Global Convergence of Block Coordinate Descent in Deep Learning

Jinshan Zeng [1 2 *] **Tim Tsz-Kit Lau** [3 *] **Shao-Bo Lin** [4] **Yuan Yao** [2]

Abstract

Deep learning has aroused extensive attention due to its great empirical success. The efficiency of the block coordinate descent (BCD) methods has been recently demonstrated in deep neural network (DNN) training. However, theoretical studies on their convergence properties are limited due to the highly nonconvex nature of DNN training. In this paper, we aim at providing a general methodology for provable convergence guarantees for this type of methods. In particular, for most of the commonly used DNN training models involving both two- and three-splitting schemes, we establish the global convergence to a critical point at a rate of $\mathcal{O}(1/k)$, where k is the number of iterations. The results extend to general loss functions which have Lipschitz continuous gradients and deep residual networks (ResNets). Our key development adds several new elements to the Kurdyka-Łojasiewicz inequality framework that enables us to carry out the global convergence analysis of BCD in the general scenario of deep learning.

1. Introduction

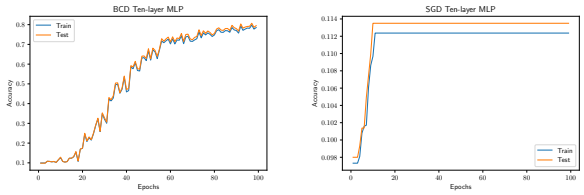
Tremendous research activities have been dedicated to deep learning due to its great success in some real-world applications such as image classification in computer vision (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012; Sainath et al., 2013), statistical machine translation (Devlin et al., 2014), and especially outperforming human in Go games (Silver et al., 2016).

The practical optimization algorithms for training neural networks can be mainly divided into three categories in terms of the amount of first- and second-order information used, namely, gradient-based, (approximate) second-order and gradient-free methods. Gradient-based methods make use of backpropagation (Rumelhart et al., 1986) to compute gradients of network parameters. Stochastic gradient descent (SGD) method proposed by Robbins & Monro (1951) serve as the basis. Much of research endeavour is devoted to adaptive variants of vanilla SGD in recent years, including AdaGrad (Duchi et al., 2011), RMSProp (Tieleman & Hinton, 2012), Adam (Kingma & Ba, 2015) and AMSGrad (Reddi et al., 2018). (Approximate) second-order methods mainly include Newton's method (LeCun et al., 2012), L-BFGS and conjugate gradient (Le et al., 2011). Despite the great success of these gradient-based methods, they may suffer from the vanishing gradient issue for training deep networks (Goodfellow et al., 2016). As an alternative to overcome this issue, gradient-free methods have been recently adapted to the DNN training, including (but not limited to) block coordinate descent (BCD) methods (Carreira-Perpiñán & Wang, 2014; Zhang & Brand, 2017; Lau et al., 2018; Askari et al., 2018; Gu et al., 2018) and alternating direction method of multipliers (ADMM) (Taylor et al., 2016; Zhang et al., 2016). The main reasons for the surge of attention of these two algorithms are twofold. One reason is that they are gradient-free, and thus are able to deal with non-differentiable nonlinearities and potentially avoid the vanishing gradient issue (Taylor et al., 2016; Zhang & Brand, 2017). As shown in Figure 1, it is observed that vanilla SGD fails to train a ten-hidden-layer MLPs while BCD still works and achieves a moderate accuracy within a few epochs. The other reason is that BCD and ADMM can be easily implemented in a distributed and parallel manner (Boyd et al., 2011; Mahajan et al., 2017), therefore in favour of distributed/decentralized scenarios.

The BCD methods currently adopted in DNN training run into two categories depending on the specific formulations of the objective functions, namely, the **two-splitting formulation** and **three-splitting formulation** (shown in 2.2 and 2.4), respectively. Examples of the two-splitting formulation include Carreira-Perpiñán & Wang (2014); Zhang & Brand (2017); Askari et al. (2018); Gu et al. (2018),

[*]Equal contribution [1]School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, Jiangxi, China [2]Department of Mathematics, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong [3]Department of Statistics, Northwestern University, Evanston, IL 60208, USA [4]Department of Mathematics, City University of Hong Kong, Kowloon, Hong Kong. Part of this work was done while Tim Tsz-Kit Lau was at Department of Mathematics, The Hong Kong University of Science and Technology. Correspondence to: Yuan Yao <yuany@ust.hk>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).



(a) BCD (b) SGD

Figure 1. Comparison of training and test accuracies of BCD and SGD for training ten-hidden-layer MLPs on the MNIST dataset. Refer to Appendix F for details of this experiment[2].

whilst Taylor et al. (2016); Lau et al. (2018) adopt the three-splitting formulation. Convergence studies of BCD methods appeared recently in more restricted settings. In Zhang & Brand (2017), a BCD method was suggested to solve the Tikhonov regularized deep neural network training problem using a lifting trick to avoid the computational hurdle imposed by ReLU. Its convergence was established through the framework of Xu & Yin (2013), where the *block multiconvexity*[3] and *differentiability* of the unregularized part of the objective function play central roles in the analysis. However, for other commonly used activations such as sigmoid, the convergence analysis of Xu & Yin (2013) cannot be directly applied since the *block multiconvexity* may be violated. Askari et al. (2018) and Gu et al. (2018) extended the lifting trick introduced by Zhang & Brand (2017) to deal with a class of strictly increasing and invertible activations, and then adapted BCD methods to solve the lifted DNN training models. However, no convergence guarantee was provided in both Askari et al. (2018) and Gu et al. (2018). Following the similar lifting trick as in Zhang & Brand (2017), Lau et al. (2018) proposed a proximal BCD based on the three-splitting formulation of the regularized DNN training problem with ReLU activation. The global convergence was also established through the analysis framework of Xu & Yin (2013). However, similar convergence results for other commonly used activation functions are still lacking.

In this paper, we aim to fill these gaps. Our main contribution is to provide a general methodology to establish the global convergence[4] of these BCD methods in the common DNN training settings, without requiring the *block multiconvexity* and *differentiability* assumptions as in Xu & Yin (2013). Instead, our key assumption is the Lipschitz continuity of the activation on any bounded set (see Assumption 1(b)). Specifically, Theorem 1 establishes the global convergence to a critical point at an $\mathcal{O}(1/k)$ rate of the BCD methods using the proximal strategy, while extensions to the prox-linear strategy for general losses are provided in Theorem 2 and to residual networks (ResNets) are shown in Theorem 3. Our assumptions are applicable to most cases appeared in the literature. Specifically in Theorem 1, if the loss function, activations, and convex regularizers are lower semicontinuous and either real-analytic (see Definition 1) or semialgebraic (see Definition 2), and the activations are Lipschitz continuous on any bounded set, then BCD converges to a critical point at an $\mathcal{O}(1/k)$ rate starting from any finite initialization, where k is the number of iterations. Note that these assumptions are satisfied by most commonly used DNN training models, where (a) the loss function can be any of the squared, logistic, hinge, exponential or cross-entropy losses, (b) the activation function can be any of ReLU, leaky ReLU, sigmoid, tanh, linear, polynomial, or softplus functions, and (c) the regularizer can be any of the squared ℓ_2 norm, squared Frobenius norm, the elementwise 1-norm, or the sum of squared Frobenuis norm and elementwise 1-norm (say, in the vector case, the elastic net by Zou & Hastie, 2005), or the indicator function of the nonnegative closed half space or a closed interval (see Proposition 1).

