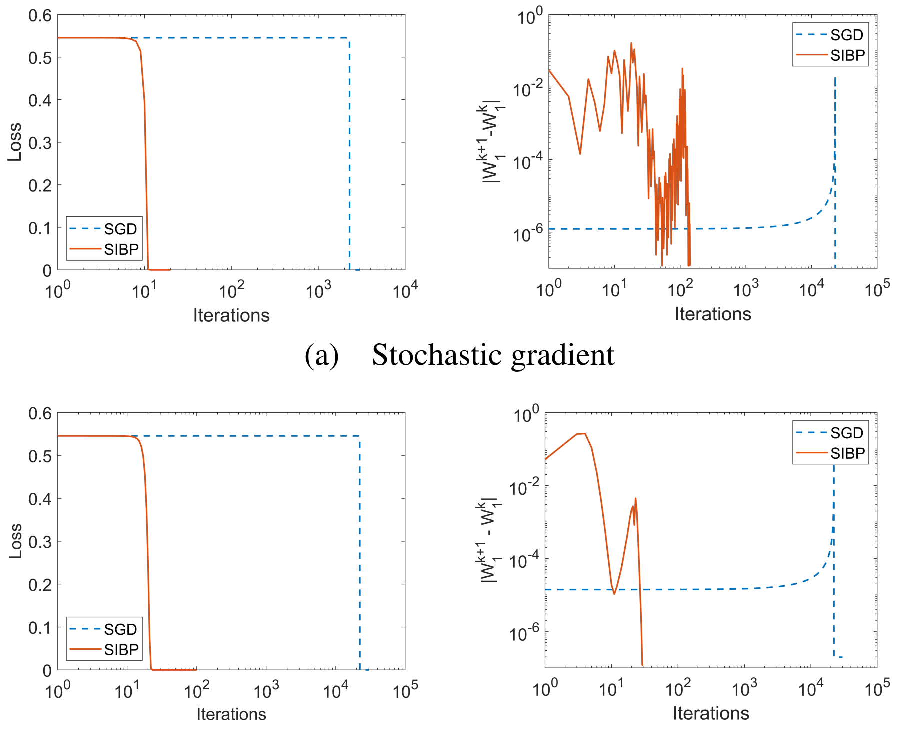


FIGURE 2. Gradient vanishing occurs when $w \in [-0.8, 0.8]$. Gradient explosion occurs when w is outside of $[-0.8, 1.2]$.



(a) Stochastic gradient

(b) Full batch gradient

FIGURE 3. Left: the loss function. Right: $|w_1^{k+1} - w_1^k|$.

$|w_1^{k+1} - w_1^k|$ shows that w_1^k of SGD moves more slowly compared to SIBP. When using stochastic gradient, the loss function decreases rapidly after 10 steps using SIBP, while it shows an obvious drop only after 1000 steps using SGD. A similar performance can be observed using full batch gradient.

5.2. **MNIST Classification.**

Shallow Network. In this part, we present an experiment using fully connected neural network on MNIST dataset. The training set contains 60000 samples and the rest 10000 samples are included in the test set. For this dataset, a shallow fully connected neural network of size

$784 \times 500 \times 10$ with ReLU activation function can usually reach 100% training accuracy by a few epochs. The training process are performed 5 times, and Table 2 shows the average training and test accuracy achieved by SGD, SIBP and ProxBP with different learning rates lr (for SGD) and λ, η for SIBP and ProxBP. We note that for both ProxBP and SIBP, λ refers to the proximal parameter and η refers to the gradient step size. It implies that after 50 epoch, the performance of SIBP method is stable with respect to different step size choices, while SGD fails for some choices of learning rate lr. For ProxBP, we present the results with $\eta = 0.1$ as the best performance is achieved with this set of parameters. The highest test accuracy scores are marked in bold in each column, and we can see that SIBP achieves the highest test accuracy compared to SGD and ProxBP.

MNIST	Training/Test accuracy				
SGD	$lr = 100$	$lr = 10$	$lr = 1$	$lr = 0.1$	$lr = 0.01$
	$-/-$	$-/-$	1.0/**0.9826**	0.9842/0.9757	0.9295/0.9301
ProxBP, $\eta = 0.1$	$\lambda = 100$	$\lambda = 10$	$\lambda = 1$	$\lambda = 0.1$	$\lambda = 0.01$
	1/0.9727	1/0.9716	1/0.9719	1/**0.9732**	1/0.9723
SIBP, $\eta = 10$	$\lambda = 100$	$\lambda = 10$	$\lambda = 1$	$\lambda = 0.1$	$\lambda = 0.01$
	1/**0.9827**	1/0.9817	1/0.9822	1/0.9822	1/0.9827
SIBP, $\eta = 1$	$\lambda = 100$	$\lambda = 10$	$\lambda = 1$	$\lambda = 0.1$	$\lambda = 0.01$
	1/**0.9830**	1/ 0.9822	1/0.9828	1/0.9814	1/0.9817
SIBP, $\eta = 0.1$	$\lambda = 100$	$\lambda = 10$	$\lambda = 1$	$\lambda = 0.1$	$\lambda = 0.01$
	0.9899/0.9727	0.9891/**0.9783**	0.9899/0.9765	0.9886/0.9776	0.9857/0.9759

TABLE 2. SIBP is stable with different choice of parameter.

Deep Network. Gradient-based methods are usually hard to backpropagate the disturbation in a deep neural network due to gradient vanishing or explosion. In this part of experiment, we attempt to show the performance of SIBP in neural networks with a deep structure compared with ProxBP, BCD and SGD. For the MNIST dataset, we train a $784 \times 600^{10} \times 10$ fully connected neural network. This neural network has 10 hidden layers and each layer contains 600 neurons. We also use ReLU activation function between each layer to increase the non-linearity.

Case 1 The parameters were initialized with $N(0, 0.01)$. In Figure 4, SGD and ProxBP can barely optimize this deep network while BCD and SIBP show an advantage. At the same time, SIBP achieve more than 80% training accuracy with less epochs than BCD.

Case 2 The parameters were initialized with $N(0, 0.025)$. In Figure 5, it can be seen that SGD still fails to train the neural networks as the previous example, while SIBP takes the effect quicker compared to ProxBP and BCD.

Case 3 We further set the weight initialized by $N(0, 0.03)$, for SGD successes to train this neural network. Figure 6 demonstrates that SIBP arrives a good training accuracy in a much earlier stage compared to SGD, BCD and ProxBP.

At last, we compare the norm of gradient $||\frac{\partial J}{\partial W_1}||$ in SIBP and SGD. Figure 7 shows that the norm of the gradient from SIBP decreases and becomes stable more quickly.

5.3. **CIFAR-10 Classification.** In this part, we show the performance on CIFAR-10 dataset using a full connected network and a Convolutional neural network (CNN). Since BCD can

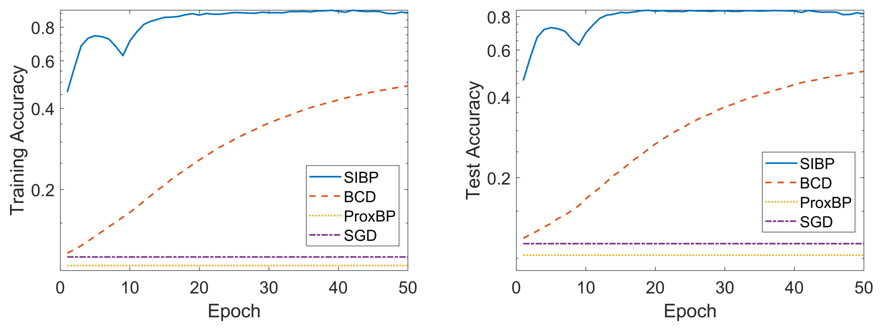


FIGURE 4. Both SGD and ProxBP suffer from gradient vanishing issue.

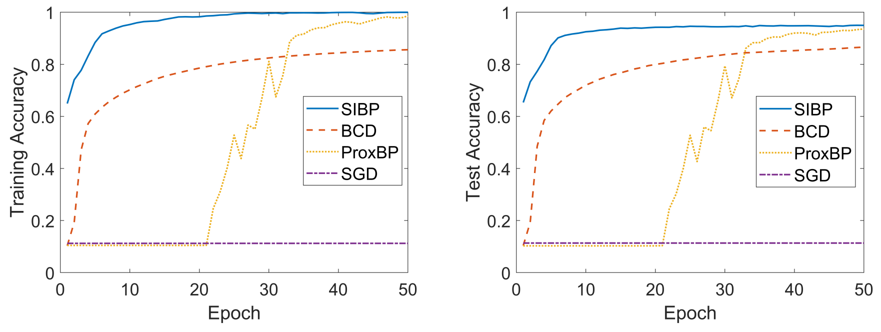


FIGURE 5. SGD fail, SIBP, BCD and ProxBP work.

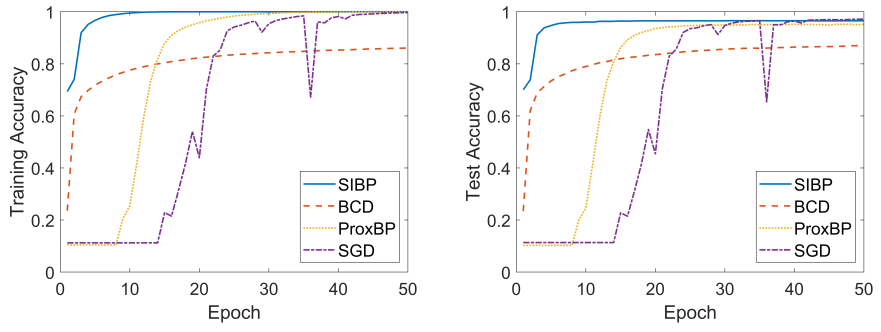
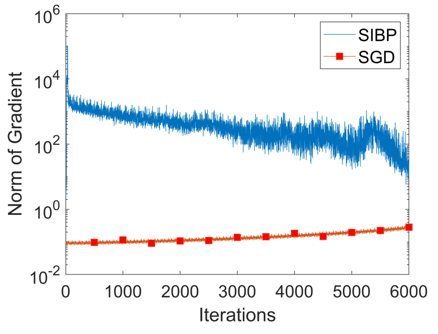
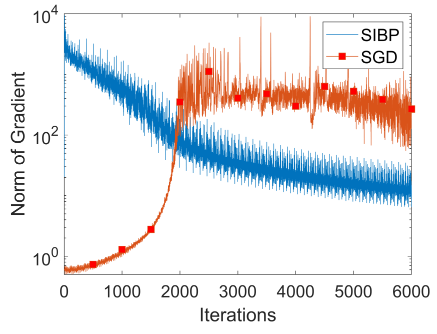


FIGURE 6. SIBP arrives at 100% accuracy much quicker than BCD, SGD and ProxBP.

not be applied in mini-batch form, here only SIBP, ProxBP and SGD(Adam) are drawn into comparisons. In Figure 8, the performance of the three methods per epoch for CIFAR-10 is shown with a neural network of size $3072 \times 2000 \times 500 \times 10$ with ReLU activation functions. The step size is set as $\eta = 0.1$ for both SIBP and ProxBP, while a smaller one for Adam to achieve a better performance. It can be seen that the trainning loss and trainning accuracy of SIBP decrease and increase faster respectively. The improvement on the test accuracy also demonstrates that the proposed SIBP method generalizes well in a comparison to the other two methods.