Our analysis is based on the Kurdyka-Łojasiewicz (KŁ) inequality (Łojasiewicz, 1993; Kurdyka, 1998) framework formulated in Attouch et al. (2013). However there are several different treatments compared to the state-of-the-art work (Xu & Yin, 2013) that enables us to achieve the general convergence guarantee aforementioned. According to Attouch et al. (2013, Theorem 2.9), the *sufficient descent*, *relative error* and *continuity* conditions, together with the *KŁ* assumption yield the global convergence of a nonconvex algorithm. In order to obtain the *sufficient descent* condition, we exploit the proximal strategy for all non-strongly convex subproblems (see Algorithm 2 and Lemma 1), without requiring the block multiconvexity assumption used in Xu & Yin (2013, Lemma 2.6). In order to establish the *relative error* condition, we use the Lipschitz continuity of the activation functions and perform some careful treatments on the specific updates of the BCD methods (see Lemma 2), without requiring the (locally) Lipschitz differentiability of the unregularized part as used in Xu & Yin (2013, Lemma 2.6). The *continuity* condition is established via the lower semicontinuity assumptions of the loss, activations and regularizers. The treatments of this paper are of their own value to the optimization community. The detailed comparisons between this paper and the existing literature can be found in Section 4.

The rest of this paper is organized as follows. Section 2 describes the BCD methods when adapted to the splitting formulations of DNN training problems. Section 3 estab-

[2]Codes available at: `https://github.com/timlautk/BCD-for-DNNs-PyTorch`.

[3]A function f with multi-block variables $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$ is called *block multiconvex* if it is convex with respect to each block variable when fixing the other blocks, and f is called *blockwise Lipschitz differentiable* if it is differentiable with respect to each block variable and its gradient is Lipschitz continuous while fixing the others.

[4]*Global convergence* refers to the case that the algorithm converges starting from any finite initialization.

lishes their global convergence results, followed by some extensions. Section 4 illustrates the key ideas of proof with some discussions. We conclude this paper in Section 5.

2. DNN training via BCD

In this section, we describe the specific forms of BCD involving both two- and three-splitting formulations.

2.1. DNN training with variable splitting

Consider N-layer feedforward neural networks with $N-1$ hidden layers of the neural networks. Particularly, let $d_i \in \mathbb{N}$ be the number of hidden units in the i-th hidden layer for $i = 1, \ldots, N-1$. Let d_0 and d_N be the number of units of input and output layers, respectively. Let $\boldsymbol{W}_i \in \mathbb{R}^{d_i \times d_{i-1}}$ be the weight matrix between the $(i-1)$-th layer and the i-th layer for any $i = 1, \ldots N$.[5] Let $\mathcal{Z} := \{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^n \subset \mathbb{R}^{d_0} \times \mathbb{R}^{d_N}$ be n samples, where $\boldsymbol{y}_j$'s are the one-hot vectors of labels. Denote $\mathcal{W} := \{\boldsymbol{W}_i\}_{i=1}^N$, $\boldsymbol{X} := (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d_0 \times n}$ and $\boldsymbol{Y} := (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n) \in \mathbb{R}^{d_N \times n}$. With the help of these notations, the DNN training problem can be formulated as the following empirical risk minimization:

$$\min_{\mathcal{W}} \mathcal{R}_n(\Phi(\boldsymbol{X}; \mathcal{W}), \boldsymbol{Y}), \tag{2.1}$$

where $\mathcal{R}_n\left(\Phi(\boldsymbol{X}; \mathcal{W}), \boldsymbol{Y}\right) := \frac{1}{n}\sum_{j=1}^n \ell\left(\Phi(\boldsymbol{x}_j; \mathcal{W}), \boldsymbol{y}_j\right)$, $\ell : \mathbb{R}^{d_N} \times \mathbb{R}^{d_N} \to \mathbb{R}_+ \cup \{0\}$ is some loss function, $\Phi(\boldsymbol{x}_j; \mathcal{W}) = \sigma_N(\boldsymbol{W}_N \sigma_{N-1}(\boldsymbol{W}_{N-1} \cdots \boldsymbol{W}_2 \sigma_1(\boldsymbol{W}_1 \boldsymbol{x}_j))$ is the neural network model with N layers and weights $\mathcal{W}$ and σ_i is the activation function of the i-th layer (generally, $\sigma_N \equiv \mathrm{Id}$, i.e., the identity function) and $\mathcal{R}_n$ is called the empirical risk (also known as the training loss).

Note that the DNN training model (2.1) is highly nonconvex as the variables are coupled via the deep neural network architecture, which brings many challenges for the design of efficient training algorithms and also its theoretical analysis. To make Problem (2.1) more computationally tractable, variable splitting is one of the most commonly used ways (Taylor et al., 2016; Zhang & Brand, 2017; Askari et al., 2018; Gu et al., 2018; Lau et al., 2018). The main idea of variable splitting is to transform a complicated problem (where the variables are coupled highly nonlinearly) into a relatively simpler one (where the variables are coupled much looser) via introducing some additional variables.

2.1.1. Two-splitting formulation.

Considering general deep neural network architectures, the DNN training problem can be naturally formulated as the following model (called **two-splitting formulation**)[6]:

$$\begin{aligned} \min_{\mathcal{W},\mathcal{V}} \mathcal{L}_0\left(\mathcal{W}, \mathcal{V}\right) &:= \mathcal{R}_n(\boldsymbol{V}_N; \boldsymbol{Y}) + \sum_{i=1}^N r_i(\boldsymbol{W}_i) + \sum_{i=1}^N s_i(\boldsymbol{V}_i) \\ \text{subject to} \quad & \boldsymbol{V}_i = \sigma_i(\boldsymbol{W}_i \boldsymbol{V}_{i-1}), \quad i = 1, \ldots, N, \end{aligned} \tag{2.2}$$

where $\mathcal{R}_n(\boldsymbol{V}_N; \boldsymbol{Y}) := \frac{1}{n}\sum_{j=1}^n \ell\left((\boldsymbol{V}_N)_{:j}, \boldsymbol{y}_j\right)$ denotes the empirical risk, $\mathcal{V} := \{\boldsymbol{V}_i\}_{i=1}^N$, $(\boldsymbol{V}_N)_{:j}$ is the j-th column of $\boldsymbol{V}_N$. In addition, r_i and s_i are extended-real-valued, nonnegative functions revealing the priors of the weight variable $\boldsymbol{W}_i$ and the state variable $\boldsymbol{V}_i$ (or the constraints on $\boldsymbol{W}_i$ and $\boldsymbol{V}_i$) for each $i = 1, \ldots N$, and define $\boldsymbol{V}_0 := \boldsymbol{X}$. In order to solve the two-splitting formulation (2.2), the following alternative minimization problem was suggested in the literature:

$$\min_{\mathcal{W},\mathcal{V}} \mathcal{L}\left(\mathcal{W}, \mathcal{V}\right) := \mathcal{L}_0\left(\mathcal{W}, \mathcal{V}\right) + \frac{\gamma}{2}\sum_{i=1}^N \|\boldsymbol{V}_i - \sigma_i(\boldsymbol{W}_i \boldsymbol{V}_{i-1})\|_F^2, \tag{2.3}$$

where $\gamma > 0$ is a hyperparameter[7].

The DNN training model (2.2) can be very general, where: (a) ℓ can be the squared, logistic, hinge, cross-entropy or other commonly used loss functions; (b) σ_i can be ReLU, leaky ReLU, sigmoid, linear, polynomial, softplus or other commonly used activation functions; (c) r_i can be the squared ℓ_2 norm, the ℓ_1 norm, the elastic net (Zou & Hastie, 2005), the indicator function of some nonempty closed convex set[8] (such as the nonnegative closed half space or a closed interval $[0, 1]$); (d) s_i can be the ℓ_1 norm (Ji et al., 2014), the indicator function of some convex set with simple projection (Zhang & Brand, 2017). Particularly, if there is no regularizer or constraint on $\boldsymbol{W}_i$ (or $\boldsymbol{V}_i$), then r_i (or s_i) can be zero.

The network architectures considered in this paper exhibit generality to various types of DNNs, including but not limited to the fully (or sparse) connected MLPs (Rosenblatt, 1961), convolutional neural networks (CNNs; Fukushima, 1980; LeCun et al., 1998) and residual neural networks (ResNets; He et al., 2016). For CNNs, the weight matrix $\boldsymbol{W}_i$ is sparse and shares some symmetry structures represented as permutation invariants, which are linear constraints and up to a linear reparameterization, so all the main results below are still valid.

Various existing BCD algorithms for DNN training (Carreira-Perpiñán & Wang, 2014; Zhang & Brand, 2017;

[5] To simplify notations, we regard the input and output layers as the 0-th and N-th layers, respectively, and absorb the bias of each layer into $\boldsymbol{W}_i$.

[6] Here we consider the regularized DNN training model. The model reduces to the original DNN training model (2.1) without regularization.

[7] In (2.5), we use a uniform hyperparameter γ for the sum of all quadratic terms for the simplicity of notation. In practice, γ can be different for each quadratic term and our proof still goes through.

[8] The indicator function $\iota_{\mathcal{C}}$ of a nonempty convex set $\mathcal{C}$ is defined as $\iota_{\mathcal{C}}(x) = 0$ if $x \in \mathcal{C}$ and $+\infty$ otherwise.

Askari et al., 2018; Gu et al., 2018) can be regarded as special cases in terms of the use of the two-splitting formulation (2.2). In fact, Carreira-Perpiñán & Wang (2014) considered a specific DNN training model with squared loss and sigmoid activation function, and proposed the method of auxiliary coordinate (MAC) based on the two-splitting formulation of DNN training (2.2), as a two-block BCD method with the weight variables $\mathcal{W}$ as one block and the state variables $\mathcal{V}$ as the other. For each block, a nonlinear least squares problem is solved by some iterative methods. Furthermore, Zhang & Brand (2017) proposed a BCD type method for DNN training with ReLU and squared loss. To avoid the computational hurdle imposed by ReLU, the DNN training model was relaxed to a smooth multiconvex formulation via lifting ReLU into a higher dimensional space (Zhang & Brand, 2017). Such a relaxed BCD is in fact a special case of two-splitting formulation (2.3) with $\sigma_i \equiv \mathrm{Id}, r_i \equiv 0, s_i(\boldsymbol{V}_i) = \iota_{\mathcal{X}}(\boldsymbol{V}_i), i = 1, \ldots, N$, where $\mathcal{X}$ is the nonnegative closed half-space with the same dimension of $\boldsymbol{V}_i$, while Askari et al. (2018) and Gu et al. (2018) extended such lifting trick to more general DNN training settings, of which the activation function can be not only ReLU, but also sigmoid and leaky ReLU. The general formulations studied in these two papers are also special cases of the two-splitting formulation with different σ_i, r_i and s_i for $i = 1, \ldots, N$.

2.1.2. Three-splitting formulation.

Note that the variables $\boldsymbol{W}_i$ and $\boldsymbol{V}_{i-1}$ are coupled by the nonlinear activation function in the i-th constraint of the two-splitting formulation (2.2), which may bring some difficulties and challenges for solving problem (2.2) efficiently, particularly, when the activation function is ReLU. Instead, the following **three-splitting formulation** was used in Taylor et al. (2016); Lau et al. (2018):

$$\min_{\mathcal{W},\mathcal{V},\mathcal{U}} \mathcal{L}_0(\mathcal{W},\mathcal{V}) \quad \text{subject to}$$
$$\boldsymbol{U}_i = \boldsymbol{W}_i \boldsymbol{V}_{i-1},\ \boldsymbol{V}_i = \sigma_i(\boldsymbol{U}_i), \quad i = 1, \ldots, N, \tag{2.4}$$

where $\mathcal{U} := \{\boldsymbol{U}_i\}_{i=1}^N$. From (2.4), the variables are coupled much more loosely, particularly for variables $\boldsymbol{W}_i$ and $\boldsymbol{V}_{i-1}$. As described later, such a three-splitting formulation can be beneficial to designing some more efficient methods, though N extra auxiliary variables $\boldsymbol{U}_i$'s are introduced. Similarly, the following alternative unconstrained problem was suggested in the literature:

$$\min_{\mathcal{W},\mathcal{V},\mathcal{U}} \overline{\mathcal{L}}(\mathcal{W},\mathcal{V},\mathcal{U}) := \mathcal{L}_0(\mathcal{W},\mathcal{V})$$
$$+ \frac{\gamma}{2} \sum_{i=1}^{N} \left[\|\boldsymbol{V}_i - \sigma_i(\boldsymbol{U}_i)\|_F^2 + \|\boldsymbol{U}_i - \boldsymbol{W}_i \boldsymbol{V}_{i-1}\|_F^2 \right]. \tag{2.5}$$

2.2. Description of BCD algorithms

In the following, we describe how to adapt the BCD method to Problems (2.3) and (2.5). The main idea of the BCD method of Gauss-Seidel type for a minimization problem with multi-block variables is to update all the variables cyclically while fixing the remaining blocks at their last updated values (Xu & Yin, 2013). In this paper, we consider the BCD method with the backward order (but not limited to this as discussed later) for the updates of variables, i.e., the variables are updated from the output layer to the input layer, and for each layer, we update the variables $\{\boldsymbol{V}_i, \boldsymbol{W}_i\}$ cyclically for Problem (2.3) as well as the variables $\{\boldsymbol{V}_i, \boldsymbol{U}_i, \boldsymbol{W}_i\}$ cyclically for Problem (2.5). Since $\sigma_N \equiv \mathrm{Id}$, the output layer is paid special attention. Particularly, for most blocks, we adopt the proximal update strategies for two major reasons: (1) To practically stabilize the training process; (2) To yield the desired "sufficient descent" property for theoretical justification. For each subproblem, we assume that its minimizer can be achieved. The BCD algorithms for Problems (2.3) and (2.5) can be summarized in Algorithms 1 and 2, respectively.