(a) Initialized by $N(0, 0.025)$



(b) Initialized by $N(0, 0.03)$

FIGURE 7. The norm of gradient $||\frac{\partial J}{\partial W_1}||$ from SIBP and SGD show that SIBP escapes from the region where gradient vanishing occurs faster than SGD.

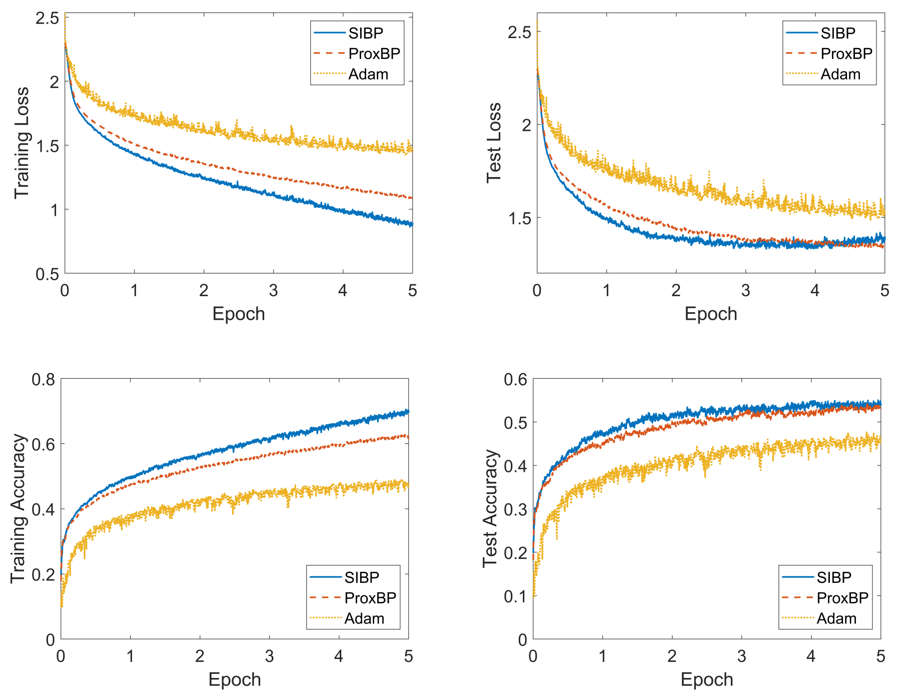
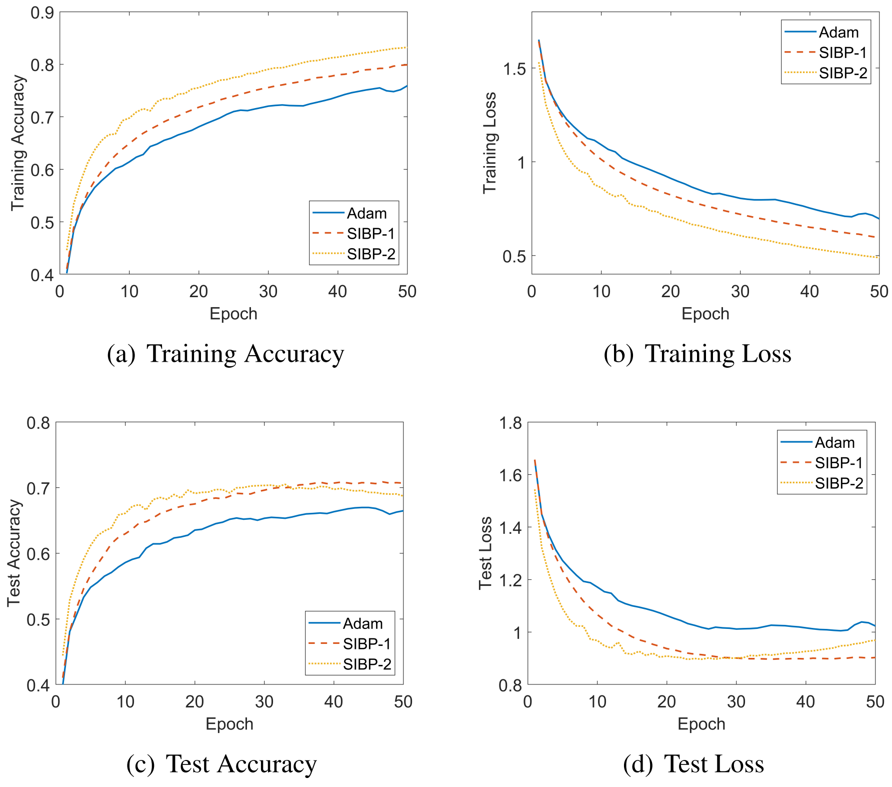


FIGURE 8. SIBP performs better than ProxBP and Adam both in training and test accuracy.

To illustrate the performance of SIBP on convolutional neural networks, we also train a LeNet on CIFAR-10 dataset. The architecture of CNN is:

$$\mathrm{Conv}[16*32*32] \to \mathrm{ReLU} \to \mathrm{Pool}[16*16*16] \to \mathrm{Conv}[20*16*16] \to \mathrm{ReLU} \to$$
$$\mathrm{Pool}[20*8*8] \to \mathrm{Conv}[20*8*8] \to \mathrm{ReLU} \to \mathrm{Pool}[20*4*4] \to \mathrm{FC} + \mathrm{Softmax}[10*1*1]$$

The size of each convolution kernel is $5*5$ and the factor of max pooling is 2. The result is shown in Figure 9. SIBP is slightly better than Adam in generalization. After 50 epochs, the test accuracy of SIBP reaches 70.55% while Adam's is around 69.6%. The curve SIBP-1 shows better generalization with stepsize $\eta = 0.5$ and the curve SIBP-2 shows slightly faster convergence with stepsize $\eta = 1$.



(a) Training Accuracy (b) Training Loss

(c) Test Accuracy (d) Test Loss

FIGURE 9. SIBP is slightly better than Adam in generalization.

Remark 5.1. To be noticed, for convolutional neural networks, the SIBP algorithm is mainly used to optimize the weights of the convolutional layers connected with activation functions. For pooling layers, we still use standard gradient descent for optimization.

5.4. **Training with small size samples.** In the training of neural network with small size samples, the landscape of the global minimum may be widespread [40]. Thus one may get a global minimum with 100% training accuracy or 0 training loss, while the testing performance can be very different. The comparison of the four methods in this setting is performed on MNIST and FashionMNIST dataset. First, a $784 \times 500 \times 10$ neural network containing ReLU activation functions is trained with 10000 training samples from MNIST. Figure 10 shows that SIBP reach 100% training accuracy with less epochs. For the test accuracy, SIBP reaches 96.7% after 20 epochs, which is still slightly higher than other methods. For the second experiment, a similar neural network with ReLU activation functions is trained on FashionMNIST dataset. A data set of 2000 training samples is used and the network size is $784 \times 1000 \times 10$. In Figure 11, SIBP achieves 82.25% accuracy on the test set which is slightly higher than other methods.

5.5. **Pruning Neural Networks by ℓ_1 penalty.** Neural network pruning is a method to simplify the connections and less the storage of trained networks, which leads to a fast test computation. By adding regularization term in the update of SIBP, we can control the sparsity of

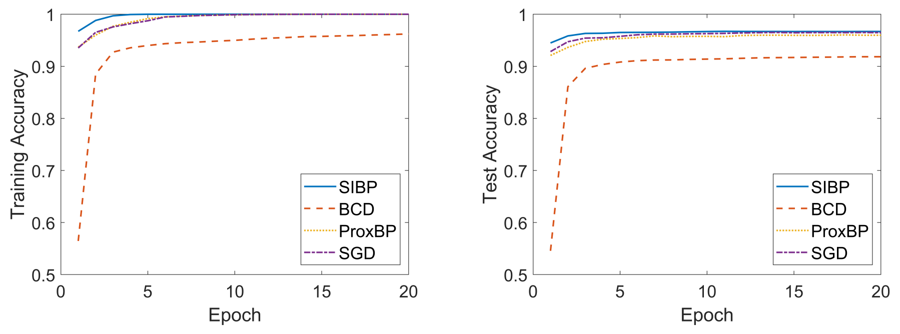


FIGURE 10. SIBP reach 100% training accuracy with less epochs on MNIST dataset.

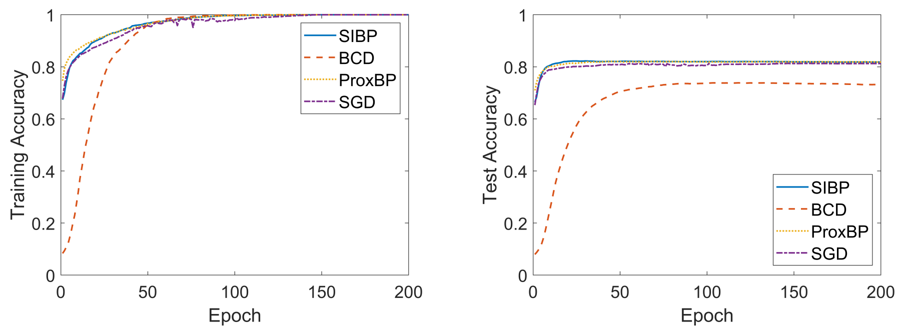


FIGURE 11. SIBP is slightly better in test accuracy on FashionMNIST dataset.

weight by adding a ℓ_1 term

$$W_i^{k+1} = \arg\min_{W_i} \|\sigma(W_i F_i^k + b_i^k) - F_{i+1}^{k+\frac{1}{2}}\|^2 + \frac{1}{2\lambda_i}\|W_i - W_i^k\|^2 + \gamma\|W_i\|_1$$

$$b_i^{k+1} = \arg\min_{b_i} \|\sigma(W_i^{k+1} F_i^k + b_i) - F_{i+1}^{k+\frac{1}{2}}\|^2 + \frac{1}{2\lambda_i}\|b_i - b_i^k\|^2 + \gamma\|b_i\|_1.$$

The norm $||\cdot||_1$ denotes the ℓ_1 norm of vector. To prune neural network, we train a LeNet-5 on MNIST dataset using ℓ_1 norm which is shown in Figure 12. With ℓ_1 regularization, SIBP can reach 98.5% training accuracy with 70% elements being zeros.