Algorithm 1 Two-splitting BCD for DNN Training (2.3)

Data: $\boldsymbol{X} \in \mathbb{R}^{d_0 \times n}$, $\boldsymbol{Y} \in \mathbb{R}^{d_N \times n}$
Initialization: $\{\boldsymbol{W}_i^0, \boldsymbol{V}_i^0\}_{i=1}^N$, $\boldsymbol{V}_0^k \equiv \boldsymbol{V}_0 := \boldsymbol{X}$
Parameters: $\gamma > 0, \alpha > 0$ [9]
for $k = 1, \ldots$ **do**
  $\boldsymbol{V}_N^k = \operatorname{argmin}_{\boldsymbol{V}_N} \{s_N(\boldsymbol{V}_N) + \mathcal{R}_n(\boldsymbol{V}_N; \boldsymbol{Y}) + \frac{\gamma}{2}\|\boldsymbol{V}_N - \boldsymbol{W}_N^{k-1}\boldsymbol{V}_{N-1}^{k-1}\|_F^2 + \frac{\alpha}{2}\|\boldsymbol{V}_N - \boldsymbol{V}_N^{k-1}\|_F^2\}$
  $\boldsymbol{W}_N^k = \operatorname{argmin}_{\boldsymbol{W}_N} \{r_N(\boldsymbol{W}_N) + \frac{\gamma}{2}\|\boldsymbol{V}_N^k - \boldsymbol{W}_N\boldsymbol{V}_{N-1}^{k-1}\|_F^2 + \frac{\alpha}{2}\|\boldsymbol{W}_N - \boldsymbol{W}_N^{k-1}\|_F^2\}$
  for $i = N-1, \ldots, 1$ **do**
    $\boldsymbol{V}_i^k = \operatorname{argmin}_{\boldsymbol{V}_i} \{s_i(\boldsymbol{V}_i) + \frac{\gamma}{2}\|\boldsymbol{V}_i - \sigma_i(\boldsymbol{W}_i^{k-1}\boldsymbol{V}_{i-1}^{k-1})\|_F^2 + \frac{\gamma}{2}\|\boldsymbol{V}_{i+1}^k - \sigma_{i+1}(\boldsymbol{W}_{i+1}^k\boldsymbol{V}_i)\|_F^2 + \frac{\alpha}{2}\|\boldsymbol{V}_i - \boldsymbol{V}_i^{k-1}\|_F^2\}$
    $\boldsymbol{W}_i^k = \operatorname{argmin}_{\boldsymbol{W}_i} \{r_i(\boldsymbol{W}_i) + \frac{\gamma}{2}\|\boldsymbol{V}_i^k - \sigma_i(\boldsymbol{W}_i\boldsymbol{V}_{i-1}^{k-1})\|_F^2 + \frac{\alpha}{2}\|\boldsymbol{W}_i - \boldsymbol{W}_i^{k-1}\|_F^2\}$
  end for
end for

One major merit of Algorithm 2 over Algorithm 1 is that in each subproblem, almost all updates are simple proximal updates[10] (or just least squares problems), which usually have closed form solutions to many commonly used DNNs, while a drawback of Algorithm 2 over Algorithm 1 is that more storage memory is required due to the introduction of additional variables $\{\boldsymbol{U}_i\}_{i=1}^N$. Some typical examples leading to the closed form solutions include: (a) r_i, s_i are 0 (i.e., no regularization), or the squared ℓ_2 norm (a.k.a. *weight decay*), or the indicator function of a nonempty closed con-

[9] In practice, γ and α can vary among blocks and our proof still goes through.

[10] For $\boldsymbol{V}_N^k$-update, we can regard $s_N(\boldsymbol{V}_N) + \mathcal{R}_n(\boldsymbol{V}_N; \boldsymbol{Y})$ as a new proximal function $\tilde{s}_N(\boldsymbol{V}_N)$.

Algorithm 2 Three-splitting BCD for DNN training (2.5)

Samples: $\boldsymbol{X} \in \mathbb{R}^{d_0 \times n}$, $\boldsymbol{Y} \in \mathbb{R}^{d_N \times n}$
Initialization: $\{\boldsymbol{W}_i^0, \boldsymbol{V}_i^0, \boldsymbol{U}_i^0\}_{i=1}^N$, $\boldsymbol{V}_0^k \equiv \boldsymbol{V}_0 := \boldsymbol{X}$
Parameters: $\gamma > 0$, $\alpha > 0$

for $k = 1, \ldots$ **do**
  $\boldsymbol{V}_N^k = \operatorname{argmin}_{\boldsymbol{V}_N} \{s_N(\boldsymbol{V}_N) + \mathcal{R}_n(\boldsymbol{V}_N; \boldsymbol{Y}) + \frac{\gamma}{2}\|\boldsymbol{V}_N - \boldsymbol{U}_N^{k-1}\|_F^2 + \frac{\alpha}{2}\|\boldsymbol{V}_N - \boldsymbol{V}_N^{k-1}\|_F^2\}$
  $\boldsymbol{U}_N^k = \operatorname{argmin}_{\boldsymbol{U}_N} \{\frac{\gamma}{2}\|\boldsymbol{V}_N^k - \boldsymbol{U}_N\|_F^2 + \frac{\gamma}{2}\|\boldsymbol{U}_N - \boldsymbol{W}_N^{k-1}\boldsymbol{V}_{N-1}^{k-1}\|_F^2\}$
  $\boldsymbol{W}_N^k = \operatorname{argmin}_{\boldsymbol{W}_N} \{r_N(\boldsymbol{W}_N) + \frac{\gamma}{2}\|\boldsymbol{U}_N^k - \boldsymbol{W}_N\boldsymbol{V}_{N-1}^{k-1}\|_F^2 + \frac{\alpha}{2}\|\boldsymbol{W}_N - \boldsymbol{W}_N^{k-1}\|_F^2\}$
  for $i = N-1, \ldots, 1$ **do**
    $\boldsymbol{V}_i^k = \operatorname{argmin}_{\boldsymbol{V}_i} \{s_i(\boldsymbol{V}_i) + \frac{\gamma}{2}\|\boldsymbol{V}_i - \sigma_i(\boldsymbol{U}_i^{k-1})\|_F^2 + \frac{\gamma}{2}\|\boldsymbol{U}_{i+1}^k - \boldsymbol{W}_{i+1}^k\boldsymbol{V}_i\|_F^2\}$
    $\boldsymbol{U}_i^k = \operatorname{argmin}_{\boldsymbol{U}_i} \{\frac{\gamma}{2}\|\boldsymbol{V}_i^k - \sigma_i(\boldsymbol{U}_i)\|_F^2 + \frac{\gamma}{2}\|\boldsymbol{U}_i - \boldsymbol{W}_i^{k-1}\boldsymbol{V}_{i-1}^{k-1}\|_F^2 + \frac{\alpha}{2}\|\boldsymbol{U}_i - \boldsymbol{U}_i^{k-1}\|_F^2\}$
    $\boldsymbol{W}_i^k = \operatorname{argmin}_{\boldsymbol{W}_i} \{r_i(\boldsymbol{W}_i) + \frac{\gamma}{2}\|\boldsymbol{U}_i^k - \boldsymbol{W}_i\boldsymbol{V}_{i-1}^{k-1}\|_F^2 + \frac{\alpha}{2}\|\boldsymbol{W}_i - \boldsymbol{W}_i^{k-1}\|_F^2\}$
  end for
end for

vex set with a simple projection like the nonnegative closed half space and the closed interval $[0, 1]$; (b) the loss function ℓ is the squared loss or hinge loss (see Lemma 14 in Appendix E.2); and (c) σ_i is ReLU (see Lemma 13 in Appendix E.1), leaky ReLU, or linear link function. For other cases in which r_i and s_i are the ℓ_1 norm, σ_i is the sigmoid function, and the loss ℓ is the logistic function, the associated subproblems can be also solved cheaply via some efficient existing methods. Discussions on specific implementations of these BCD methods can be referred to Appendix A.

3. Global convergence analysis of BCD

In this section, we establish the global convergence of both Algorithm 1 for Problem (2.3), and Algorithm 2 for Problem (2.5), followed by some extensions.

3.1. Main assumptions

First of all, we present our main assumptions, which involve the definitions of *real analytic* and *semialgebraic* functions.