5.6. **Nonlinear CG steps.** In this section we show the effect of different nonlinear CG steps in SIBP method given a fixed computation time. Though more steps in nonlinear CG will lead to a more precise solution of sub problems, few steps cost less time and may lead to a faster convergence. We train a $3072 \times 4000 \times 1000 \times 4000 \times 10$ fully connected neural network with different steps in nonlinear CG. Figure 13 demonstrate that 5 iterations achieves higher performance in training accuracy in terms of computation time.

6. CONCLUSIONS

We proposed a novel optimization scheme in order to overcome the difficulties of small step-size and vanishing gradient in training neural networks. The computation of new scheme is in

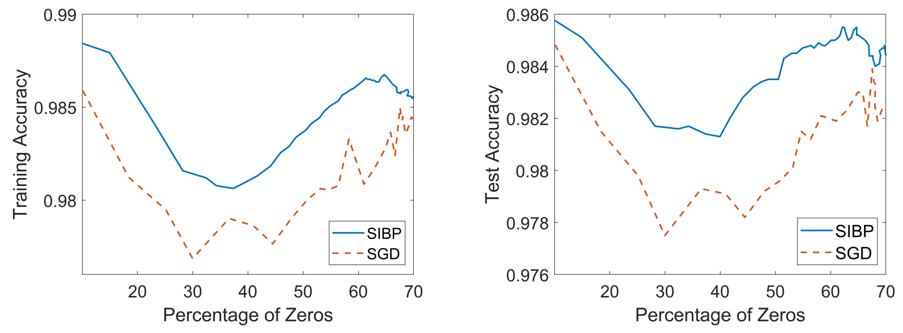


FIGURE 12. SIBP with ℓ_1 regularization is able to prune neural network as well as SGD.

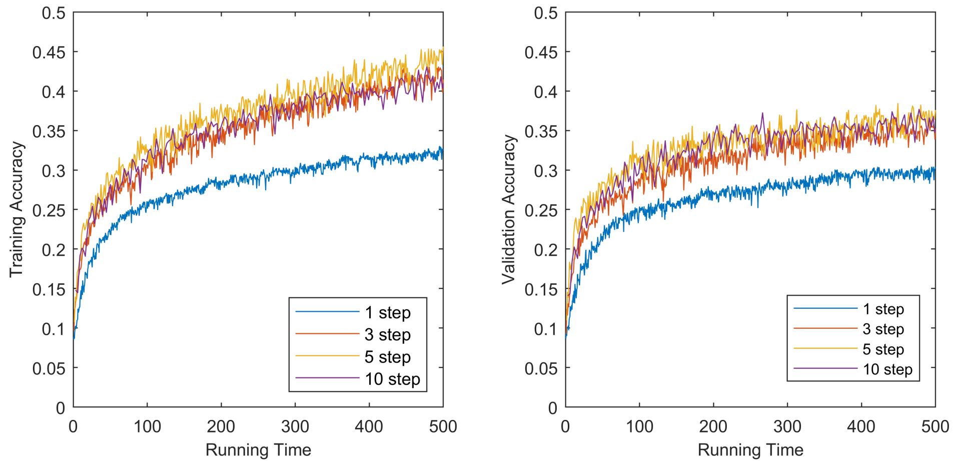


FIGURE 13. Too many redundant steps in nonlinear CG can be ineffective.

the spirit of error back propagation, with an implicit update on the parameters set and semi-implicit updates on the hidden neurons. The convergence result is established for the proposed algorithm under some standard assumptions. The experiments on an synthesize example, MNIST and CIFAR-10 demonstrate that the proposed SIBP has better performance per epoch compared to SGD and ProxBP. It is demonstrated in the experiment that larger step sizes can be adopted without losing stability and performance.

APPENDIX A. INEQUALITY PROOF

Part 1: For $\eta > 0$, inequality (4.10) holds, i.e.,

$$\langle -\frac{1}{\lambda_i\eta}(W_i^1 - W_i^0)(I + \lambda_i \mathscr{A}_i^0), W_i^1 - W_i^0\rangle \leq -\frac{1}{\lambda_i\eta}||W_i^1 - W_i^0||^2$$

where $\mathscr{A}_i^0$ is the linear transformation defined in (4.7).

Proof. Considering the s-th row vector $w_s^0, w_s^1, (W_i^0\mathscr{A}_i^0)_s$ of W_i^0, W_i^1 and $W_i^0\mathscr{A}_i^0$, we have

$$\langle (W_i^0\mathscr{A}_i^0)_s, w_s^0\rangle = (W_i^0\mathscr{A}_i^0)_s {w_s^0}^T = [w_s^0F_i^0 \odot \partial\sigma(w_s^0F_i^0) \odot \partial\sigma(w_s^1F_i^0)]{F_i^0}^T {w_s^0}^T \geq 0. \quad \text{(A.1)}$$

Aggregating inequality (A.1), we obtain

$$\langle W_i^0 \mathscr{A}_i^0, W_i^0 \rangle = \sum_s \langle (W_i^0 \mathscr{A}_i^0)_s, w_s^0 \rangle \geq 0.$$