Let $h : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ be an extended-real-valued function (respectively, $h : \mathbb{R}^p \rightrightarrows \mathbb{R}^q$ be a point-to-set mapping), its *graph* is defined by

$$\operatorname{Graph}(h) := \{(\boldsymbol{x}, y) \in \mathbb{R}^p \times \mathbb{R} : y = h(\boldsymbol{x})\},$$
$$(\text{resp. } \operatorname{Graph}(h) := \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^p \times \mathbb{R}^q : \boldsymbol{y} \in h(\boldsymbol{x})\}),$$

and its domain by $\operatorname{dom}(h) := \{\boldsymbol{x} \in \mathbb{R}^p : h(\boldsymbol{x}) < +\infty\}$ (resp. $\operatorname{dom}(h) := \{\boldsymbol{x} \in \mathbb{R}^p : h(\boldsymbol{x}) \neq \varnothing\}$). When h is a proper function, i.e., when $\operatorname{dom}(h) \neq \varnothing$, the set of its global minimizers (possibly empty) is denoted by

$$\operatorname{argmin} h := \{\boldsymbol{x} \in \mathbb{R}^p : h(\boldsymbol{x}) = \inf h\}.$$

Definition 1 (Real analytic) A function h with domain an open set $U \subset \mathbb{R}$ and range the set of either all real or complex numbers, is said to be **real analytic** at u if the function h may be represented by a convergent power series on some interval of positive radius centered at u, i.e., $h(x) = \sum_{j=0}^{\infty} \alpha_j (x-u)^j$, for some $\{\alpha_j\} \subset \mathbb{R}$. The function is said to be **real analytic** on $V \subset U$ if it is real analytic at each $u \in V$ (Krantz & Parks, 2002, Definition 1.1.5). The real analytic function f over $\mathbb{R}^p$ for some positive integer $p > 1$ can be defined similarly.

According to Krantz & Parks (2002), typical real analytic functions include polynomials, exponential functions, and the logarithm, trigonometric and power functions on any open set of their domains. One can verify whether a multivariable real function $h(\boldsymbol{x})$ on $\mathbb{R}^p$ is analytic by checking the analyticity of $g(t) := h(\boldsymbol{x} + t\boldsymbol{y})$ for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$.

Definition 2 (Semialgebraic)

(a) A set $\mathcal{D} \subset \mathbb{R}^p$ is called semialgebraic (Bochnak et al., 1998) if it can be represented as

$$\mathcal{D} = \bigcup_{i=1}^{s} \bigcap_{j=1}^{t} \{\boldsymbol{x} \in \mathbb{R}^p : P_{ij}(\boldsymbol{x}) = 0, Q_{ij}(\boldsymbol{x}) > 0\},$$

where P_{ij}, Q_{ij} are real polynomial functions for $1 \leq i \leq s, 1 \leq j \leq t$.

(b) A function $h : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ (resp. a point-to-set mapping $h : \mathbb{R}^p \rightrightarrows \mathbb{R}^q$) is called *semialgebraic* if its graph $\operatorname{Graph}(h)$ is semialgebraic.

According to Łojasiewicz (1965); Bochnak et al. (1998) and Shiota (1997, I.2.9, page 52), the class of semialgebraic sets are stable under the operation of finite union, finite intersection, Cartesian product or complementation. Some typical examples include polynomial functions, the indicator function of a semialgebraic set, and the Euclidean norm (Bochnak et al., 1998, page 26).

Assumption 1 *Suppose that*

(a) the loss function ℓ is a proper lower semicontinuous[11] *and nonnegative function,*

(b) the activation functions σ_i ($i = 1 \ldots, N-1$) are Lipschitz continuous on any bounded set,

(c) the regularizers r_i and s_i ($i = 1, \ldots, N$) are nonegative lower semicontinuous convex functions, and

(d) all these functions ℓ, σ_i, r_i and s_i ($i = 1, \ldots, N$) are either real analytic or semialgebraic, and continuous on their domains.

[11] A function $f : \mathcal{X} \to \mathbb{R}$ is called *lower semicontinuous* if $\liminf_{x \to x_0} f(x) \geq f(x_0)$ for any $x_0 \in \mathcal{X}$.

According to Krantz & Parks (2002); Łojasiewicz (1965); Bochnak et al. (1998) and Shiota (1997, I.2.9, page 52), most of the commonly used DNN training models (2.2) can be verified to satisfy Assumption 1 as shown in the following proposition, the proof of which is provided in Appendix B.

Proposition 1 *Examples satisfying Assumption 1 include:*

(a) ℓ *is the squared, logistic, hinge, or cross-entropy losses;*

(b) σ_i *is ReLU, leaky ReLU, sigmoid, hyperbolic tangent, linear, polynomial, or softplus activations;*

(c) r_i *and* s_i *are the squared* ℓ_2 *norm, the* ℓ_1 *norm, the elastic net, the indicator function of some nonempty closed convex set (such as the nonnegative closed half space, box set or a closed interval* $[0,1]$*), or* 0 *if no regularization.*

3.2. Main theorem

Under Assumption 1, we state our main theorem as follows.

Theorem 1 *Let* $\{\mathcal{Q}^k := \big(\{\boldsymbol{W}_i^k\}_{i=1}^N, \{\boldsymbol{V}_i^k\}_{i=1}^N\big)\}_{k\in\mathbb{N}}$ *and* $\{\mathcal{P}^k := \big(\{\boldsymbol{W}_i^k\}_{i=1}^N, \{\boldsymbol{V}_i^k\}_{i=1}^N, \{\boldsymbol{U}_i^k\}_{i=1}^N\big)\}_{k\in\mathbb{N}}$ *be the sequences generated by Algorithms 1 and 2, respectively. Suppose that Assumption 1 holds, and that one of the following conditions holds: (i) there exists a convergent subsequence* $\{\mathcal{Q}^{k_j}\}_{j\in\mathbb{N}}$ *(resp.* $\{\mathcal{P}^{k_j}\}_{j\in\mathbb{N}}$*); (ii)* r_i *is coercive*[12] *for any* $i = 1,\dots,N$*; (iii)* $\mathcal{L}$ *(resp.* $\overline{\mathcal{L}}$*) is coercive. Then for any* $\alpha > 0$*,* $\gamma > 0$ *and any finite initialization* $\mathcal{Q}^0$ *(resp.* $\mathcal{P}^0$*), the following hold*

(a) $\{\mathcal{L}(\mathcal{Q}^k)\}_{k\in\mathbb{N}}$ *(resp.* $\{\overline{\mathcal{L}}(\mathcal{P}^k)\}_{k\in\mathbb{N}}$*) converges to some* $\mathcal{L}^*$ *(resp.* $\overline{\mathcal{L}}^*$*).*

(b) $\{\mathcal{Q}^k\}_{k\in\mathbb{N}}$ *(resp.* $\{\mathcal{P}^k\}_{k\in\mathbb{N}}$*) converges to a critical point of* $\mathcal{L}$ *(resp.* $\overline{\mathcal{L}}$*).*

(c) $\frac{1}{K}\sum_{k=1}^K \|\boldsymbol{g}^k\|_F^2 \to 0$ *at the rate* $\mathcal{O}(1/K)$ *where* $\boldsymbol{g}^k \in \partial\mathcal{L}(\mathcal{Q}^k)$*. Similarly,* $\frac{1}{K}\sum_{k=1}^K \|\bar{\boldsymbol{g}}^k\|_F^2 \to 0$ *at the rate* $\mathcal{O}(1/K)$ *where* $\bar{\boldsymbol{g}}^k \in \partial\overline{\mathcal{L}}(\mathcal{P}^k)$*.*

Note that the DNN training problems (2.3) and (2.5) in this paper generally do not satisfy such a *Lispchitz differentiable* property, particularly, when ReLU activation is used. Compared to the existing literature, this theorem establishes the global convergence without the *block multiconvexity* and *Lipschitz differentiability* assumptions used in Xu & Yin (2013), which are often violated by the DNN training problems due to the nonlinearity of the activations.