Thus linear transformation $\mathscr{A}_i^0$ is symmetrical semi-positive. Hence, $\lambda_{\min}(I + \lambda_i \mathscr{A}_i^0) \geq 1$. Thus it is easy to see that

$$\langle -\frac{1}{\lambda_i \eta}(W_i^1 - W_i^0)(I + \lambda_i \mathscr{A}_i^0), W_i^1 - W_i^0 \rangle \leq -\frac{1}{\lambda_i \eta}||W_i^1 - W_i^0||^2.$$

$\square$

Part 2: If $\eta \leq \bar{\eta}_0$, then the following inequality holds, i.e.,

$$||\nabla_{W_i} J(W_i^0) - \partial\sigma(W_i^1 F_i^0) \odot \delta_{i+1}^0 F_i^{0^T}||^2 \leq \sum_{j=i}^{N-1} \bar{D}_j(r_0, \bar{\eta}_0)||W_j^1 - W_j^0||^2. \tag{A.2}$$

Therefore inequality (4.11) holds.

$$\langle \nabla_{W_i} J(W_i^0) - \partial\sigma(W_i^1 F_i^0) \odot \delta_{i+1}^0 F_i^{0^T}, W_i^1 - W_i^0 \rangle \leq \sum_{j=i}^{N-1} A_{i,j}(r_0, \bar{\eta}_0)||W_j^1 - W_j^0||^2.$$

All constants are determined by r_0, $\bar{\eta}_0$, $\{\lambda_i\}$, activation function σ, and the loss function J.

Proof. By the local Lipschitz continuity of $\partial\sigma$ in the bounded set $B(r_1)$, we have

$$||\partial\sigma(W_i^1 F_i^0)||^2 ||F_i^{0^T}||^2 \leq D(r_0, \bar{\eta}_0)$$

and

$$||\partial\sigma(W_i^1 F_i^0) - \partial\sigma(W_i^0 F_i^0)||^2 \leq \tilde{D}(r_0, \bar{\eta}_0)||W_i^0 - W_i^1||^2.$$

Firstly, we prove the following inequality

$$\begin{aligned}
&||\frac{\partial J}{\partial F_i^0} - \delta_i^0||^2 = ||W_i^{0^T}(\partial\sigma(W_i^0 F_i^0) \odot \frac{\partial J}{\partial F_{i+1}^0}) - W_i^{1^T}(\partial\sigma(W_i^0 F_i^0) \odot \delta_{i+1}^0)||^2 \\
\leq& 2||(W_i^1 - W_i^0)^T(\partial\sigma(W_i^0 F_i^0) \odot \frac{\partial J}{\partial F_{i+1}^0})||^2 + 2||W_i^{1^T}(\partial\sigma(W_i^0 F_i^0) \odot (\frac{\partial J}{\partial F_{i+1}^0} - \delta_{i+1}^0))||^2 \\
:=& D_i(r_0, \bar{\eta}_0)||W_i^1 - W_i^0||^2 + D_{i+1}^0(r_0, \bar{\eta}_0)||\frac{\partial J}{\partial F_{i+1}^0} - \delta_{i+1}^0||^2 \\
\leq& D_i(r_0, \bar{\eta}_0)||W_i^1 - W_i^0||^2 + D_{i+1}(r_0, \bar{\eta}_0)||W_{i+1}^1 - W_{i+1}^0|| + D_{i+2}^0(r_0, \bar{\eta}_0)||\frac{\partial J}{\partial F_{i+2}^0} - \delta_{i+2}^0||^2 \\
\leq& \sum_{j=i}^{N-1} D_i(r_0, \bar{\eta}_0)||W_j^1 - W_j^0||^2 + D_N^0(r_0, \bar{\eta}_0)||\frac{\partial J}{\partial F_N^0} - \delta_N^0||^2 \\
:=& \sum_{j=i}^{N-1} D_i(r_0, \bar{\eta}_0)||W_j^1 - W_j^0||^2.
\end{aligned}$$

Secondly, we can obtain that

$$\begin{aligned}
&||\nabla_{W_i}J(W_i^0) - \partial\sigma(W_i^1 F_i^0) \odot \delta_{i+1}^0 F_i^{0^T}||^2 \\
\leq & 2||(\partial\sigma(W_i^0 F_i^0) - \partial\sigma(W_i^1 F_i^0)) \odot \frac{\partial J}{\partial F_{i+1}^0} F_i^{0^T}||^2 + 2||\partial\sigma(W_i^1 F_i^0) \odot (\frac{\partial J}{\partial F_{i+1}^0} - \delta_{i+1}^0) F_i^{0^T}||^2 \\
\leq & 2\tilde{D}(r_0, \bar{\eta}_0)||\frac{\partial J}{\partial F_{i+1}^0}||^2 ||F_i^{0^T}||^2 ||W_i^0 - W_i^1||^2 + 2D(r_0, \bar{\eta}_0) * \sum_{j=i+1}^{N-1} D_j(r_0, \bar{\eta}_0)||W_j^1 - W_j^0||^2 \\
:= & \sum_{j=i}^{N-1} \bar{D}_j(r_0, \bar{\eta}_0)||W_j^1 - W_j^0||^2.
\end{aligned}$$

Finally, we have

$$\begin{aligned}
&\langle \nabla_{W_i}J(W_i^0) - \partial\sigma(W_i^1 F_i^0) \odot \delta_{i+1}^0 F_i^{0^T}, W_i^1 - W_i^0 \rangle \\
\leq & \frac{1}{2}||\nabla_{W_i}J(W_i^0) - \partial\sigma(W_i^1 F_i^0) \odot \delta_{i+1}^0 F_i^{0^T}||^2 + \frac{1}{2}||W_i^1 - W_i^0||^2 \\
\leq & \sum_{j=i}^{N-1} \frac{1}{2}\bar{D}_j(r_0, \bar{\eta}_0)||W_j^1 - W_j^0||^2 + \frac{1}{2}||W_i^1 - W_i^0||^2 \\
:= & \sum_{j=i}^{N-1} A_{i,j}(r_0, \bar{\eta}_0)||W_j^1 - W_j^0||^2.
\end{aligned}$$

□

Part 3: If $\eta \leq \bar{\eta}_0$, then the following inequality holds, i.e.,

$$\frac{1}{2\eta^2}||\partial\sigma(W_i^1 F_i^0) \odot \mathbf{O}(||W_i^1 F_i^0 - W_i^0 F_i^0||^2) F_i^{0^T}||^2 \leq E_i(r_0, \bar{\eta}_0)||W_i^1 - W_i^0||^2.$$

Therefore inequality (4.12) holds

$$\langle -\frac{1}{\eta}\partial\sigma(W_i^1 F_i^0) \odot \mathbf{O}(||W_i^1 F_i^0 - W_i^0 F_i^0||^2) F_i^{0^T}, W_i^1 - W_i^0 \rangle \leq B_i(r_0, \bar{\eta}_0)||W_i^1 - W_i^0||^2.$$

All constants are determined by r_0, $\bar{\eta}_0$, $\{\lambda_i\}$, activation function σ, and loss function J.

Proof. Recall that $\mathbf{O}(||W_i^1 F_i^0 - W_i^0 F_i^0||^2)$ is the second order remainder term of $\sigma(W_i^1 F_i^0) - \sigma(W_i^0 F_i^0)$ using Taylor expansion.

$$\sigma(W_i^1 F_i^0) - \sigma(W_i^0 F_i^0) = \partial\sigma(W_i^0 F_i^0) \odot (W_i^1 F_i^0 - W_i^0 F_i^0) + \mathbf{O}(||W_i^1 F_i^0 - W_i^0 F_i^0||^2).$$

Given $W^0, W^1 \in B(r_1)$, one has

$$||\mathbf{O}(||W_i^1 F_i^0 - W_i^0 F_i^0||^2)|| \leq \tilde{E}_i(r_0, \bar{\eta}_0)||W_i^1 F_i^0 - W_i^0 F_i^0||^2.$$

It follows that

$$\begin{aligned}
&\frac{1}{2\eta^2}||\partial\sigma(W_i^1F_i^0)\odot\mathbf{O}(||W_i^1F_i^0-W_i^0F_i^0||^2)F_i^{0^T}||^2\\
\leq&\frac{1}{2\eta^2}\tilde{E}_i(r_0,\bar{\eta}_0)^2||W_i^1F_i^0-W_i^0F_i^0||^4||\partial\sigma(W_i^1F_i^0)||^2||F_i^0||^2\\
\leq&\frac{1}{2\eta^2}\tilde{E}_i(r_0,\bar{\eta}_0)^2*2\lambda_i\eta^2C_i(r_0,\bar{\eta}_0)||W_i^1F_i^0-W_i^0F_i^0||^2||\partial\sigma(W_i^1F_i^0)||^2||F_i^0||^4\\
:=&E_i(r_0,\bar{\eta}_0)||W_i^1-W_i^0||^2.
\end{aligned}$$

Furthermore,

$$\begin{aligned}
&\langle-\frac{1}{\eta}\partial\sigma(W_i^1F_i^0)\odot\mathbf{O}(||W_i^1F_i^0-W_i^0F_i^0||^2)F_i^{0^T},W_i^1-W_i^0\rangle\\
\leq&\frac{1}{2\eta^2}||\partial\sigma(W_i^1F_i^0)\odot\mathbf{O}(||W_i^1F_i^0-W_i^0F_i^0||^2)F_i^{0^T}||^2+\frac{1}{2}||W_i^1-W_i^0||^2\\
\leq&E_i(r_0,\bar{\eta}_0)||W_i^1-W_i^0||^2+\frac{1}{2}||W_i^1-W_i^0||^2\\
:=&B_i(r_0,\bar{\eta}_0)||W_i^1-W_i^0||^2.
\end{aligned}$$

As above, constant B_i is determined by r_0, $\bar{\eta}_0$, $\{\lambda_i\}$, the activation function σ, and the loss function J. □

References

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, vol. 60, pp. 84–90, AcM New York, NY, USA, 2017.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, *et al.*, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Processing magazine, 29 (2012), 82-97.

[3] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, B. Ramabhadran, Improvements to deep convolutional neural networks for lvcsr, in: IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 315–320, IEEE, 2013.

[4] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (2-11), 2493-2537.

[5] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Statist. 22 (1951), 400–407.

[6] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nature, 323 (1986), 533-536.

[7] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Machn. Learn. Res. 12 (2011), 7.

[8] Y. E. Nesterov, A method of solving a convex programming problem with convergence rate o (1/kˆ 2), in: Doklady Akademii Nauk, vol. 269, pp. 543–547, Russian Academy of Sciences, 1983.

[9] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: International Conference on Machine Learning, pp. 1139–1147, PMLR, 2013.