[12] An extended-real-valued function $h : \mathbb{R}^p \to \mathbb{R}\cup\{+\infty\}$ is called coercive if and only if $h(\boldsymbol{x}) \to +\infty$ as $\|\boldsymbol{x}\| \to +\infty$.

3.3. Extensions

We extend the established convergence results to the BCD methods for general losses with the prox-linear strategy, and the BCD methods for training ResNets.

3.3.1. Extension to prox-linear

Note that in the $\boldsymbol{V}_N$-update of both Algorithms 1 and 2, the empirical risk is involved in the optimization problems. It is generally hard to obtain its closed-form solution except for some special cases such as the case that the loss is the square loss. For other smooth losses such as the logistic, cross-entropy, and exponential losses, we suggest using the following *prox-linear* update strategies, that is, for some parameter $\alpha > 0$, the $\boldsymbol{V}_N$-update in Algorithm 1 is

$$\boldsymbol{V}_N^k = \operatorname*{argmin}_{\boldsymbol{V}_N} \Big\{ s_N(\boldsymbol{V}_N) + \langle \nabla\mathcal{R}_n(\boldsymbol{V}_N^{k-1};\boldsymbol{Y}), \boldsymbol{V}_N - \boldsymbol{V}_N^{k-1}\rangle + \frac{\alpha}{2}\|\boldsymbol{V}_N - \boldsymbol{V}_N^{k-1}\|_F^2 + \frac{\gamma}{2}\|\boldsymbol{V}_N - \boldsymbol{W}_N^{k-1}\boldsymbol{V}_{N-1}^{k-1}\|_F^2 \Big\}, \tag{3.1}$$

and the $\boldsymbol{V}_N$-update in Algorithm 2 is

$$\boldsymbol{V}_N^k = \operatorname*{argmin}_{\boldsymbol{V}_N} \Big\{ s_N(\boldsymbol{V}_N) + \langle \nabla\mathcal{R}_n(\boldsymbol{V}_N^{k-1};\boldsymbol{Y}), \boldsymbol{V}_N - \boldsymbol{V}_N^{k-1}\rangle + \frac{\alpha}{2}\|\boldsymbol{V}_N - \boldsymbol{V}_N^{k-1}\|_F^2 + \frac{\gamma}{2}\|\boldsymbol{V}_N - \boldsymbol{U}_N^{k-1}\|_F^2 \Big\}. \tag{3.2}$$

From (3.1) and (3.2), when s_N is zero or its proximal operator can be easily computed, then $\boldsymbol{V}_N$-updates for both BCD algorithms can be implemented with explicit expressions. Therefore, the specific uses of these BCD methods are very flexible, mainly depending on users' understanding of their own problems.

The claims in Theorem 1 still hold for the prox-linear updates adopted for the $\boldsymbol{V}_N$-updates if the loss is smooth with Lipschitz continuous gradient, as stated in the following Theorem 2.

Theorem 2 (Global convergence for prox-linear update) *Consider adopting the prox-linear updates (3.1), (3.2) for the* $\boldsymbol{V}_N$*-subproblems in Algorithms 1 and 2, respectively. Under the conditions of Theorem 1, if further* $\nabla\mathcal{R}_n$ *is Lipschitz continuous with a Lipschitz constant* L_R *and* $\alpha > \max\left\{0, \frac{L_R-\gamma}{2}\right\}$*, then all claims in Theorem 1 still hold for both algorithms.*

The proof of Theorem 2 is presented in Appendix D. It establishes the global convergence of a BCD method for the commonly used DNN training models with nonlinear losses, such as logistic or cross-entropy losses, etc. Equipped with the prox-linear strategy, all updates of BCD can be implemented easily and allow large-scale distributed computations.

3.3.2. Extension to ResNet Training

In this section, we first adapt the BCD method to the residual networks (ResNets; He et al., 2016), and then extend the established convergence results of BCD to this case. Without loss of generality, similar to (2.2), we consider the following simplified ResNets training problem,

$$\min_{\mathcal{W},\mathcal{V}} \ \mathcal{R}_n(\boldsymbol{V}_N;\boldsymbol{Y}) + \sum_{i=1}^{N} r_i(\boldsymbol{W}_i) + \sum_{i=1}^{N} s_i(\boldsymbol{V}_i)$$
$$\text{s.t.} \quad \boldsymbol{V}_i - \boldsymbol{V}_{i-1} = \sigma_i(\boldsymbol{W}_i\boldsymbol{V}_{i-1}), \quad i = 1,\dots,N. \tag{3.3}$$

Since the ReLU activation is usually used in ResNets, we only consider the **three-splitting formulation** of (3.3):

$$\min_{\mathcal{W},\mathcal{V},\mathcal{U}} \ \mathcal{R}_n(\boldsymbol{V}_N;\boldsymbol{Y}) + \sum_{i=1}^{N} r_i(\boldsymbol{W}_i) + \sum_{i=1}^{N} s_i(\boldsymbol{V}_i)$$
$$\text{s.t.}\ \boldsymbol{U}_i = \boldsymbol{W}_i\boldsymbol{V}_{i-1},\ \boldsymbol{V}_i - \boldsymbol{V}_{i-1} = \sigma_i(\boldsymbol{U}_i),\ i = 1,\dots,N,$$

and then adapt BCD to the following minimization problem,

$$\min_{\mathcal{W},\mathcal{V},\mathcal{U}} \quad \overline{\mathcal{L}}_{\text{res}}(\mathcal{W},\mathcal{V},\mathcal{U}), \tag{3.4}$$

where $\mathcal{W} := \{\boldsymbol{W}_i\}_{i=1}^N$, $\mathcal{V} := \{\boldsymbol{V}_i\}_{i=1}^N$, $\mathcal{U} := \{\boldsymbol{U}_i\}_{i=1}^N$ as defined before, and

$$\overline{\mathcal{L}}_{\text{res}}(\mathcal{W},\mathcal{V},\mathcal{U}) := \mathcal{R}_n(\boldsymbol{V}_N;\boldsymbol{Y}) + \sum_{i=1}^{N} r_i(\boldsymbol{W}_i) + \sum_{i=1}^{N} s_i(\boldsymbol{V}_i)$$
$$+ \frac{\gamma}{2}\sum_{i=1}^{N}\left[\|\boldsymbol{V}_i - \boldsymbol{V}_{i-1} - \sigma_i(\boldsymbol{U}_i)\|_F^2 + \|\boldsymbol{U}_i - \boldsymbol{W}_i\boldsymbol{V}_{i-1}\|_F^2\right].$$

When applied to (3.4), we use the same update order of Algorithm 2 but slightly change the subproblems according to the objective $\overline{\mathcal{L}}_{\text{res}}$ in (3.4). The specific BCD algorithm for ResNets is presented in Algorithm 3 in Appendix D.

Similarly, we establish the convergence of BCD for the DNN training model with ResNets (3.4) as follows.

Theorem 3 (Convergence of BCD for ResNets) *Let $\{\{\boldsymbol{W}_i^k,\boldsymbol{V}_i^k,\boldsymbol{U}_i^k\}_{i=1}^N\}_{k\in\mathbb{N}}$ be a sequence generated by BCD for the DNN training model with ResNets (i.e., Algorithm 3). Let assumptions of Theorem 1 hold. Then all claims in Theorem 1 still hold for BCD with ResNets by replacing $\overline{\mathcal{L}}$ with $\overline{\mathcal{L}}_{\text{res}}$.*

Moreover, consider adopting the prox-linear update for the $\boldsymbol{V}_N$-subproblem in Algorithm 3, then under the assumptions of Theorem 2, all claims of Theorem 2 still hold for Algorithm 3.