[10] T. Tieleman, G. Hinton, Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning, Technical Report, 2017.

[11] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014.

[12] S. J. Reddi, S. Kale, S. Kumar, On the convergence of adam and beyond, arXiv preprint arXiv:1904.09237, 2019.

[13] G. Montavon, G. Orr, K.-R. Müller, Neural networks: tricks of the trade, vol. 7700, Springer, 2012.

[14] J. Martens, Deep learning via hessian-free optimization., in: ICML, vol. 27, pp. 735–742, 2010.

[15] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, A. Y. Ng, On optimization methods for deep learning, in: Proceedings of the 28th International Conference on International Conference on Machine Learning, pp. 265–272, 2011.

[16] S.-I. Amari, H. Park, K. Fukumizu, Adaptive method of realizing natural gradient learning for multilayer perceptrons, Neural Computation, 12 (2000), 1399–1409.

[17] J. Martens, New insights and perspectives on the natural gradient method, J. Mach. Learn. Res. 21 (2020),1–76.

[18] M. Yang, D. Xu, Z. Wen, M. Chen, P. Xu, Sketch-based empirical natural gradient methods for deep learning, J. Sci. Comput. 92 (2022), 94.

[19] J. Hu, R. Ao, A. M.-C. So, M. Yang, Z. Wen, Riemannian natural gradient methods, arXiv preprint arXiv:2207.07287, 2022.

[20] M. Yang, D. Xu, Q. Cui, Z. Wen, P. Xu, Ng+, A multi-step matrix-product natural gradient method for deep learning, arXiv preprint arXiv:2106.07454, 2021.

[21] T. Frerix, T. Möllenhoff, M. Moeller, D. Cremers, Proximal backpropagation, arXiv preprint arXiv:1706.04638, 2017.

[22] A. Askari, G. Negiar, R. Sambharya, L. E. Ghaoui, Lifted neural networks, arXiv preprint arXiv:1805.01532, 2018.

[23] M. Carreira-Perpinan, W. Wang, Distributed optimization of deeply nested systems, in: Artificial Intelligence and Statistics, pp. 10–19, PMLR, 2014.

[24] F. Gu, A. Askari, L. El Ghaoui, Fenchel lifted networks: A lagrange relaxation of neural network training, in: International Conference on Artificial Intelligence and Statistics, pp. 3362–3371, PMLR, 2020.

[25] T. T.-K. Lau, J. Zeng, B. Wu, Y. Yao, A proximal block coordinate descent algorithm for deep neural network training, in: International Conference on Learning Representations (ICLR), Workshop Track, 2018.

[26] Z. Zhang, M. Brand, Convergent block coordinate descent for training tikhonov regularized deep neural networks, vol. 30, 2017.

[27] J. Zeng, T. T.-K. Lau, S. Lin, Y. Yao, Global convergence of block coordinate descent in deep learning, in: International conference on machine learning, pp. 7313–7323, PMLR, 2019.

[28] Y. Xie, Z. Li, H. Zhao, Gradient-free neural network training based on deep dictionary learning with the log regularizer, in: Pattern Recognition and Computer Vision: 4th Chinese Conference, PRCV 2021, Beijing, China, October 29–November 1, 2021, Proceedings, Part IV 4, pp. 561–574, Springer, 2021.

[29] Y. Xu, W. Yin, A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion, SIAM J. Imaging Sci. 6 (2013), 1758–1789.

[30] K. Kurdyka, On gradients of functions definable in o-minimal structures, in Annales de l'institut Fourier, 48 (1998), 769–783.

[31] S. Łojasiewicz, Sur la géométrie semi-et sous-analytique, Annales de l'Institut Fourier 43 (1993), 1575–1595.

[32] J. Li, M. Xiao, C. Fang, Y. Dai, C. Xu, Z. Lin, Training neural networks by lifted proximal operator machines, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44 (2020), 3334–3348.

[33] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, T. Goldstein, Training neural networks without gradients: A scalable admm approach, in International Conference on Machine Learning, pp. 2722–2731, PMLR, 2016.

[34] Z. Zhang, Y. Chen, V. Saligrama, Efficient training of very deep neural networks for supervised hashing, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1487–1495, 2016.

[35] Q. Sun, H. Dong, Z. Chen, W. Dian, J. Sun, Y. Sun, Z. Li, B. Dong, Layer-parallel training of residual networks with auxiliary variables, in: The Symbiosis of Deep Learning and Differential Equations, 2021.

[36] S. Gunther, L. Ruthotto, J. B. Schroder, E. C. Cyr, N. R. Gauger, Layer-parallel training of deep residual neural networks, SIAM J. Math. Data Sci. 2 (2020), 1–23.

[37] A. Kirby, S. Samsi, M. Jones, A. Reuther, J. Kepner, V. Gadepally, Layer-parallel training with gpu concurrency of deep residual neural networks via nonlinear multigrid, in: 2020 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–7, IEEE, 2020.

[38] M. Pyeon, J. Moon, T. Hahn, G. Kim, Sedona: Search for decoupled neural networks toward greedy block-wise learning, in: International Conference on Learning Representations, 2021.

[39] R. Sun, Optimization for deep learning: theory and algorithms, arXiv preprint arXiv:1912.08957, 2019.

[40] L. Wu, C. Ma, W. E, *et al.*, How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective, 32nd Conference on Neural Information Processing Systems, NeurIPS, Montreal, 2018.