The proof of this theorem is presented in Appendix D. ResNets is one of the most popular network architectures used in the deep learning community and has profound applications in computer vision. How to efficiently train ResNets is thus very important, especially since it is not of a fully-connected structure. This theorem, for the first time, shows that the BCD method might be a good candidate for the training of ResNets with global convergence guarantee.

4. Keystones and discussions

In this section, we present the keystones of our proofs followed by some discussions.

4.1. Main ideas of proofs

Our proofs follow the analysis framework formulated in Attouch et al. (2013), where the establishments of the *sufficient descent* and *relative error* conditions and the verifications of the *continuity* condition and *KŁ* property of the objective function are the four key ingredients. In order to establish the *sufficient descent* and *relative error* properties, two kinds of assumptions, namely, **(a)** multiconvexity and differentiability assumption, and **(b)** (blockwise) Lipschitz differentiability assumption on the unregularized part of objective function are commonly used in the literature, where Xu & Yin (2013) mainly used **assumption (a)**, and the literature (Attouch et al., 2013; Xu & Yin, 2017; Bolte et al., 2014) mainly used **assumption (b)**. Note that in our cases, the unregularized part of $\mathcal{L}$ in (2.3),

$$\mathcal{R}_n(\boldsymbol{V}_N;\boldsymbol{Y}) + \frac{\gamma}{2}\sum_{i=1}^{N}\|\boldsymbol{V}_i - \sigma_i(\boldsymbol{W}_i\boldsymbol{V}_{i-1})\|_F^2,$$

and that of $\overline{\mathcal{L}}$ in (2.5),

$$\mathcal{R}_n(\boldsymbol{V}_N;\boldsymbol{Y}) + \frac{\gamma}{2}\sum_{i=1}^{N}\left[\|\boldsymbol{V}_i - \sigma_i(\boldsymbol{U}_i)\|_F^2 + \|\boldsymbol{U}_i - \boldsymbol{W}_i\boldsymbol{V}_{i-1}\|_F^2\right]$$

usually do not satisfy any of the block multiconvexity and differentiability assumptions (i.e., **assumption (a)**), and the blockwise Lipschitz differentiability assumption (i.e., **assumption (b)**). For instance, when σ_i is ReLU or leaky ReLU, the functions $\|\boldsymbol{V}_i - \sigma_i(\boldsymbol{W}_i\boldsymbol{V}_{i-1})\|_F^2$ and $\|\boldsymbol{V}_i - \sigma_i(\boldsymbol{U}_i)\|_F^2$ are non-differentiable and nonconvex with respect to $\boldsymbol{W}_i$-block and $\boldsymbol{U}_i$-block, respectively.

In order to overcome these challenges, in this paper, we first exploit the proximal strategies for all the non-strongly convex subproblems (see Algorithm 2) to cheaply obtain the desired sufficient descent property (see Lemma 1), and then take advantage of the Lipschitz continuity of the activations as well as the specific splitting formulations to yield the desired relative error property (see Lemma 2). Below we present these two key lemmas, while leaving other details in Appendix (where the verification of the KŁ property for the concerned DNN training models satisfying Assumption 1

can be referred to Proposition 2 in Appendix C.1, and the verification of the *continuity* condition is shown by (C.19) in Appendix C.3.2). Based on Lemmas 1 and 2, Proposition 2 and (C.19), we prove Theorem 1 according to Attouch et al. (2013, Theorem 2.9), with details shown in Appendix C.

4.2. Sufficient descent lemma

We state the established sufficient descent lemma as follows.

Lemma 1 (Sufficient descent) *Let $\{\mathcal{P}^k\}_{k\in\mathbb{N}}$ be a sequence generated by the BCD method (Algorithm 2). Then, under the assumptions of Theorem 1,*

$$\overline{\mathcal{L}}(\mathcal{P}^k) \leq \overline{\mathcal{L}}(\mathcal{P}^{k-1}) - a\|\mathcal{P}^k - \mathcal{P}^{k-1}\|_F^2, \tag{4.1}$$

for some constant $a > 0$ specified in the proof.

From Lemma 1, the Lagrangian sequence $\{\overline{\mathcal{L}}(\mathcal{P}^k)\}$ is monotonically decreasing, and the descent quantity of each iterate can be lower bounded by the discrepancy between the current iterate and its previous iterate. This lemma is crucial for the global convergence of a nonconvex algorithm. It tells at least the following four important items: **(i)** $\{\overline{\mathcal{L}}(\mathcal{P}^k)\}_{k\in\mathbb{N}}$ is convergent if $\overline{\mathcal{L}}$ is lower bounded; **(ii)** $\{\mathcal{P}^k\}_{k\in\mathbb{N}}$ itself is bounded if $\overline{\mathcal{L}}$ is coercive and $\mathcal{P}^0$ is finite; **(iii)** $\{\mathcal{P}^k\}_{k\in\mathbb{N}}$ is square summable, i.e., $\sum_{k=1}^{\infty}\|\mathcal{P}^k - \mathcal{P}^{k-1}\|_F^2 < \infty$, implying its asymptotic regularity, i.e., $\|\mathcal{P}^k - \mathcal{P}^{k-1}\|_F \to 0$ as $k \to \infty$; and **(iv)** $\frac{1}{K}\sum_{k=1}^{K}\|\mathcal{P}^k - \mathcal{P}^{k-1}\|_F^2 \to 0$ at a rate of $\mathcal{O}(1/K)$. Leveraging Lemma 1, we can establish the global convergence (i.e., the whole sequence convergence) of BCD in DNN training settings. In contrast, Davis et al. (2019) only establish the subsequence convergence of SGD in DNN training settings. Such a gap between the subsequence convergence of SGD in Davis et al. (2019) and the whole sequence convergence of BCD in this paper exists mainly because SGD can only achieve the descent property but not the sufficient descent property.

It can be noted from Lemma 1 that neither *multiconvexity and differentiability* nor *Lipschitz differentiability* assumptions are imposed on the DNN training models to yield this lemma, as required in the literature (Xu & Yin, 2013; Attouch et al., 2013; Xu & Yin, 2017; Bolte et al., 2014). Instead, we mainly exploit the proximal strategy for all nonstrongly convex subproblems in Algorithm 2 to establish this lemma.

4.3. Relative error lemma

We now present the obtained relative error lemma.

Lemma 2 (Relative error) *Under the conditions of Theorem 1, let $\mathcal{B}$ be an upper bound of $\mathcal{P}^{k-1}$ and $\mathcal{P}^k$ for any positive integer k, $L_{\mathcal{B}}$ be a uniform Lipschitz constant of σ_i on the bounded set $\{\mathcal{P} : \|\mathcal{P}\|_F \leq \mathcal{B}\}$. Then for any positive integer k, it holds that,*

$$\|\bar{\boldsymbol{g}}^k\|_F \leq \bar{b}\|\mathcal{P}^k - \mathcal{P}^{k-1}\|_F, \quad \bar{\boldsymbol{g}}^k \in \partial\overline{\mathcal{L}}(\mathcal{P}^k)$$

for some constant $\bar{b} > 0$ specified later in the proof, where

$$\partial\overline{\mathcal{L}}(\mathcal{P}^k) := (\{\partial_{\boldsymbol{W}_i}\overline{\mathcal{L}}\}_{i=1}^N, \{\partial_{\boldsymbol{V}_i}\overline{\mathcal{L}}\}_{i=1}^N, \{\partial_{\boldsymbol{U}_i}\overline{\mathcal{L}}\}_{i=1}^N)(\mathcal{P}^k).$$

Lemma 2 shows that the subgradient sequence of the Lagrangian is upper bounded by the discrepancy between the current and previous iterates. Together with the asymptotic regularity of $\{\mathcal{P}^k\}_{k\in\mathbb{N}}$ yielded by Lemma 1, Lemma 2 shows the critical point convergence. Also, together with the claim **(iv)** implied by Lemma 1, namely, the $\mathcal{O}(1/K)$ rate of convergence of $\frac{1}{K}\sum_{k=1}^{K}\|\mathcal{P}^k - \mathcal{P}^{k-1}\|_F^2 \to 0$, Lemma 2 yields the $\mathcal{O}(1/K)$ rate of convergence (to a critical point) of BCD, i.e., $\frac{1}{K}\sum_{k=1}^{K}\|\bar{\boldsymbol{g}}^k\|_F \to 0$ at the rate of $\mathcal{O}(1/K)$.

From Lemma 2, both differentiability and (blockwise) Lipschitz differentiability assumptions are not imposed. Instead, we only use the Lipschitz continuity (on any bounded set) of the activations, which is a very mild and natural condition satisfied by most commonly used activation functions. In order to achieve this lemma, we also need to do some special treatments on the specific updates of BCD algorithms as demonstrated in Appendix C.3.1.

5. Conclusion

The empirical efficiency of BCD methods in deep neural network (DNN) training has been demonstrated in the literature. However, the theoretical understanding of their convergence is still very limited and it lacks a general framework due to the fact that DNN training is a highly nonconvex problem. In this paper, we fill this void by providing a general methodology to establish the global convergence of the BCD methods for a class of DNN training models, which encompasses most of the commonly used BCD methods in the literature as special cases. Under some mild assumptions, we establish the global convergence at a rate of $\mathcal{O}(1/k)$, with k being the number of iterations, to a critical point of the DNN training models with several variable splittings. Our theory is also extended to residual networks with general losses which have Lipschitz continuous gradients. Such work may lay a theoretical foundation of BCD methods for their applications to deep learning.

Acknowledgments

The work of Jinshan Zeng is supported in part by the National Natural Science Foundation (NNSF) of China (No. 61603162, 61876074). The work of Tim Tsz-Kit Lau is supported by the University Fellowship for first-year Ph.D. students by The Graduate School at Northwestern University. The work of Shao-Bo Lin is supported in part by

the NNSF of China (No. 61876133). The work of Yuan Yao is supported in part by HKRGC grant 16303817, 973 Program of China (No. 2015CB85600), NNSF of China (No. 61370004, 11421110001), as well as grants from Tencent AI Lab, Si Family Foundation, Baidu BDI, and Microsoft Research-Asia. The authors thank Prof. Jian Peng from UIUC for helpful discussions.

References

Askari, A., Negiar, G., Sambharya, R., and EI Ghaoui, L. Lifted neural networks. *arXiv preprint arXiv:1805.01532*, 2018.

Attouch, H. and Bolte, J. On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116(1–2):5–16, 2009.

Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010.

Attouch, H., Bolte, J., and Svaiter, B. F. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

Bochnak, J., Coste, M., and Roy, M.-F. *Real algebraic geometry*, volume 3. Ergeb. Math. Grenzgeb. Springer-Verlag, Berlin, 1998.

Bolte, J., Daniilidis, A., and Lewis, A. The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17:1205–1223, 2007a.

Bolte, J., Daniilidis, A., Lewis, A., and Shiota, M. Clark subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18(2):556–572, 2007b.

Bolte, J., Sabach, S., and Teboulle, M. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1):459–494, 2014.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. In *Foundations and Trends® in Machine Learning*, volume 3, pp. 1–122. Now Publishers, Inc., 2011.

Carreira-Perpiñán, M. and Wang, W. Distributed optimization of deeply nested systems. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

Davis, D., Drusvyatskiy, D., Kakade, S., and Lee, J. D. Stochastic subgradient method converges on tame functions. *Foundations of Computational Mathematics*, to appear, 2019.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Fukushima, K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Gu, F., Askari, A., and El Ghaoui, L. Fenchel lifted networks: a Lagrange relaxation of neural network training. *arXiv preprint arXiv:1811.08039*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Ji, N.-N., Zhang, J.-S., and Zhang, C.-X. A sparse-response deep belief network based on rate distortion theory. *Pattern Recognition*, 47(9):3179–3191, 2014.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Krantz, S. and Parks, H. R. *A primer of real analytic functions*. Birkhäuser, second edition, 2002.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Kurdyka, K. On gradients of functions definable in o-minimal structures. *Annales de l'institut Fourier*, 48 (3):769–783, 1998.

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

Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., and Ng, A. Y. On optimization methods for deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2011.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In Montavon, G., Orr, G. B., and Müller, K.-R. (eds.), *Neural Networks: Tricks of the Trade: Second Edition*, pp. 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

Liao, Q. and Poggio, T. Theory II: Landscape of the empirical risk in deep learning. *arXiv preprint arXiv:1703.09833*, 2017.

Łojasiewicz, S. Une propriété topologique des sous-ensembles analytiques réels. In: Les Équations aux dérivées partielles. *Éditions du centre National de la Recherche Scientifique, Paris*, pp. 87–89, 1963.

Łojasiewicz, S. *Ensembles semi-analytiques*. Institut des Hautes Etudes Scientifiques, 1965.

Łojasiewicz, S. Sur la geometrie semi-et sous-analytique. *Annales de l'institut Fourier*, 43(5):1575–1595, 1993.

Mahajan, D., Keerthi, S. S., and Sundararajan, S. A distributed block coordinate descent method for training ℓ_1 regularized linear classifiers. *Journal of Machine Learning Research*, 18:1–35, 2017.

Mordukhovich, B. S. *Variational analysis and generalized differentiation I: Basic Theory*. Springer, 2006.

Nesterov, Y. *Lectures on convex optimization*. Springer, second edition, 2018.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of Adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2018.

Robbins, H. and Monro, S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3): 400–407, 1951.

Rockafellar, R. T. and Wets, R. J.-B. *Variational analysis*. Grundlehren Math. Wiss. 317, Springer-Verlag, New York, 1998.

Rosenblatt, F. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Spartan Books, Washington DC, 1961.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Sainath, T. N., Mohamed, A.-r., Kingsbury, B., and Ramabhadran, B. Deep convolutional neural networks for LVCSR. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

Shiota, M. *Geometry of subanalytic and semialgebraic sets*, volume 150 of *Progress in Mathematics*. Birkhäuser, Boston, 1997.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., and Goldstein, T. Training neural networks without gradients: A scalable ADMM approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

Tieleman, T. and Hinton, G. Lecture 6.5-RMSProp: Divide the gradient by running average of its recent magnitude. *Coursera Neural Networks for Machine Learning*, 4(2), 2012.

Wang, Y., Yin, W., and Zeng, J. Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63, 2019.

Xu, Y. and Yin, W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

Xu, Y. and Yin, W. A globally convergent algorithm for nonconvex optimization based on block coordinate update. *Journal of Scientific Computing*, 72:700–734, 2017.

Zhang, Z. and Brand, M. Convergent block coordinate descent for training Tikhonov regularized deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Zhang, Z., Chen, Y., and Saligrama, V. Efficient training of very deep neural networks for supervised hashing. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.